



Model Order Determination: A Multi-Objective Evolutionary Neural Network Scheme

Rui Ligeiro¹ · Joao Paulo Carvalho²

Received: 16 December 2021 / Accepted: 6 April 2022 / Published online: 26 April 2022
© The Author(s), under exclusive licence to Springer Nature Singapore Pte Ltd 2022

Abstract

Knowing the relevant amount of information needed to correctly predict patterns present in a data series is an important question to address. This is known as the problem of model order determination and is not adequately solved yet. This paper proposes to determine model order based on the scheme of a topological dynamic neural network that examines the dimensionality of the non-linear function that reconstructs the process. The novelty of the approach lies in the use of a neural network optimized by an evolutionary multi-objective selection mechanism that is capable of determining model order and performing robust estimations based on joint minimization of the length of the past and of the prediction error. Since the size of the input layer of the neural network is associated with the model order, the results show that the model order can be determined by the Pareto-optimal solutions that emerge from the optimization process. The practicality of the model is demonstrated on three univariate examples extracted from a time series database.

Keywords Model order · Artificial neural networks · Evolutionary neural networks · Multi-objective optimization · Time series forecasting · Pareto-optimal curve

Introduction

Systems with causal associations and temporal dependence are of great importance in many natural and social phenomena. In many of these systems, their most important features can be isolated by a collection of observations made sequentially through time. It is possible to discover some structures that emerge from the models behind these systems. Knowing the model order, i.e., the relevant amount of information needed by an observer to correctly predict patterns present in a data series, is therefore an essential question to address. Especially when long memory or long-range dependence arise in the analysis of data, indicating that the behavior of a time-dependent process denotes statistically significant correlations across very large time scales. Simply put, model

order refers to the optimal number of inputs to be used by a model, which is an important pre-processing stage, especially when using large-scale dynamic systems [1].

Time series forecasting is a well-known procedure for projecting future behaviors of such systems based on the present and past observations of time-ordered information. It is an extremely topical area of signal processing research that involves fields like statistics, economics, physics, signal processing and computational intelligence, just to name a few. It has many applications in areas as diverse as finance, economy, medicine, energy, weather, geology, geophysics, biology, among others [2–5].

Classical methods for time series forecasting, including autoregressive integrated moving average (ARIMA) [6], Kalman filter [7] and Bayesian Forecasting [8] are still effective and very useful today. More recently, deep learning [9, 10] has brought improvements in time series forecasting, especially in the analysis of large-scale data. Long-Short Term Memory (LSTM) [11] networks are a gated architecture variant of Recurrent Neural Networks (RNN), that has been successfully used for handling sequential data. However, LSTMs have a complex structure, need massive training samples, and show limitations when facing time interval irregularities in data. LSTMs are also intimately

✉ Rui Ligeiro
rui.ligeiro@gmail.com

Joao Paulo Carvalho
joao.carvalho@inesc-id.pt

¹ Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

² INESC-ID/Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

connected with the determination of the model order, since one of LSTMs' most important hyperparameters, the number of layers, is optimal when its value corresponds to that of the model order.

When modelling time series data, an observer cannot obtain full information about the underlying system with some degree of uncertainty being nearly unavoidable. Despite these facts, patterns often arise in the analysis of spatial or time series data, indicating that the behavior of a time-dependent process denotes statistically significant correlations across short or even very large time scales. During (or prior to) the construction of any such model, the amount of past memory or spatial range needed should be accurately determined and such determination should not be hindered by the assumptions implied in the model.

The dichotomy between the past length and the forecasting error accepted by the observer is central when tackling signal processing and pattern recognition problems. Particularly in the minimal classification of relevant patterns raised by the process's causal structures, whose complexity can be, informally, related to the model order. The overall picture is simple: increasing the length of the past should lead to lower prediction errors, but at the unbearable cost of affecting model complexity, whereas the short-length-past should provide tractable models but with higher error. Even knowing that these premises are not always true, it is clear that the underlying objectives have optimal solutions (minimization of the length of the past and minimization of the error) that conflict with each other, and for that reason, this problem can be seen as a multi-objective optimization problem (see Fig. 1). In this work we propose to combine Layered Feedforward Neural Networks (FNN) and Genetic Algorithms (GA) to address the problem of the determination of the model order of a generic ordered series. We address it as a multi-objective optimization problem in time series forecasting—minimization of the length of the past and minimization of the error: given a generic time series, we use a feedforward neural network classifier (FNN) optimized using a GA, to make short-term predictions of its future behavior based on the present and past observations. The goal is not to use the (NN) as an optimal forecaster: the NN is only used as a tool to determine the model order

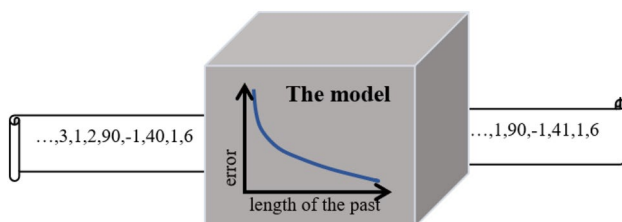


Fig. 1 Illustration of a generic model. The estimation considers its error predictions and the length of the past

size. Subsequently, any forecasting method can be optimized based on this knowledge.

Since the size of the input layer of the optimal FNN corresponds to the de facto model order, the proposed multi-objective optimization problem allows us to determine the model order based on the resulting Pareto front.

This article is organized as follows: “[Research Background](#)” reviews the most important aspects of Evolutionary Artificial Neural Networks (EANNs), Lipschitz indexes, Multi-Objective Optimization Problems (MOPs), as well as other topics related to this study. In “[Model Order as a Multi-Objective Optimization Problem](#)” the problem of model order determination is discussed as a multi-objective optimization problem, and next, in “[Methodology](#)” the proposed method is detailed. Applications and the obtained results are presented in “[Applications](#)” and summarized in “[Conclusion](#)”, the conclusion.

Research Background

For the benefit of the reader, we present a brief preliminary overview describing some research background concerning Lipschitz indexes, model structure, EANNs, MOPs and other topics related to this study.

A. Lipschitz Indexes

In this work, Model Order refers to the optimal number of relevant past inputs used in a model. One of the best-known methods used to determine model order is the Lipschitz indexes [19]. This method does not depend on the use of any approximation method or particular structure because it is based on the continuity property of the non-linear function that represents the input–output model of a continuous dynamic system. To determine the order, it is necessary to calculate the Lipschitz quotients; as a ratio of the absolute values between two output values and the distance between two points in the input space. Next, the Lipschitz indexes are determined by the weighted geometric mean of the largest Lipschitz quotients calculated from the set of input–output pairs in the data.

More formally, from a set of candidate points given, the goal is to estimate the relevant inputs of the general nonlinear input–output formulation:

$$y = f(x) = f(x_1, x_2, \dots, x_n) \quad (1)$$

It is necessary to reconstruct the nonlinear function $f(x)$ only from data, i.e., from the input–output data pairs (x_j, y_2) . The Lipschitz quotients are then determined by:

$$q_{ij} = \frac{|y_i - y_j|}{|x_i - x_j|}, (i \neq j) \text{ and } i, j = 1, 2, \dots, N \quad (2)$$

where N is the number of samples in the dataset, x is the input and y is the output of the system.

