Time Series Prediction Using Random Weights Fuzzy Neural Networks

Antonello Rosato and Massimo Panella
Department of Information Engineering, Electronics and Telecommunications (DIET)
University of Rome "La Sapienza"
Via Eudossiana 18, 00184 Rome, Italy
Email: {antonello.rosato, massimo.panella}@uniroma1.it

***Abstract*—In this paper, we introduce Random Weights Fuzzy Neural Networks as a suitable tool for solving prediction problems. The generalization capability of these randomized fuzzy neural networks is exploited in order to estimate accurately the sample be predicted from a multidimensional input. The latter is obtained by applying an embedding technique to the time series, which selects only the meaningful past samples to be used for prediction. We tested the proposed approach on real-world time series pertaining to the application context of power delivery. We proved the efficacy of the proposed approach by comparing its forecasting accuracy with respect to other prediction systems based on well-known data-driven regression models.**

I. Introduction

Fuzzy Neural Networks (FNNs) do provide a well suited solution for classification, regression and prediction problems, where supervised learning is used in real-world use cases relevant to physics, engineering, computer science, medicine, bioinformatics, econometrics, and even quantum computation [1]. This work is based on the Adaptive Neuro-Fuzzy Inference System (ANFIS) [2], which established itself as a pivotal tool for applications such as rule-based process control, pattern recognition, function approximation, etc.

In recent years, the ANFIS model has been reformulated in various fashions, particularly using a randomized version of the FNNs approach. In [3], this new model has been introduced as a Random Weights Fuzzy Neural Network (RWFNN), where the parameters of the membership functions are randomly selected instead of being estimated during the learning process.

It has been proven that this particular data-driven Fuzzy Inference System (FIS) has an advantage in terms of efficiency in certain applications [4]–[6]. The idea proposed in this work is based on the intuition of exploiting the RWFNN generalization capability to solve a prediction problem, by using a proper embedding technique to select the past samples of time series to be used for predicting the future ones [7].

Regarding the usage of randomized neural network models, a consistent investigation has been carried out both on classical Artificial Neural Networks [8]–[10] and on FNNs [11], [12]. In fact, it was proved that, if the parameters of the membership functions are kept fixed and only the consequent parameters are tuned, the resulting model is equivalent to a functional-link network, where the membership functions represent the functional expansion.

The novelty of the present work resides in solving the forecasting problem with the generalization capability of data-driven function approximation models. The optimal estimation of RWFNN parameters is obtained by solving a Regularized Least Squares (RLS) problem; such parameters represent the consequent part of randomized Sugeno first-order type fuzzy rules, which implement the reasoning system for estimating the expected outcome of the regression model. If we borrow from signal processing and chaotic modeling the embedding approach on the time series to be analyzed, it is possible to solve the prediction problem by using the RWFNN regressor [13].

Among supervised learning problems, we focus our attention on real-world time series forecasting: in this context, training a randomized architecture model with a data-driven approach can still be considered as an open challenge. In particular, regarding fluctuating and non-stationary time series, one of the most demanding tasks is reconstructing the state space evolution of energy-related physical systems, due to their intrinsic volatility [14]–[16]. The high diffusion of Renewable Energy Sources (RESs) consumers and production units causes the call for a sound forecasting algorithm to deploy in the smart grid scenario, giving us a good context in which to test the proposed RWFNN approach.

The present paper aims at assessing the efficacy of RWFNNs in solving time series forecasting problems. To this end, we consider energy-related time series on a real-world application context. The prediction performance is compared with respect to other feed-forward regression models adopted for prediction, such as linear Least Squares Estimator (LSE), Radial Basis Function (RBF), Mixture of Gaussian (MoG) neural networks, and Adaptive Neuro-Fuzzy Inference System (ANFIS). As the experimental tests illustrated in the following are encouraging, next developments may consider also recurrent and/or deep architectures for further comparisons as well as for extending the underlying novelty represented by randomized fuzzy rules applied to regression and prediction.

