Evolving Reservoir Weights in the Frequency Domain

Sebastián Basterrech
Faculty of Electrical Engineering and Computer Science,
VŠB-Technical University of Ostrava
Ostrava, Czech Republic
Sebastian.Basterrech@vsb.cz

Gerardo Rubino
Inria Rennes – Bretagne Atlantique
Rennes, France
Gerardo.Rubino@inria.fr

ABSTRACT

Reservoir Computing models are a class of recurrent neural networks that have enjoyed recent attention, in particular, their main family, Echo State Networks (ESNs). These models have a large number of hidden-hidden weights (in the so-called reservoir) forming a recurrent topology. The reservoir is randomly connected with fixed weights during learning: only readout parameters (from reservoir to output neurons) are trained; the reservoir weights are frozen after initialized. Since the reservoir structure is fixed during learning, only its initialization process has an impact on the model's performance.

In this work, we introduce an evolutionary method for adjusting the reservoir non-null weights. Moreover, the evolutionary process runs on the frequency space corresponding to a Fourier transformation of the weights. We combine an evolutionary search in the Fourier space with supervised learning for the readout weights. The resulting algorithm, called EvoESN (Evolutionary ESN), obtains promising results modeling two well-known problems of the chaotic time-series area.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Bio-inspired approaches**; **Genetic algorithms**;

KEYWORDS

Neuroevolution, Recurrent Neural Networks, Echo State Network, Genetic Algorithms, Reservoir Computing

ACM Reference Format:
Sebastián Basterrech and Gerardo Rubino. 2021. Evolving Reservoir Weights in the Frequency Domain. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459457

1 INTRODUCTION

A Recurrent Neural Network (RNN) is a dynamical system with the ability of conveying information across time. The recurrences ensure that historical information from the input data can be stored in internal hidden states. In spite of the good properties and encouraging experimental success, RNNs are complex and difficult to properly train, especially when the network is large and there are long-term data dependencies [5].

During the last 20 years, a type of RNNs named Reservoir Computing (RC) has captured considerable attention, due to its outstanding results for modeling chaotic systems and solving real-world problems [5]. A pioneer RC model is the Echo State Network (ESN) [5]. The model uses a recurrent hidden-to-hidden structure (the *reservoir*) to memorize patterns of the input history. The reservoir is driven by the input signal, and projects the input into a high-dimensional space. The reservoir weights are realizations of i.i.d. random variables. To guarantee specific stability conditions, the random initialised weights are scaled in order to keep control over the spectral radius of W^{r} matrix [1, 3]. The training algorithm adjusts only the rest of the model parameters, which composes a fast adapting readout layer. The dynamics of the model is described in terms of a discrete-time system with an input signal $u(t) \in \mathbb{R}^K$ and a hidden state $x(t) \in \mathbb{R}^N$. The recurrent state $x(t)$ and the model's output $y(t)$ are computed according to the following expressions [3]: $x(t) = f\big(W^{\mathrm{in}}u(t) + W^{\mathrm{r}}x(t-1) + W^{\mathrm{fb}}y(t) + \varepsilon(t)\big)$, and $y(t) = g(W^{\mathrm{out}}[u(t); x(t)])$, where t is the discrete time index, W^{in} is a matrix with input-to-reservoir weights, W^{r} has the reservoir internal weights, W^{out} is the reservoir-to-output weight matrix, W^{fb} is the matrix with feedback connections and $\varepsilon(t)$ is a noise vector. The $[\cdot;\cdot]$ operation denotes vector concatenation. Functions $f(\cdot)$ and $g(\cdot)$ are predefined coordinate-wise functions. Matrix W^{fb} and vector $\varepsilon(t)$ are optional (they are set to zero in the case of standard ESN).

Since the large recurrent structure is fixed during the training, the RC family has also received some critics [6]. In this work, we present an evolutionary computational model for creating efficient reservoirs called EVOlutionary Echo State Network (EvoESN). We combine ideas regarding indirect encoding of the weight connections [4], EVOLINO [6], and the efficiency of ESNs [5]. In the proposed model, the reservoir has a fixed pattern of connectivity (as in standard ESNs), and we apply the evolutionary process to find good weight connections. The evaluation of the proposed computational model was made over two well-known benchmark problems: Mackey Glass system and Lorenz system [1, 3, 6].

2 EVOLUTIONARY ECHO STATE NETWORK

Let us denote by M the number of non-null weights in W^{r}. Since reservoirs are designed to be sparse for efficiency reasons, then $M \ll N^2$ [5]. We arbitrary design the pattern of connectivity of the reservoir, i.e. we choose M positions in W^{r} with non-zeros values. Let's denote by p the positions of those non-null coefficients, a vector having size M. By $p_k = (i, j)$ we mean that the kth position in p refers to the weight in row i and column j of W^{r}. We also use a vector v having dimension M, containing those weights, that

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459457

is, $v_k = W^r_{p_k} = W^r_{i,j}$. During the procedures described below, p (the positions) remains fixed and v (the values) evolves. In order to obtain a good reservoir, we move vector v to the frequency domain using the DCT transform. Then, we change it running a Genetic Algorithm (GA) and we go back to weights through the inverse DCT. We evaluate the introduced changes by training the ESN with the just obtained reservoir. This process is repeated until some convergence condition is satisfied.