From the Lipschitz quotients, the indexes, also called Lipschitz numbers, are calculated as:

$$Q(s) \equiv \left[\sqrt{s} \prod_{k=1}^r q^{(s)}(k) \right]^{\frac{1}{r}} \quad (3)$$

where r is a positive integer number, recommended to be one or two percent of the sample size, and $q^{(s)}(k)$ is the k^{th} largest Lipschitz quotient among all $q_{ij}^{(s)}$. To determine the order, it is necessary to find the point where the slope of the $Q(s)$ curve changes from decreasing in a pronounced way to decreasing in a less pronounced way (examples can be seen later on in “Methodology”).

Applications of Lipschitz indexes have been the object of study in several works [20–22]. The main drawback of the Lipschitz indexes method for model order determination is its high sensitivity to noisy data. The problem of order determination needs more attention from researchers; however, this approach can still have an important role in the settlement of the number of past inputs to use.

B. Evolutionary Artificial Neural Networks

Neural network architecture design and parameter setting is very much a human expert’s endeavor that requires significant effort and tedious trial-and-error procedures.

FNNs are universal approximators, good at fitting non-linear functions and for that reason have been used successfully to solve time series forecasting problems [12–14]. The accuracy of the approximation depends not only on the complexity of the problem but also on the architecture of the network, as well as, the amount and quality of labeled data available for training [40]. The backpropagation algorithm [15] is a gradient search technique conventionally applied in neural networks during the supervised training phase. However, gradient search techniques tend to get trapped at local minima. Drawbacks of the local search backpropagation, as well as its mitigation techniques have been widely discussed in literature ([16–18] and [45] are good examples among many others).

Constructive algorithms have been successfully used in the past for good network architecture. These methods start with a small network structure and then add additional hidden units and weights until a satisfactory solution is found. They always search for small network solutions first. In theory, they are thus more computationally economical than

pruning algorithms, in which most of the training time is spent on networks larger than necessary [23].

EANNs refer to a special class of artificial neural networks (ANNs) in which evolution is another fundamental form of adaptation in addition to learning [24]. EANNs employ evolutionary computation techniques like Genetic Algorithms (GA) [25] to perform various tasks, such as connection weight training, architecture design, learning rule adaptation and neuron activation function selection among others. ANN performance depends on its ability to learn, which is directly related with the network topology, training algorithm, as well as the neuron activation function.

One distinct feature of EANNs is their adaptability to a dynamic environment. EAs are not problem-specific and adjust an entire population of candidate solutions to an objective without direct intervention. They are not susceptible to initial conditions of training and are particularly useful for dealing with large complex problems which generate many local optima; they are less likely to be trapped in local optimal solutions than traditional gradient-based search algorithms. However, EANNs can be very slow when dealing with problems that require multiple processing layers to learn representations of data with multiple levels of abstraction.

Convolutional neural networks (CNN) are complex network models that have recently demonstrated impressive performance in image classification and object detection. Pruning deep learning models is important for achieving generalization improvements [26–30]. In a typical deep-learning system, there may be hundreds of millions of adjustable weights and hundreds of labelled examples with which to train the machine (see [31] for more information about deep learning and convolutional networks).

C. Multi-Objective Evolutionary Algorithms

In general, problems with multiple objectives have a set of optimal solutions known as Pareto-optimal solutions, instead of a single optimal solution. Often, multi-objective problems are converted into a single objective in order to simplify its analysis. However, this approach does not find multiple Pareto-optimal solutions present in the feasible objective space in a single simulation run, while, on the other hand, multi-objective evolutionary algorithms (MOEAs) are good at finding multiple solutions in one simulation run. Evolutionary Algorithms are powerful stochastic search and optimization algorithms developed by taking inspiration from the biological process of evolution.

MOEAs work with a population of multiple solutions processed in every generation and, for that reason, it is possible to choose one, or more solution(s) from the resulting set by using relevant complementary information and considerations. The work of Fonseca and Fleming’s [32, 33], Coello

Coello [34] and Deb and colleague's [35–38] stood out in the use of evolutionary algorithms applied to MOPs and deserve special attention from researchers in future studies.

Model Order as a Multi-Objective Optimization Problem

When considering a time series signal without knowing the dynamical system that represents its temporal evolution, the problem to solve is:

-By looking at the data, what is the relevant (minimum) amount of information necessary to accurately predict future values? Or in other words, what is the model order for robust estimation?

In our approach, the problem can be reformulated in a more concrete way:

-What is the relevant number of neurons in the input layer of the network that guarantees optimal forecasting?

A main concern in supervised learning is to avoid overfitting. To achieve good generalization, it is necessary to control the complexity of the learned function. If it is too complex, it may incorporate irrelevant properties of the dataset on which it is trained (overfitting), thus performing poorly on future data. On the other hand, a very simple function may not be able to capture the main behavior of the underlying relationship (underfitting) [39].

For a chosen input length, too many neurons in the hidden layer may cause overfitting, while too few neurons may cause underfitting. Moreover, and not forgetting the role of the hidden layer dynamic structure, if we increase the number of neurons in the input layer we may tend to predict successfully and naturally the opposite is also true, i.e., short-length-past generates higher estimation error. These objectives have different and conflicting optimal solutions (minimization of the length of the past and minimization of the error), which means that we can treat this problem as a multi-objective optimization problem. It is important to mention at this point that the answer to the problem may not be confined to a single model order but to multiple Pareto-optimal model orders present in the feasible objective space. Also, the method used here for determination of the model order is independent from the data series as well as the technique chosen to make the forecast.

Methodology

In the proposed learning system, the signal to be learned is a sequence of discrete time sampling points, spread at an interval Δ in the form:

$$x[n] = x[n\Delta], n = 1, 2, \dots, N \quad (4)$$

where n is a discrete time step and N is the length of the series. For simplicity we consider the set \vec{X} to be a vector, where the x values are real values in the form:

$$\vec{X} = \{x[n-j], x[n-j+1], x[n-j+2], \dots, x[n-j+p]\} \\ J = 0, 1, \dots, N \quad (5)$$

where j is decreased to a discrete time step n to indicate the starting index of the sliding window and p is its length. The sliding window moves along the signal gathering all values of the vector for processing. The elements of the vector \vec{X} are inputs to a feedforward neural network with three layers: input, hidden and output, with $\{W\}$ denoting the full set of connection strengths and $Y(\vec{X}) = f_w(\vec{X})$ the output.

The aim is to choose a set of connection strengths that sustain the prediction:

$$\hat{f}_w(\vec{X}, t-1) = x[t] \quad (6)$$

Our approach consists of the application of a genetic algorithm using a multi-objective selection mechanism to evolve a neural network that determines a model order with robust estimations (lower error). At each iteration the fitness over a population of candidates is evaluated (individual ANNs that are candidate solutions to the problem) based on performance on the problem. If a candidate is evaluated favorably then it is allowed to propagate via producing offspring, otherwise the candidate is discarded. The output of the dynamic network is the model order and a non-linear function that starts off with a large error when making predictions that learns to make better estimations with increasing iterations.