The rest of the paper is organized as follows. In Sect. II we introduce the structure of RWFNNs. In Sect. III we describe the embedding used in the training phase. In Sect. IV we present the experimental results obtained through numerical simulations. Finally, in Sect. V we summarize our proposal and we discuss future works.

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

II. Random Weights Fuzzy Neural Networks

In the following, we provide the fundamental concepts of the RWFNN model, describing its architecture as in [3]. We show that the problem of training a RWFNN can be formulated as an RLS problem.

An RWFNN is constituted by five feed-forward layers, each layer is in turn constituted by a set of nodes and each node is associated with a fuzzy rule. Each node performs a particular operation on the signals coming from the previous layer, and sends the result of the calculation to the nodes in the next layer. There are no connections between nodes in the same layer.

Let us consider the problem of estimating a scalar output $y \in \mathbb{R}$ from a d-dimensional input $\mathbf{x} = [x_1, \ldots, x_d]^{\mathrm{T}}$. Several alternatives are possible for the fuzzification of crisp inputs, the composition of input membership functions (MFs), and the way rule outputs are combined [17]. Let m be the predefined number of rules of the RWFNN network; usually, the structure of the fuzzy inference system can be summarized as follows.

- Layer 1. Every node i in this layer, $i = 1 \ldots m$, is associated with an input MF $\mu_{(i,j)}(x_j, \boldsymbol{\alpha})$ operating on the jth dimension of the input vector $\mathbf{x}$ for the ith rule. The parameters of the antecedents $\boldsymbol{\alpha}$ are chosen at the beginning of the learning process from a fixed probability distribution, which is independent on the training data.

- Layer 2. Every node in this second layer corresponds to an *if-then* rule of the FIS. If the adopted operator for the logical AND is the algebraic product, then the output of the ith node is:

$$w_i(\mathbf{x}) = \prod_{j=1}^{d} \mu_{(i,j)}(x_j),\ i = 1 \ldots m. \tag{1}$$

- Layer 3. Normalization:

$$\overline{w}_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_{h=1}^{m} w_h(\mathbf{x})},\ i = 1 \ldots m. \tag{2}$$

- Layer 4. Local output of the ith rule:

$$\tilde{f}_i(\mathbf{x}) = \overline{w}_i(\mathbf{x})(\boldsymbol{\beta}_i^{\mathrm{T}} \mathbf{x}^{+}),\ i = 1 \ldots m, \tag{3}$$

  where $\boldsymbol{\beta}_i$ is the $(d+1)$-dimensional vector given by $\boldsymbol{\beta}_i = [\beta_{(i,0)}, \ldots, \beta_{(i,d)}]^{\mathrm{T}}$ and $\mathbf{x}^{+} = [1, \mathbf{x}]^{\mathrm{T}}$.

- Layer 5. This layer is constituted by a single node that computes the overall output $\hat{y}$ as the sum of all the normalized firing strengths:

$$\hat{y} = \sum_{i=1}^{m} \tilde{f}_i\,. \tag{4}$$

Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be the set of data available for the training phase, and denote with $x_{(r,j)}$ the jth component of the rth input vector. We define the hidden matrix $\mathbf{H} = \left[\mathbf{H}^1, \ldots, \mathbf{H}^m\right]$, where:

$$\mathbf{H}^i = \begin{bmatrix} \overline{w}_i & \overline{w}_i x_{(1,1)} & \cdots & \overline{w}_i x_{(1,d)} \\ \overline{w}_i & \overline{w}_i x_{(2,1)} & \cdots & \overline{w}_i x_{(2,d)} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{w}_i & \overline{w}_i x_{(n,1)} & \cdots & \overline{w}_i x_{(n,d)} \end{bmatrix},\ i = 1 \ldots m. \tag{5}$$

and the output vector $\mathbf{y} = [y_1, \ldots, y_n]^{\mathrm{T}}$. Rearranging the parameters $\boldsymbol{\beta}$ in the form:

$$\boldsymbol{\beta} = [\boldsymbol{\beta}_{(1,0)} \ldots \boldsymbol{\beta}_{(1,d)} \boldsymbol{\beta}_{(2,0)} \ldots \boldsymbol{\beta}_{(2,d)} \ldots \boldsymbol{\beta}_{(m,0)} \ldots \boldsymbol{\beta}_{(m,d)}]^{\mathrm{T}}, \tag{6}$$

the optimization problem for training a RWFNN can be reformulated as a Least Squares (LS) problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{2} \left\| \mathbf{y} - \mathbf{H}\boldsymbol{\beta} \right\|_2^2, \tag{7}$$

where $\|\cdot\|_2$ is the l^2 norm and $p = m(d+1)$. The formulation in (7) may suffer of numerical instability due to the possible small values of $\overline{w}_i$, additionally, the optimal solution is undetermined if $n < p$. For this reason, we modify the optimization problem by setting it in the form of a RLS problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{2} \left\| \mathbf{y} - \mathbf{H}\boldsymbol{\beta} \right\|_2^2 + \frac{\lambda}{2} \left\| \boldsymbol{\beta} \right\|_2^2 \tag{8}$$

where $\lambda > 0$ is the *regularization factor*. The new problem (8) is strictly convex, and then it has a unique solution that can be obtained in closed form as:

$$\boldsymbol{\beta}^{*} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda \mathbf{I}\right)^{-1} \mathbf{H}^{\mathrm{T}} \mathbf{y}\,. \tag{9}$$

III. Embedding of Time Series for Solving Prediction Problems

Since forecasting is of actual importance, the pertinent literature proposes plenty of methods to solve the prediction problem with neural and fuzzy neural networks [18]. Indeed, by properly transforming the said problem into a function approximation one, such models are well-suited for the case.

In the previous section, we reached a unique solution for estimating a scalar output from a d-dimensional input $\mathbf{x}$, solving a RLS problem. If we now consider a time series $S(t)$, its future samples can be predicted by means of a generalized function approximation $y = f(\mathbf{x})$, $f : \mathbb{R}^d \to \mathbb{R}$.

For instance, resuming the d-dimensional input $\mathbf{x}$ introduced earlier, each input vector can be made of d subsequent samples of $S(t)$ and the scalar output y, which is the sample to be predicted, can be obtained by a linear model $f(\cdot)$:

$$\mathbf{x} = \left[S(t)\ S(t-1) \ldots\ S(t-d+1)\right], \tag{10}$$

$$y = S(t+m)\,, \tag{11}$$

$$f_{\mathrm{lin}}(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j x_j. \tag{12}$$

Then, we obtain:

$$\widetilde{S}(t+m) = \sum_{j=1}^{d} \lambda_j S(t-j+1) \ , \tag{13}$$

where $\widetilde{S}(t+m)$ is the prediction of the true value $S(t+m)$ at the time distance m. Considering the statistical properties of $S(t)$, as the autocorrelation function, it is possible to determinate the parameters λ_j, $j = 1 \dots N$, of the function $f_{\text{lin}}(\cdot)$.

In our case, input vectors $\mathbf{x}$ are obtained through the so called 'embedding technique', which makes use of previous samples of $S(t)$ to build the vectors themselves. Thus, the dimension d of the input vector corresponds to the embedded samples of the original time series at time t. Two are the parameters to be set in this regard, the embedding dimension d and the time lag θ, resulting in the following embedding:

$$\mathbf{x} = \left[S(t)\ S(t-\theta)\ S(t-2\theta) \dots\ S(t-(d-1)\theta)\right] . \tag{14}$$

These two parameters are estimated by using the False Nearest Neighbors (FNN) algorithm for the embedding dimension and the Average Mutual Information (AMI) criterion for the time lag [19]. In the following of the paper, both d and θ will be chosen by such algorithms via the VRA software available at https://visual-recurrence-analysis.software.informer.com/download/.