Let us denote $\alpha = (\alpha_1, \ldots, \alpha_C)$ the vector with the first C DCT coefficients, $C \leq M$, that is randomly initialized. Denote now by $\phi(\cdot)$ the inverse DCT transformation. In case $C < M$, we extend α with zeros (*padding*) until dimension M is reached, in order to make $\phi(\alpha)$ a vector with dimension M. Once we have computed α (the *chromosome*) and $v = \phi(\alpha)$ (the *phenotype*), the ESN is trained using a classic technique [5]. The whole procedure is repeated until some appropriate convergence condition is satisfied. The algorithm can then be summarized as follows: (i) Choose M, C and initialize α. (ii) Extend α with zeros until dimension M is reached. (iii) Compute $v = \phi(\alpha)$. (iv) Store the values in v to their positions p in the reservoir. (v) Use some standard training schema for adjusting the ESN readout weights. (vi) Compute a fitness function using the trained ESN and testing data. (vii) If the convergence condition is not satisfied, apply the evolution operations (run the GA), compute a new vector α and repeat from (ii).

3 EXPERIMENTAL RESULTS

We evaluate the performance of the proposed model on well-known benchmark problems. Due to constrains in space, we report only some results obtained over Mackey-Glass and Lorenz systems. In both benchmark problems, we adopted the experimental setting well-described in [3], i.e. we applied the same network architectures and learning configurations. The inverse DCT was made with the orthogonal norm using the `fftpack Python` package. The evolutionary search was conducted employing GAs, and it was implemented using the DEAP library [2]. Table 1 shows the obtained results for Lorenz task. We show the NRMSE_{84} error which is a standard measure for MGS and also used in Lorenz task [3, 6]. The error has same order of magnitude than that obtained by Jaeger *et al.* using a feedback connection and a refined learning method [3]. We present the results for EvoESN when the number C of coefficients in the frequency representation has values 50, 100 and 150. In addition, we present the achieved error when the model predicts a 600 time horizon (600H). Figure 1 illustrates the power of the proposed approach. The red curves (errors obtained by EvoESN with 500 coefficients) correspond to the evolution of the errors against the generations, for different independent experiments on the MGS problem. In horizontal lines we show Evolino [6] and ESN with Feedback connections and refined learning method (ESN-FB II) [3]. After a relatively few number of generations, it is possible to reach the best registered performance for solving the MG task. We expect to do further studies using statistical analysis and a full evaluation of the GA parameters, in order to explore the differences between ESN with feedback connections and EvoESN on the MSG problem.

4 CONCLUSIONS

This work introduces new insights in the Reservoir Computing (RC) paradigm. We present a new computational model entitled EvoESN. We added to the standard RC approach an evolutionary procedure for finding a *good* set of reservoir weights. The proposed method projects the reservoir weights into the frequency domain using the DCT. Then, the reservoir weights are found in the Fourier space using an evolutionary algorithm. Promising results have been achieved modeling two popular chaotic systems.

Table 1: Results for Lorenz system prediction.

Model	Coeff.	Error	Metric
EvoESN	50	-3.4666	log_{10} NRMSE_{84}
EvoESN	100	-3.6014	log_{10} NRMSE_{84}
EvoESN	150	-4.1379	log_{10} NRMSE_{84}
EvoESN	50	-0.2704	log_{10} absolute error (600H)
EvoESN	150	-0.1407	log_{10} absolute error (600H)

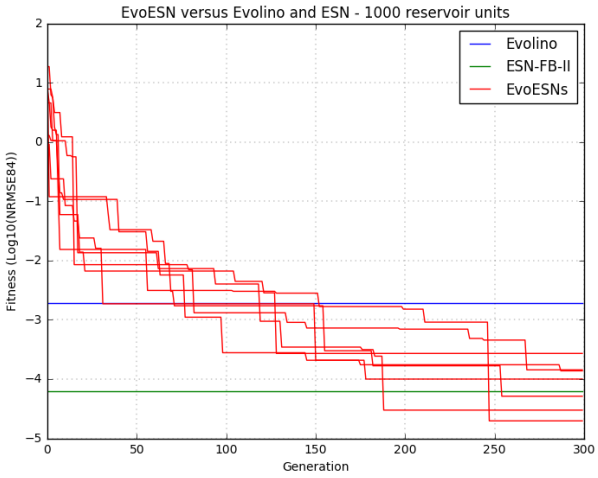

Figure 1: NRMSE_{84} (log 10 scale) of the MGS prediction.

ACKNOWLEDGEMENTS

This work was supported by the GACR-Czech Science Foundation project no. 21-33574K "Lifelong Machine Learning on Data Streams".

REFERENCES

[1] Sebastián Basterrech. 2017. Empirical analysis of the necessary and sufficient conditions of the echo state property. In *International Joint Conference on Neural Networks, IJCNN'17*. IEEE Press, Anchorage, AK, USA, 888–896. https://doi.org/10.1109/IJCNN.2017.7965946

[2] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research* 13 (jul 2012), 2171–2175.

[3] Herbert Jaeger and Harald Haas. 2004. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science* 304 (April 2004), 78–80. Issue 5667. https://doi.org/10.1126/science.1091277

[4] Jan Koutnik, Faustino Gomez, and Jürgen Schmidhuber. 2010. Evolving Neural Networks in Compressed Weight Space. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation* (Portland, Oregon, USA) *(GECCO '10)*. Association for Computing Machinery, New York, NY, USA, 619–626. https://doi.org/10.1145/1830483.1830596

[5] Mantas Lukoševičius and Hebert Jaeger. 2009. Reservoir Computing Approaches to Recurrent Neural Network Training. *Computer Science Review* 3 (2009), 127–149. Issue 3. https://doi.org/10.1016/j.cosrev2009.03.005

[6] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez. 2007. Training Recurrent Networks by Evolino. *Neural Networks* 19 (2007), 757–779.