The scheme starts with a population of predetermined random networks containing an equally distributed variable number of input neurons; the same number of ANNs with one input neuron, the same number of ANNs with two input neurons, and so on. The maximum number of input neurons is recommended to be the sample size divided by ten. The number of neurons in the hidden layer are random but less than a predefined value. After scoring each network by its ability to predict, a multi-objective selection mechanism is then applied as follows: first, minimization of the error—a worthy parent network is selected via a tournament selection method (the best one is chosen between a pair of networks); then minimization of the length of the past—the other parent network is chosen via the roulette wheel selection method among the networks with input neurons less than a value that starts large and progressively decreases during the iterations. The algorithm terminates after a finite number of iterations.

The most important steps to evolve the network are as follows:

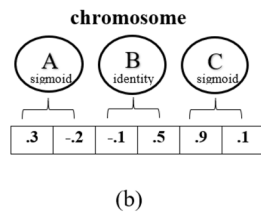
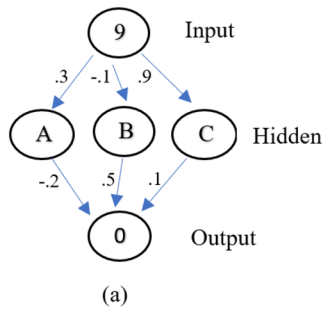


Fig. 2 A network with three neurons in the hidden layer (a) and its chromosome representation (b)

Initialize Population

The population is created with N random networks (chromosomes) containing an equally distributed variable number of input neurons, a hidden layer with a random number of neurons that is less than a predefined limiting value, and a single output neuron. Only one output neuron is necessary as it represents the single prediction value for an input vector. For a better understanding, see Fig. 2 for an illustration of the network and its chromosome representation. Weights are randomly generated between 0 and 1 and one of the following activation functions is chosen with equal probability: identity, sigmoid, radial basis and reLU (see [41, 42] for details and references therein). It is very unlikely that these networks have a good score without learning, therefore the next phase is very important.

Learning and Training

Evaluate Population

Every network is evaluated by comparing its estimation with the target value. The Mean Squared Error (MSE) loss function measures the error of the networks for all possible input vectors in the series. As expected, the decay of the loss function is more pronounced and with a longer tail for data with less uncertainty. Chromosomes are chosen for production of the next generation through their fitness value. In elitism selection mechanisms the best chromosomes are preferred against others. Chromosomes with higher fitness values are more likely to be chosen than others in selection methods like roulette wheel or tournament selection. Here, a parent network is selected via a tournament selection method among all networks. The other parent network is chosen via the roulette wheel selection method among the networks with input neurons less than a value that starts large and progressively decreases during the iterations.

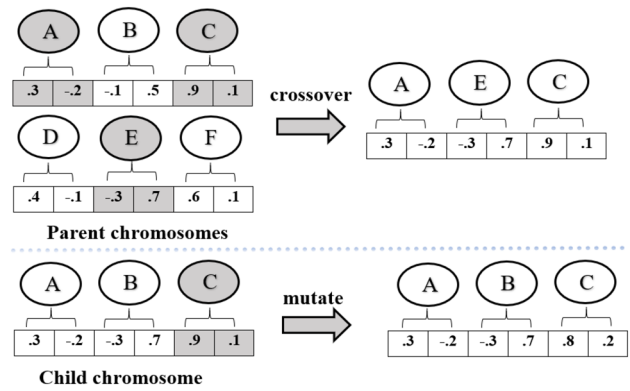


Fig. 3 Crossover (a) and mutate (b) operation examples

Reproduce Population

The Crossover operation is applied to the parents selected in Step 2. The generated child is formed by exchanging some portion of the elements of a pair of fittest parents (weights, activation function). This combination can be achieved by copying continuous or alternate parts of both parents. Mutation intends to provoke a small perturbation in the weights of the produced children, thus, the exploration of new regions on the search space (see Fig. 3). After a fixed number of I iterations the algorithm terminates, otherwise it jumps to step 2.1 (every chromosome is evaluated by comparing its estimation with the target value) and the process repeats again, starting a new iteration loop with the resulting population of chromosomes.

One of the key advantages of this approach is the dynamic character of the network. The fact that in the same network we can have neurons with different activation functions, increases the fitting and non-linear nature of the derived function. Also, implicit in the algorithm is the assumption that we are in the basis of attraction of the global minimum. GA permits this by performing an evolution strategy with mutation and crossover operators that explore new regions of the search space, as well as by keeping some of the non-top networks: this helps to find potentially successful combinations between worse-performers and top-performers. Algorithm 1—Multi-Objective Evolutionary ANN is outlined next in pseudo code.

Algorithm 1 - Multi-Objective Evolutionary ANN

Input: D dataset, N population size, M max number of iterations, H max number of neurons in the hidden layer
Output: P list of ANNs

```

1:  $i = 0$ 
2:  $D' = \text{normalize}(D)$ 
3:  $p = \text{length}(D') / 10$ 
4:  $l = p$ 
5:  $Q = \phi$ 
6:  $P = \text{createPopulation}(N, p, H)$ 
7: while  $i < M$  do
8:   if  $\text{mod}(i, M / p) == 0$  then
9:      $l = l - 1$ ;
10:  end if
11:   $P' = \text{getPopulationWithPastLimit}(P, l)$ 
12:  while  $j < N$  and  $P' \neq \phi$  do
13:     $\text{evaluate}(P, D')$ 
14:     $p1 = \text{tournamentSelection}(P)$ 
15:     $p2 = \text{rouletteWheelSelection}(P')$ 
16:     $c = \text{reproduce}(p1, p2)$ 
17:     $c = \text{mutate}(c)$ 
18:     $Q \cup \{c\}$ 
19:     $j = j + 1$ 
20:  end while
21:   $P = Q$ 
22:   $i = i + 1$ 
23: end while

```

Let us now discuss the implications of introducing an operator that selects chromosomes with a number of inputs less than a fixed limit that decreases its value during the algorithm iterations (see lines 11 and 15 of Algorithm 1). Initially, candidate solutions are spread all over the search space in a uniform manner. During the iterations the model gradually reduces the search space aiming to explore shorter-length-past solutions, preserving however diversity by considering solutions throughout the whole population of chromosomes (see line 14 of the Algorithm 1). Therefore, by combining the population of all chromosomes P with a subset of chromosomes P' that gradually decreases the length of the past, the model converges to a set of Pareto-optimal solutions containing the networks capable of determining model order and performing robust estimations.

Applications

In this section we show how the model order emerges from the method proposed here.

Since there is no other well-known method with insights for the problem under analysis in this study, results obtained from the scheme are compared with the corresponding ones reached by the Lipschitz index calculation.

Now that we are equipped with a system that predicts patterns, it is important to assess its performance considering

two important factors: a variable length of the past and the corresponding error obtained in prediction.