The performances of a predictor that relies on a linear approximation model, as the one in (10), are very poor when it is applied to data sequences in real environments. They often have noisy and chaotic components that force to wisely choose the embedding parameters, as well as the function approximation model, by using appropriate procedures [20].

In this context, the underlying, unknown system is observed through $S(t)$ and its state-space evolution is obtained by the trajectories of embedded vectors. The estimated sample $\widetilde{S}(t+m)$ predicted at a distance m will be:

$$\widetilde{S}(t+m) = f\left(\mathbf{x}\right) \ , \tag{15}$$

where $f(\cdot)$ is the regression model to be determined. Thus, such a function will approximate the link from the reconstructed state $\mathbf{x}$ to the corresponding output y [19]. It is important to note that $f(\cdot)$ must be non-linear since the considered system has a complex behavior.

As said, because of the intrinsic nonlinearity and non-stationarity of a chaotic system, the regression model should be a nonlinear function, which must be determined using data-driven techniques that are typical of the neural network approach. In fact, in this work we will use the non-linear function stemming from the RWFNN reasoning in the previous section. Along, other predictors will be employed, where the nonlinear function is based on other models, namely:

- LSE: the relationship between the value to be predicted and the current ones is modeled as a linear function, an LS algorithm is used to estimate the parameters [21];
- RBF: a neural network that builds up a function approximation model with a usually multiquadratic radial basis functions [22];
- MoG: a neural network model in which a density mixture of Gaussian components are used in the joint input-output space; the mixture parameters are estimated via the Expectation-Maximization [23];
- ANFIS: a data driven fuzzy inference system based on a set of Sugeno first-order type fuzzy rules trained by back-propagation [2].

IV. Experimental Tests

In order to asses the performance of the proposed RWFNN-based approach, we consider real-world data in the renewable energy context. The neural architectures with which we compare the RWFNN model were introduced in the previous section, i.e. RBF, MoG and ANFIS. The baseline for the tests is the linear predictor reported in (10), whose parameters are estimated through a common LSE.

A. Data Description

The proposed approach focuses on a particular application, regarding the power plant located in Sant'Eusanio del Sangro, in the Abruzzo region of Italy. The data is relative to a single PV plant that belongs to a broader system with other interconnected plants operated by the same agent. The complete numerical dataset is available at https://github.com/max-panella/panella/raw/master/FUZZ2020_Dataset.zip.

In the considered prediction problem, the adopted time series is composed by the output voltage of the plant sub-sampled at one sample per hour. The time series is linearly normalized in the range $[-0.5, +0.5]$, so as to satisfy the numerical requirements of the data driven prediction models.

The adopted training set is a single month (mostly 31 days) of the time series, then resulting in 744 samples. During learning, the considered regression model is trained by using an input-output dataset that is built, through embedding, on the first 500 samples of the adopted time series. The remaining 244 samples are used to cross-validate the complexity of the adopted regression model (i.e., order of linear predictor, Gaussian kernels in RBF and MoG, number of fuzzy rules in ANFIS and RWFNN).

The final model found by the training algorithm is tested on the next operation day (i.e., 24 samples) after the end of the month. In this regard, we are facing a very challenging prediction problem, as all of the 24 values of the new day must be predicted at once, for regulating purposes of the energy distribution system. Thus, in the following we will consider an unusual prediction distance, which is set to $m = 24$, in a 'multi-step-ahead' prediction problem.

The performances are measured by the Normalized Mean Squared Error (NMSE), which is defined as:

$$\text{NMSE} = \frac{\sum_n \left(S(t) - \widetilde{S}(t)\right)^2}{\sum_n \left(S(t) - \bar{S}\right)^2} \ , \tag{16}$$

where $\widetilde{S}(t)$ is the predicted value of $S(t)$ and $\bar{S}$ is the average on the considered values of $S(t)$.