For simplicity, we only explore univariate data collected from real datasets obtained from the Time Series Data Library [43]. We are looking for datasets with numeric values only, however the same approach can be applied with other alphabets. A set of three datasets were chosen to span a variety of scenarios; prominent trends with seasonal, cyclic and periodic fluctuations of data, and irregular behavior of data. Although we have applied the approach to other datasets of the library, for simplicity, we show here relevant partial results, deemed sufficient to support our proposal.

A simulation explores solutions in all generations of chromosomes. For the input parameters, several combinations of value variations were evaluated to optimize the performance and computational cost of the algorithm. It was noted that parameter values affect the prediction and convergence speed of the model. To reach the desired degree of accuracy, the parameters were initialized with the following empirical values:

Parameter	Value
Population size (N)	1000
Max number of generations (M)	5000
Max number of neurons in the hidden layer (H)	10

All datasets were first normalized by removing its mean value within a certain time period (8-time steps), and then bounded between 0 and 1 by using the following basic function to each value in the series:

$$\text{Normalize}(X) = \frac{(X - \min)}{(\max - \min)} \quad (7)$$

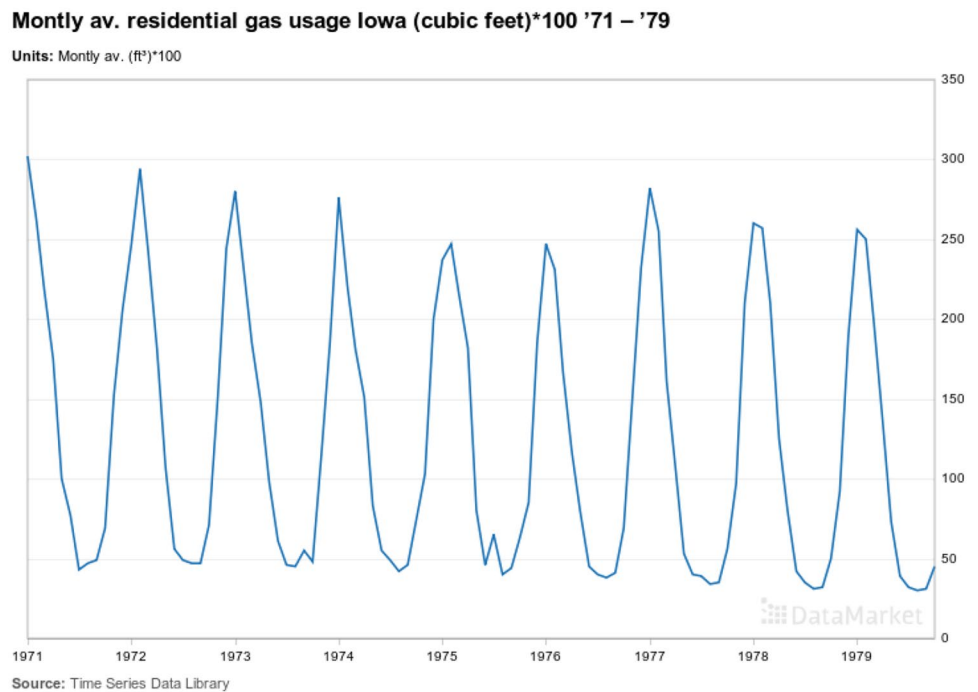
where min and max are minimum and maximum value in the data series respectively.

For each isolated prediction (see Fig. 2), the error is calculated by the difference between the observed value and the corresponding forecast value of the model. The predicting error of the dataset is determined by the average value of all isolated predictions per iteration.

Before the normalization of the signal, the Lipschitz index is calculated for every dataset with the NNSYSID toolbox [44]. Results are compared with those obtained by our multi-objective model.

To assess the model's performance, many simulations were done with similar/comparable results. For demonstration we randomly pick the results of a simulation for each dataset under study. The results can be visualized in a graph with the past length and corresponding error obtained. In each graph, a curve determines the Pareto-optimal solutions for the dataset in the feasible objective region. Note

Fig. 4 Monthly av. residential gas usage in Iowa (cubic feet) × 100'71 –'79



that since we are using a stochastic model, some simulation results contain small deviations, however the main behavior of the model remains unchanged. With this in mind, and considering the importance of properly evaluating the model’s consistency, a Box and Whiskers plot was generated for each data series. The graph shows the Pareto-optimal solutions obtained for 10 simulations; the top and bottom of the whisker line indicate the maximum and minimum observations of the distribution, the ends of the box are the upper and lower quartiles, the median is marked by the horizontal line inside the box, the cross represents the mean value and the dots represent outliers.

All experiments were conducted on a Laptop with an Intel® Core™ i7—2.50 GHz CPU, and with 16 GB of RAM.

Dataset A. Montly av. Residential Gas Usage in Iowa

This dataset refers to the monthly average (cubic feet × 100) residential gas usage in Iowa from 1971 until 1979.

It shows a strong seasonality. The peaks of gas usage present in Fig. 4 are related with seasons of cold periods.

Figure 5 shows the curve of the Lipschitz index (see “Research Background” for details) of the dataset A, for varying lag space from 1 to 10. As expected, the Lipschitz index obtained is decreasing as the lag space is increasing. No sharp breakpoint is clearly perceptible, nevertheless 7 seems to be the best choice for the optimal model order candidate.

Next, Fig. 6 shows the graph of the results obtained by a simulation of the model in the dataset A. Since this dataset has 106 values, then the maximum past allowed by the

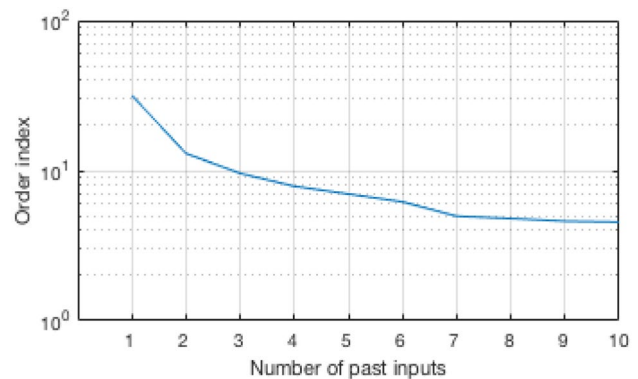


Fig. 5 Order index vs number of past inputs for the dataset A

model is 10 that is approximately a decimal part of the size of the series. At the end of the simulation, the 1000 networks are arranged in groups of 1, 2, 3, 4, 6 and 8 inputs. Networks with 6 inputs have the highest occurrences. It is easy to see that some of them exhibit good performance while others do not. This happens mostly due to the maintenance of non-top networks during evolution, as well as, the mutations and crossover operations performed in order to explore new regions of the search space. Also, as already discussed before, changes in the number of neurons in the hidden layers may drastically affect performance of the networks. In simple terms, variations on network topologies during the learning/training phase may result in large performance adjustments.