We tested the proposed approach by using four different datasets, all taken from the described time series relative to the year 2016. These sets exhibit some different behaviors, as for the maximum values reached in a day and for the volatility of the time series. The training set is a whole month of a different season (winter, spring, summer, and autumn), as described in the following and shown from Fig. 1 to Fig. 4:

- Winter: from January 1 to January 31, 2016, standard embedding parameters $\theta = 13$, $d = 25$;
- Spring: from March 31 to April 30, 2016, standard embedding parameters $\theta = 10$, $d = 20$;
- Summer: from July 1 to July 31, 2016, standard embedding parameters $\theta = 10$, $d = 14$;
- Autumn: from October 1 to October 31, 2016, standard embedding parameters $\theta = 10$, $d = 13$.

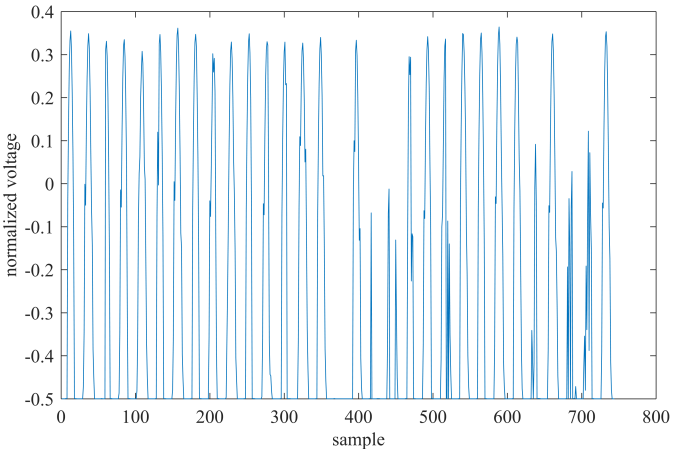

Fig. 3. Summer training set.

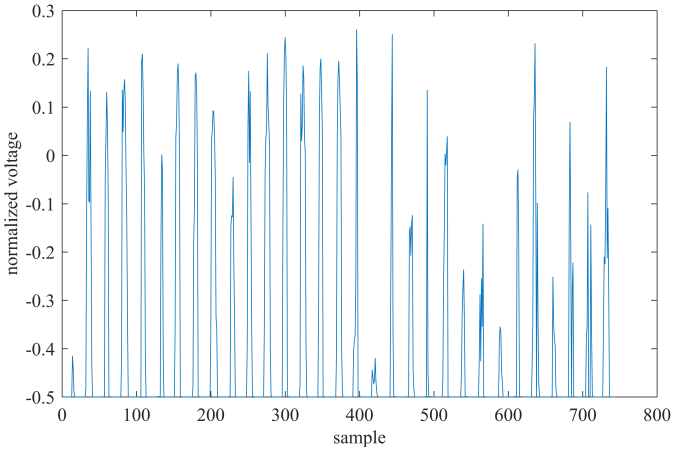

Fig. 1. Winter training set.

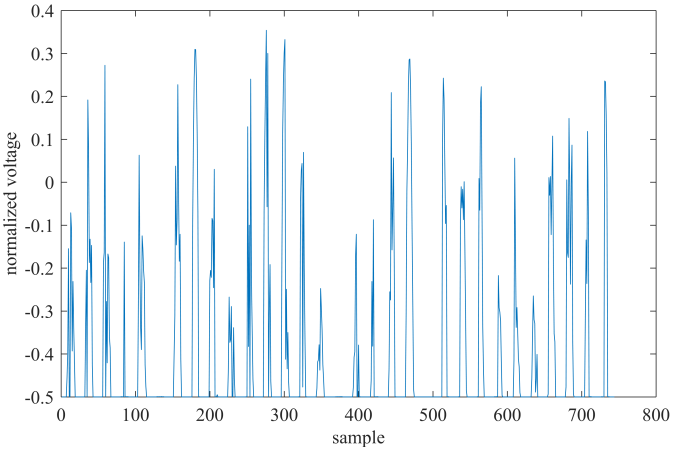

Fig. 4. Autumn training set.

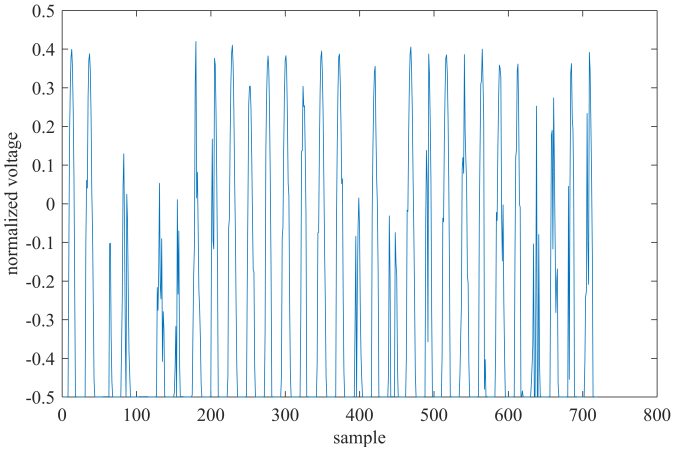

Fig. 2. Spring training set.

B. Numerical Results

The numerical results on the test set for all of the considered datasets are reported in Table I, while the graphical behavior is shown from Fig. 5 to Fig. 8 by using the proposed RWFNN-based prediction model. All of the experiments had been performed using Matlab™ R2019a on a machine equipped with Intel® Core™ i7-3770K 64-bit CPU at 3.50 GHz, 32 GB RAM, and NVIDA GTX 680 GPU.

TABLE I
NUMERICAL RESULTS (NMSE) FOR DIFFERENT TIME SERIES AND PREDICTION MODELS

Prediction model	Winter	Spring	Summer	Autumn
LSE	0.825	0.066	0.057	0.234
RBF	0.616	0.062	0.062	0.132
MoG	0.753	0.063	0.045	0.155
ANFIS	0.677	0.065	0.044	0.159
RWFNN	0.591	0.059	0.040	0.107

The obtained results are coherent in showing the best prediction accuracy of the proposed RWFNN model, although there are many differences among the various seasons; winter

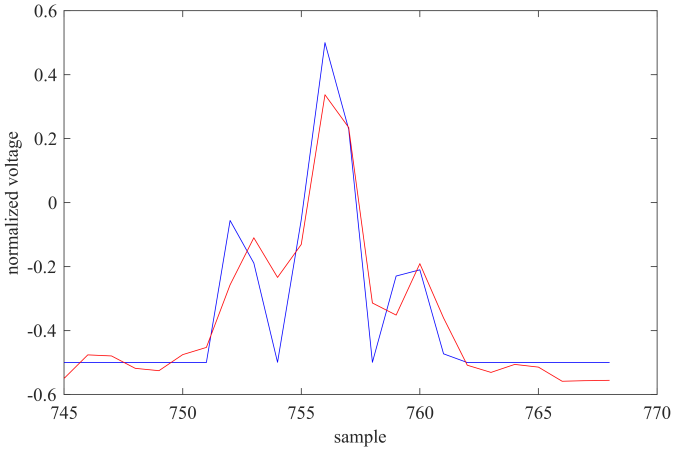

Fig. 5. Winter test set (blue) and predicted time series (red) using RWFFN.

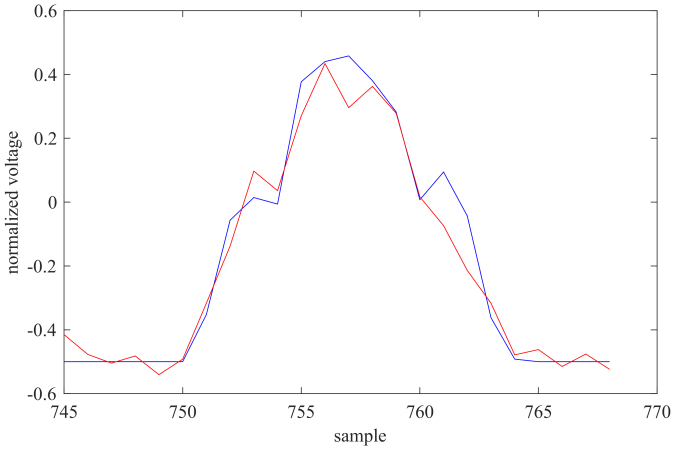

Fig. 6. Spring test set (blue) and predicted time series (red) using RWFFN.