Fig. 6 Pareto-optimal solutions of the model in the dataset A

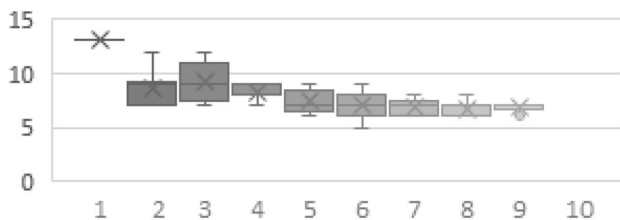
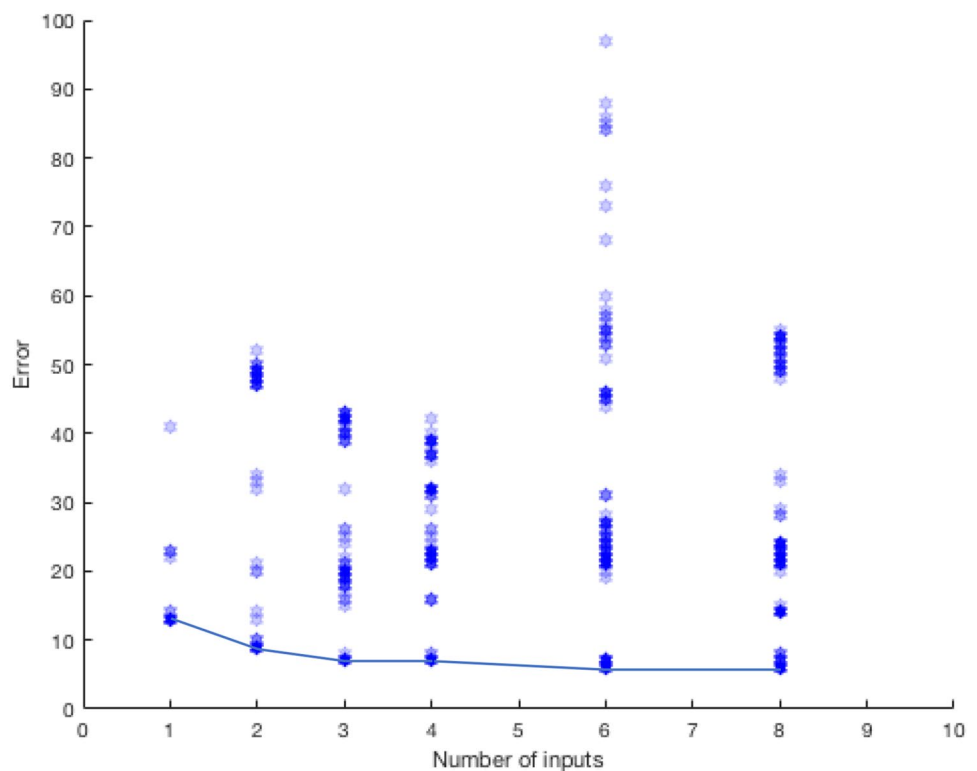


Fig. 7 Box and Whiskers plot containing the Pareto-optimal solutions obtained during 10 simulations of the model in the dataset A. The X axis indicates the past length and the Y axis the error

The Pareto-optimal curve is obtained by joining the optimal points of the networks by considering its number of inputs. By analyzing the shape of the Pareto-optimal curve in the objective space, one can see that no past length stands out significantly from all the others. However, a model order of 3, 4, 6, or 8 seems to be very good candidates for this dataset when not accepting an error less than 10.

Four order values were discarded as good solutions to be proposed from the model (5, 7, 9 and 10).

Figure 7 depicts the Box and Whiskers plot containing the Pareto-optimal solutions obtained during 10 simulations of the model in the dataset A. The NN with the best performance of all simulations has 6 inputs and gives an error value of 5. Now that more simulations were performed, networks with 5, 7 and 9 input neurons could also contribute to the Pareto-optimal curve. The mean value is very close to

the median and interquartile range, especially lower in the results obtained with networks with past size 1 and 9.

Considering the mean or median values of the plot it can easily be seen that it shows similar performance to the one obtained by the graph in Fig. 7.

Comparing the results with those obtained from the Lipschitz index, one can easily see that our model provides more insightful information. Furthermore, the model order obtained by the Lipschitz index is included in the Pareto-optimal solutions depicted in the Box and Whiskers plot.

Dataset B. Quarterly Production of Gas in Australia

This dataset refers to the quarterly production of gas (million megajoules) in Australia, from July 1956 until September 1994. This dataset (see Fig. 8) shows an increasing trend of consumption and some seasonal oscillations.

In Fig. 9 one can examine the Lipschitz index curve that determines the model order for the dataset B. There is a definite sharp breakpoint at value 4 indicating the number of optimal past inputs.

Like the previous case, the graph of the dataset B (see Fig. 10) reveals that some networks perform badly in contrast with others that show very good results. Naturally, the results are directly related to the classification of data under study.

Fig. 8 Quarterly production of Gas in Australia: million megajoules. Includes natural gas from July 1989. Mar 1956 – Sep 1994

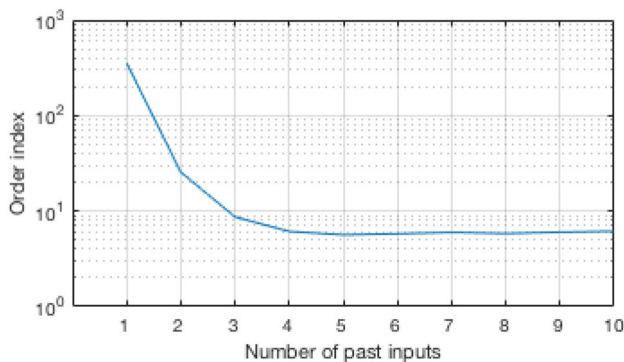
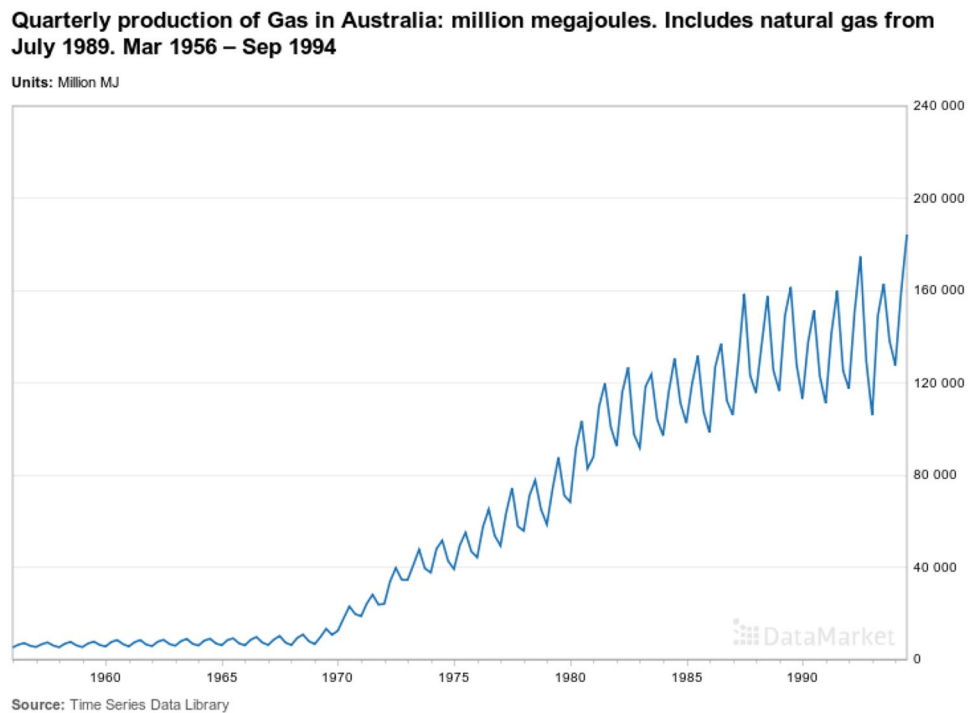


Fig. 9 Order index vs number of past inputs for the dataset B

Since this dataset has 155 values, then the maximum past allowed by the model is 15 that is approximately a decimal part of the size of the series.

Seven of the fifteen possible solutions were not considered as valid from the model (2, 4, 8, 10, 13, 14 and 15). When analyzing the shape of the Pareto-optimal curve, it is easy to see that the past length 5 stands out significantly from all the others. However, like we have seen before, this plot represents a single simulation and for that reason it is important to verify if this behavior can be generalized or not.

The Box and Whiskers plot containing the Pareto-optimal solutions obtained during 10 simulations of the dataset B can be seen in Fig. 11. There are six NNs with different input lengths that obtained the best performance (error 4) at

least one time. Now that more simulations were performed, networks with 2, 4, 8, 10, 13 and 14 input neurons could also contribute to the Pareto-optimal curve. Still, there is no single ANN with past length 15. There are cases where the mean value differs from the median and the interquartile range is especially lower on the results obtained with networks with past size 1, 11, 13 and 14. Interquartile is lower due to two distinct reasons: homogeneity of results (in every simulation ANNs with past length 1 gave result 11) and lack of contribution (ANNs with past length 11, 13 and 14 only had one or two Pareto-optimal solutions during the 10 simulations).

Again, like the previous example, when joining the mean or median points of the plot it can easily be seen that it shows a curve shape like the one obtained by the graph in Fig. 10. Note that due to the stochastic behavior of the model, the imaginary curve presents a sharp point on input 4 instead of 5 and there are differences for inputs higher than 10 that should be ignored since they are irrelevant for understanding this phenomenon, and to avoid erroneous interpretations.

Again, the model order obtained by the Lipschitz index is included in the Pareto-optimal solutions depicted in the Box and Whiskers plot.

Dataset C. Montly Percipitation (in mm) in London

This dataset refers to the monthly precipitation (in mm) in London, United Kingdom, from Jan 1983 until April 1994 (Fig. 12). This dataset does not show any visible trend.

Fig. 10 Pareto-optimal solutions of the model in the dataset B

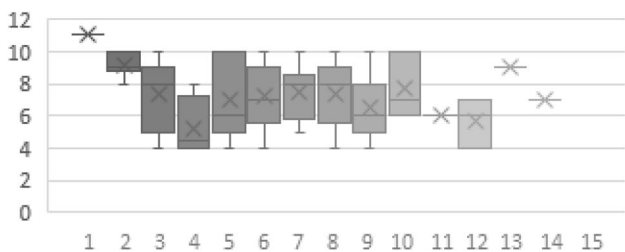
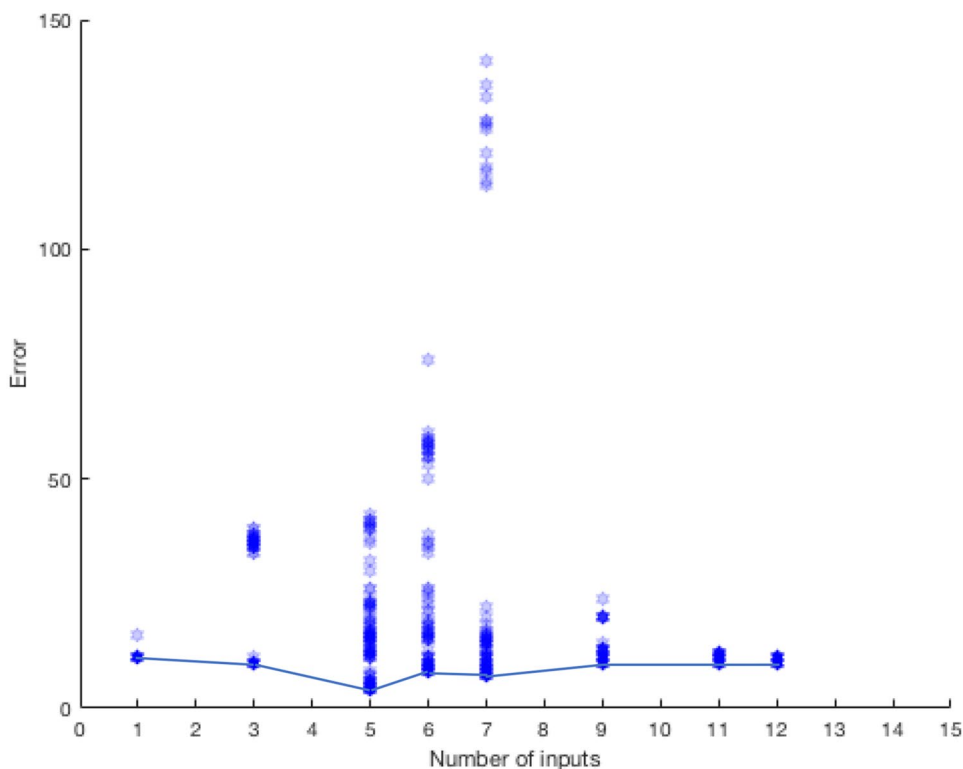


Fig. 11 Box and Whiskers plot containing the Pareto-optimal solutions obtained during 10 simulations of the model in the dataset B. The X axis indicates the past length and the Y axis the error

Like the previous two examples the same method was applied to calculate the number of relevant inputs for dataset C. Similarly, to dataset A, no number of inputs clearly stands out from the others as the optimal order. As already mentioned, the main problem with the Lipschitz method is that if noise is present in data, the estimation of the order will be improper, or the graph has no sharp breakpoint. By looking at Fig. 13, 5 seems to be the most likely optimal order for the dataset.

The graph of the dataset C depicted in Fig. 14 indicates homogeneous performance.

Since this dataset has 136 values, then the maximum past allowed by the model is 13, that is approximately a decimal part of the size of the series. At the end of the simulation, the networks are arranged only in two groups

of 1 and 9 inputs. The model performs worse here (minimum error 11) than in the previous datasets because this data has no relevant structures or patterns. There are more occurrences with better performance of networks with 9 inputs in the population than with 1 input. There is no single network in the plot with any other input count. In contrast with the two Pareto-optimal graphs analyzed before for the previous datasets no network performs badly. A model order of 9 seems to be the best candidate for this dataset.