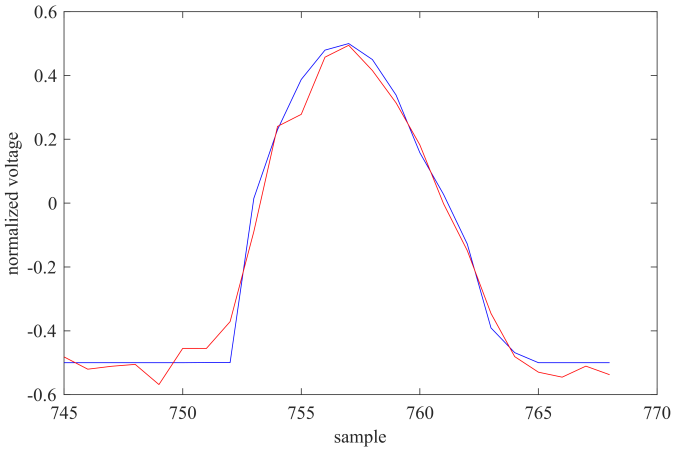

Fig. 7. Summer test set (blue) and predicted time series (red) using RWFFN.

seems to be the hardest one to be predicted. For every season, there are improvements in terms of NMSE reduction with respect to other models that, considering the best one among them, range from 4% up 18%.

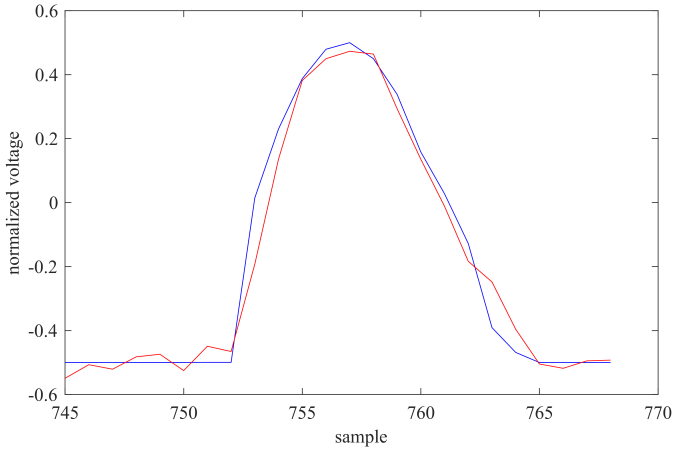

Fig. 8. Autumn test set (blue) and predicted time series (red) using RWFFN.

Overall, the usefulness of the RWFNN approach is reinforced by the results, in which even a slight NMSE decrease must be considered a significant amelioration, since the physical quantities in play are quite large, although normalized.

V. Conclusions

In this paper, we have proposed a new method of solving prediction tasks using the RWFNN paradigm. By formulating forecasting as a function approximation problem, we are able to make use of the embedding technique in order to appropriately select the meaningful samples of the time series to be predicted. A straightforward RLS problem is defined for training the network parameters of the randomized FIS.

The performance of the proposed model has been assessed through a comprehensive benchmarking on a real-world application considering well-known regression models. The obtained numerical results prove the feasibility and the efficacy of the proposed approach.

In future works, a distributed implementation of the forecasting RWFNN paradigm might be considered, where local predictors can have the ability to exchange information with other prediction agents in the network and to reach consensus among some model parameters, so as to achieve a better prediction accuracy.