Figure 15 depicts the Box and Whiskers plot of the dataset C containing the Pareto-optimal solutions obtained during 10 simulations. The trend of the graph is totally different from the previous datasets. Despite there being more than one input that could contribute to the Pareto-optimal curve, in all simulations each input gave a constant error value: 12 for input 1 and 11 for inputs 5, 7, 8, 9 and 10. All these values are good candidates for model order, especially inputs 5 and 7 that occurred more often in the simulations.

As shown in the previous two datasets, the model order obtained by the Lipschitz index is included in the Pareto-optimal solutions depicted in the Box and Whiskers plot.

Fig. 12 Monthly precipitation (in mm), Jan 1983–April 1994. London, United Kingdom

Monthly precipitation (in mm), Jan 1983 – April 1994. London, United Kingdom.

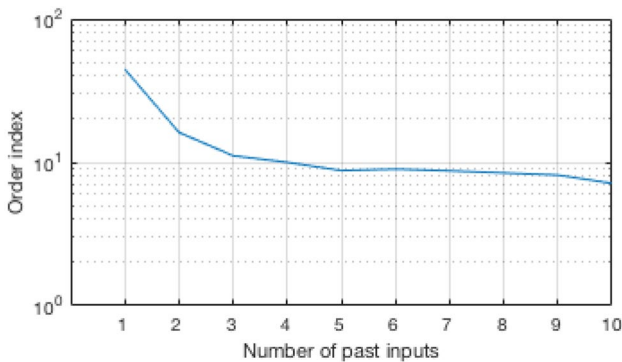
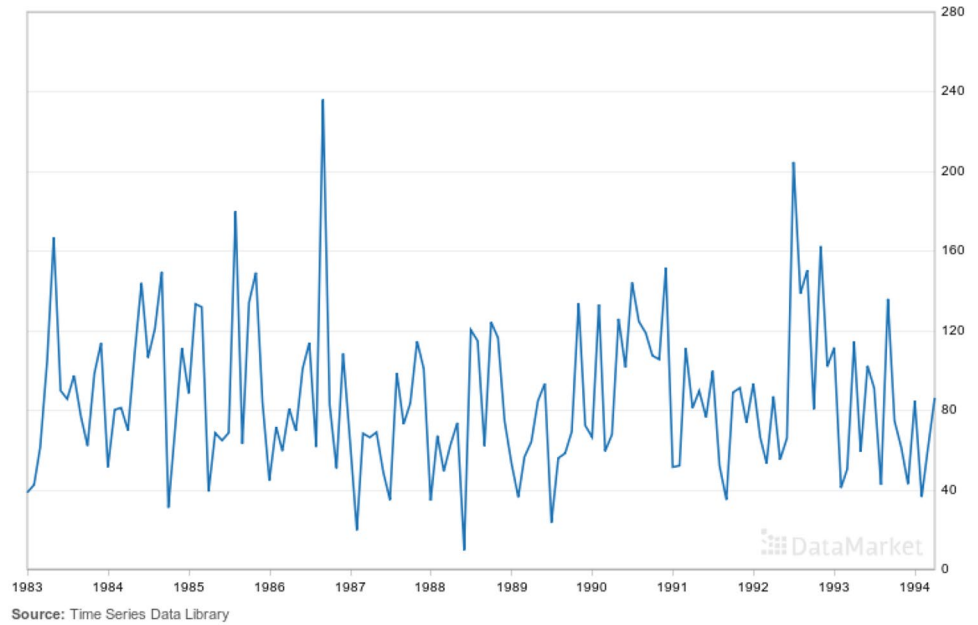


Fig. 13 Order index vs number of past inputs for the dataset C

Conclusion

The main goal of this research is to develop a scheme that determines the model order of a dataset. The Lipschitz index method involves moderate computational effort, but

it is very sensitive to noise and data distribution. In contrast with the Lipschitz index method, the general idea behind our approach lies in formulating this issue as a multi-objective optimization problem.

In our model, the parameter setting method is clear and easy to apply to any dataset under evaluation however it requires a relatively high computational cost for training the networks. Results show that model order can be determined by the Pareto-optimal solutions that emerge from a population of neural networks optimized by an evolutionary multi-objective selection mechanism.

When analyzing the several order candidates that belong to the Pareto-optimal solutions of a dataset, one can have an accurate idea of the performance variations performed by different topologies. For future research we plan to integrate our model with LSTM, in order to optimize its structure, as well as, to help find the best window length for input without requiring a huge amount of training samples.

Fig. 14 Pareto-optimal solutions of the model in the dataset C

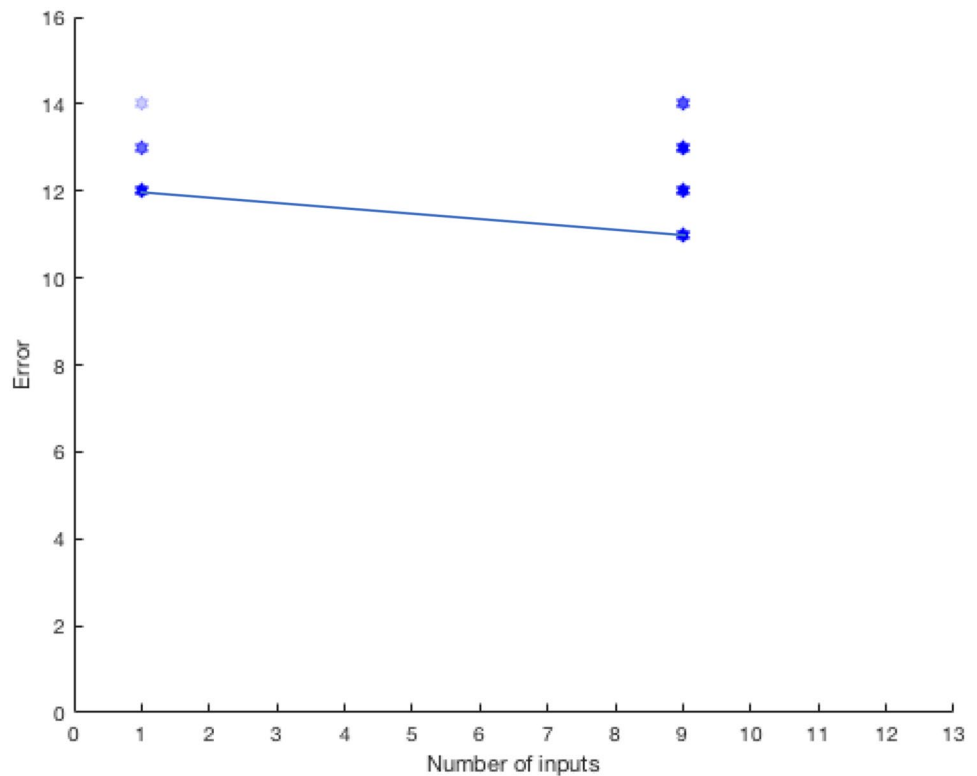


Fig. 15 Box and Whiskers plot containing the Pareto-optimal solutions obtained during 10 simulations of the model in the dataset C. The X axis indicates the past length and the Y axis the error

Acknowledgements This work was partially supported by Fundação para a Ciência e a Tecnologia (FCT), through Portuguese national funds Ref. UIDB/50021/2020.

Declarations

Conflict of Interest The authors declare that there is no conflict of interest.

References

- Anders U, Korn O. Model selection in neural networks. *Neural Netw.* 1999;12(2):309–23.
- De Jan G G, Hyndman RJ. 25 years of time series forecasting. *Int J Forecast.* 2006;22(3):443–73.
- Cyril V, et al. Machine learning methods for solar radiation forecasting: a review. *Renew Energy.* 2017;105:569–82.
- Taylor SJ, Letham B. Forecasting at scale. *Am Stat.* 2018;72(1):37–45.
- Georgia P, Tyralis H, Koutsoyiannis D. Predictability of monthly temperature and precipitation using automatic time series forecasting methods. *Acta Geophys.* 2018;66(4):807–31.
- Box GEP, et al. *Time series analysis: forecasting and control.* Hoboken: John Wiley & Sons; 2015.
- Kalman RE. A new approach to linear filtering and prediction problems. *J Basic Eng.* 1960;82(1):35–45 (**Transaction of the ASME**).
- Andy P, West M, Harrison J. *Applied Bayesian forecasting and time series analysis.* Boston: Springer; 1994. (**Chapman and Hall/CRC**).
- Zhang CY, Chen CLP, Gan M, Chen L. Predictive deep Boltzmann machine for multiperiod wind speed forecasting. *IEEE Trans Sustain Energy.* 2015;6:1416–25.
- Busseti E, Osband I, Wong S. “Deep learning for time series modelling.” Technical report. Stanford: Stanford University; 2012.
- Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput.* 1997;9(8):1735–80.
- Peter ZG, Qi M. Neural network forecasting for seasonal and trend time series. *Eur J Oper Res.* 2005;160(2):501–14.
- Khashei M, Bijari M. A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. *Appl Soft Comput.* 2011;11(2):2664–75.
- da Matteo F, Yao X. Short-term load forecasting with neural network ensembles: a comparative study [application notes]. *IEEE Comput Intell Mag.* 2011;6(3):47–56.
- Rumelhart DE, Hinton GE, Williams RJ. Learning internal representation by error propagation. In: Rumelhart DE, McClelland

- JL, editors. *Parallel distributed processing*. Cambridge: MIT Press; 1986. (Vol 1, Chap 8).
16. Lippmann RP. Pattern classification using neural networks. *IEEE Commun Mag.* 1989;27(11):47–50.
 17. Kamruzzaman Joarder, Syed Mahfuzul Aziz. A note on activation function in multilayer feedforward learning. *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*. Vol. 1. IEEE, 2002.
 18. Principe JC, Chen B. Universal approximation with convex optimization: gimmick or reality? [Discussion forum]. *IEEE Comput Intell Mag.* 2015;10(2):68–77.
 19. He Xiangdong, Haruhiko Asada 1993 A new method for identifying orders of input-output models for nonlinear dynamic systems. *American Control Conference*. IEEE, 1993.
 20. Bomberger JD, Seborg DE. Determination of model order for NARX models directly from input-output data. *J Process Control.* 1998;8(5–6):459–68.
 21. Sragner L, Horvath G 2003 "Improved model order estimation for nonlinear dynamic systems." *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings of the Second IEEE International Workshop on*. IEEE, 2003.
 22. Dário B, Carvalho JP, Morgado-Dias F. Comparing different solutions for forecasting the energy production of a wind farm. *Neural Comput Appl.* 2020;32(20):15825–33.
 23. Kwok T-Y, Yeung D-Y. Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Trans Neural Netw.* 1997;8(3):630–45.
 24. Yao X. Evolutionary artificial neural networks. *Int J Neural Syst.* 1993;4(03):203–22.
 25. Holland JH. *Adaptation in natural and artificial systems*. Ann Arbor: Univ. of Michigan Press; 1975.
 26. Zheng Q, et al. Improvement of generalization ability of deep CNN via implicit regularization in two-stage training process. *IEEE Access.* 2018;6:15844–69.
 27. Qinghe Z, et al. PAC-Bayesian framework based drop-path method for 2D discriminative convolutional network pruning. *Multidimens Syst Signal Process.* 2020;31(3):793–827.
 28. Qinghe Z, et al. Spectrum interference-based two-level data augmentation method in deep learning for automatic modulation classification. *Neural Comput Appl.* 2021;33(13):7723–45.
 29. Qinghe Z, et al. Layer-wise learning based stochastic gradient descent method for the optimization of deep convolutional neural network. *J Intell Fuzzy Syst.* 2019;37(4):5641–54.
 30. Qinghe Z, et al. A full stage data augmentation method in deep convolutional neural network for natural image classification. *Discret Dyn Nat Soc.* 2020;2020:1–11.
 31. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature.* 2015;521(7553):436.
 32. Fonseca Carlos M, Fleming PJ. Genetic algorithms for multiobjective optimization: formulation discussion and generalization. *Icga.* 1993;93:416–23.
 33. Fonseca CM, Fleming PJ. An overview of evolutionary algorithms in multiobjective optimization. *Evol Comput.* 1995;3(1):1–16.
 34. CoelloCoello CA. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Comput Intell Mag.* 2006;1(1):28–36.
 35. Srinivas N, Deb K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evol Comput.* 1994;2(3):221–48.
 36. Kalyanmoy D. *Multi-objective optimization using evolutionary algorithms*, vol. 16. Hoboken: John Wiley & Sons; 2001.
 37. Kalyanmoy D, et al. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *International Conference on Parallel problem solving from nature*. Heidelberg: Springer; 2000.
 38. Deb K, Jain H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Trans Evol Comput.* 2014;18(4):577–601.
 39. Figueiredo Mário AT, Anil K Jain. "Bayesian learning of sparse classifiers." *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE, 2001.
 40. Ligeiro R, Vilela Mendes R. Detecting and quantifying ambiguity: a neural network approach. *Soft Comput.* 2018;22(8):2695–703.
 41. Gecynalda S, da Gomes S, Teresa Ludermir B, Leyla Lima MMR. Comparison of new activation functions in neural network for forecasting financial time series. *Neural Comput Appl.* 2011;20(3):417–39.
 42. Karlik B, Vehbi Olgac A. Performance analysis of various activation functions in generalized MLP architectures of neural networks. *Int J Artif Intell Expert Syst.* 2011;1(4):111–22.
 43. Hyndman RJ, Akram M. *Time series data library*. 2010. Available from internet: <http://robjhyndman.com/TSDL>.
 44. Norgaard M, Ravn O, Poulsen NK. NNSYSID-toolbox for system identification with neural networks. *Math Comput Model Dyn Syst.* 2002;8(1):1–20.
 45. Xue Yu, Wang Y, Liang J. A self-adaptive gradient descent search algorithm for fully-connected neural networks. *Neurocomputing.* 2022;478:70–80.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.