References

[1] M. Panella and G. Martinelli, "Neurofuzzy networks with nonlinear quantum learning," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 3, pp. 698 –710, 2009.

[2] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[3] R. Fierimonte, M. Barbato, A. Rosato, and M. Panella, "Distributed learning of random weights fuzzy neural networks," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2016, pp. 2287–2294.

[4] A. Cabalar, A. Cevik, and C. Gokceoglu, "Some applications of adaptive neuro-fuzzy inference system (anfis) in geotechnical engineering," *Computers and Geotechnics*, vol. 40, p. 14–33, 03 2012.

[5] M. Salleh and K. Hussain, "A review of training methods of anfis for applications in business and economics," *International Journal of u- and e- Service, Science and Technology*, vol. 9, pp. 165–172, 07 2016.

[6] B. Savkovic, P. Kovac, B. a. Z. Dudic, D. Rodic, M. Taric, and M. Gregus, "Application of an adaptive "neuro-fuzzy" inference system in modeling cutting temperature during hard turning," *Applied Sciences*, vol. 9, p. 3739, 09 2019.

[7] M. Casdagli, "Nonlinear prediction of chaotic time series," *Physica D: Nonlinear Phenomena*, vol. 35, no. 3, pp. 335 – 356, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167278989900742

[8] W. F. Schmidt, M. A. Kraaijveld, and R. P. W. Duin, "Feedforward neural networks with random weights," in *11th IAPR International Conference on Pattern Recognition, 1992*. IEEE, 1992, pp. 1–4.

[9] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[10] L. Wang, B. Liu, and C. Wan, "A novel RBF neural network with fast training and accurate generalization," in *Computational and Information Science*. Springer, 2004, pp. 166–171.

[11] ——, "On the universal approximation theorem of fuzzy neural networks with random membership function parameters," in *Advances in Neural Networks–ISNN 2005*. Springer, 2005, pp. 45–50.

[12] Y.-L. He, X.-Z. Wang, and J. Z. Huang, "Fuzzy nonlinear regression analysis using a random weight network," *Information Sciences*, 2016, in press.

[13] M. Panella, "A hierarchical procedure for the synthesis of ANFIS networks," *Advances in Fuzzy Systems*, vol. 2012, pp. 1–12, 2012.

[14] F. Chen, D. Liu, and X. Xiong, "Research on stochastic optimal operation strategy of active distribution network considering intermittent energy," *Energies*, vol. 10, no. 4, pp. 1996–1073, 2017.

[15] X. Han, S. Liao, X. Ai, W. Yao, and J. Wen, "Determining the minimal power capacity of energy storage to accommodate renewable generation," *Energies*, vol. 10, no. 4, pp. 1996–1073, 2017.

[16] A. Rosato, R. Altilio, R. Araneo, and M. Panella, "Prediction in photovoltaic power by neural networks," *Energies*, vol. 10, no. 7, 2017.

[17] J.-S. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: a Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ, USA: Prentice Hall, 1997.

[18] F. Masulli and L. Studer, "Time series forecasting and neural networks," in *Proc. of IJCNN'99, Washington D.C., USA*, 1999.

[19] H. Abarbanel, *Analysis of Observed Chaotic Data.* Springer, New York, 1996.

[20] M. Panella, A. Rizzi, F. M. F. Mascioli, and G. Martinelli, "ANFIS synthesis by hyperplane clustering," in *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, vol. 1, July 2001, pp. 340–345 vol.1.

[21] H. W. Bode and C. E. Shannon, "A simplified derivation of linear least square smoothing and prediction theory," *Proceedings of the IRE*, vol. 38, no. 4, pp. 417–425, 1950.

[22] A. Bors and I. Pitas, "Median radial basis function neural network," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1351–1364, 1996.

[23] M. Panella, A. Rizzi, and G. Martinelli, "Refining accuracy of environmental data prediction by MoG neural networks," *Neurocomputing*, vol. 55, pp. 521–549, 2003.