Parameterized Pattern Generation via Regression in the Model Space of Echo State Networks

Witali Aswolinskiy and Jochen Steil

Research Institute for Cognition and Robotics - CoR-Lab
Universitätsstraße 25, 33615 Bielefeld, Germany
waswolinskiy@cor-lab.uni-bielefeld.de
https://www.cor-lab.de/

Abstract. Recurrent neural networks capable of sequential pattern generation could facilitate new types of applications like music generation. Here, we explore the capability of echo state networks for parameterized pattern generation and present a new approach utilizing regression in the model space. The goal of the learning is a system that can generate patterns for previously unseen parameterizations. Contrary to other approaches, where a single network is trained to generate all pattern parameterizations, we learn to generate a different network for each pattern parameterization. We evaluate the classical and our modular approach on several synthetic, periodic datasets. We show that regression in the model space of echo state networks can generate parameterized patterns more precisely than a single echo state network.

Keywords: time series generation, pattern generation, echo state network, reservoir computing, model space

1 Introduction

Sequential pattern generation has potentially many applications in signal processing, e.g. filling gaps in time series, computational creativity, e.g music generation and time series modelling. Compared to the main areas of machine learning such as classification, regression and clustering, few advances have been made in pattern generation. The reasons for this include the lack of datasets and benchmarks and the difficulty of training recurrent neural networks, especially to generate stable output. Recently, several variants of Echo State Networks (ESNs, [6]) were applied to a range of pattern generation tasks including frequency modulation [7, 9, 10] and learning human motion [13, 8].

Here, we focus on parameterized pattern generation: Given a set of pattern sequences shaped by control parameters, the goal is to learn to generate patterns for new control parameter values. In a sine wave generator, for example, the control parameter would be the frequency and the goal the generation of a sine wave with a frequency not used during training. This is a more difficult task than to learn to reproduce patterns, since it involves the learning of the underlying dynamical system producing the patterns and requires the learner to generalize in the space of the control parameters. The solution for this type of task with ESNs, as applied for similar tasks in [7, 9, 10, 13], is to train a single network, which receives the control parameter as input. We propose a different solution, where for each pattern a new network is generated based on the value of the control parameter. This approach is inspired by learning in the model space [3], which was successfully applied to time series classification [4, 2] and to the similar problem of modelling parameterized processes [1].

Fig. 1 visualizes the core architectures of both approaches. In contrast to the classical approach, in our modular approach for each control parameter value the *generalist* creates a *specialist* generator, which is only responsible for generating the corresponding pattern. The *generalist* is responsible for generalizing in the control parameter space, so that the *specialists* can concentrate on generating their specific patterns. Thus, the *generalist* maps the control parameter space to the space of *specialist* models - we refer therefore to our approach in accord with [1] as model space regression (MSR).

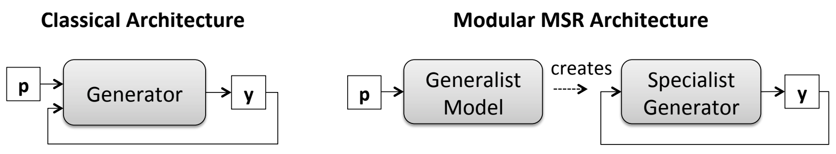

Fig. 1: Pattern generation with a classic (left) and a modular architecture (right). The control parameter $\boldsymbol{p}$ shapes the produced output $\boldsymbol{y}$.

The remainder of this paper is structured as follows. In the next section, we describe the basic, classical ESN pattern generator. In Section 3 we present our modular approach. In Section 4, we compare both approaches on several synthetic datasets. The paper closes with a discussion and some concluding remarks.

2 Echo State Network pattern generator with the control parameters as inputs

An ESN consists of two parts: A reservoir of recurrently connected neurons and a linear readout. The reservoir provides a non-linear fading memory of the inputs. For pattern generation, the network operates with output feedback (cf. Fig. 2). The reservoir states $\boldsymbol{x} \in \mathbb{R}^N$ and readouts $\boldsymbol{y} \in \mathbb{R}^O$ are updated according to:

$$\boldsymbol{x}(k) = (1-\lambda)\boldsymbol{x}(k-1) + \lambda f(\boldsymbol{W}^{rec}\boldsymbol{x}(k-1) + \boldsymbol{W}^{in}\boldsymbol{u}(k) + \boldsymbol{W}^{back}\boldsymbol{d}(k)) \qquad (1)$$

$$\boldsymbol{y}(k) = \hat{\boldsymbol{d}}(k+1) = \boldsymbol{W}^{out}\boldsymbol{x}(k), \qquad (2)$$

where $\boldsymbol{u}(k) \in \mathbb{R}^U$ and $\boldsymbol{d}(k) \in \mathbb{R}^O$ with $k = 1, \ldots, K$ are the input and output sequences, respectively; λ is the leak rate, f the activation function, e.g. *tanh*, $\mathbf{W}^{rec}$ the recurrent weight matrix, $\mathbf{W}^{in}$ the input weight matrix, $\mathbf{W}^{out}$ the matrix from the reservoir to the output and $\mathbf{W}^{back}$ the matrix from the output to the reservoir. $\mathbf{W}^{in}$, $\mathbf{W}^{rec}$ and $\mathbf{W}^{back}$ are initialized randomly, scaled and remain fixed. $\boldsymbol{W}^{rec}$ is typically scaled to achieve a spectral radius smaller than one.

The readout is trained to predict the next pattern step, using the training sequence, which is known as teacher forcing [7]:

$$E(\boldsymbol{W}^{out}) = \frac{1}{K-1}\sum_{k=2}^{K}(\boldsymbol{d}(k) - \boldsymbol{W}_i^{out}\boldsymbol{x}(k-1))^2 + \alpha \left\|\boldsymbol{W}^{out}\right\|^2, \tag{3}$$

$$\boldsymbol{W}^{out} = (\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{D}, \tag{4}$$

where $\boldsymbol{D}$ are the row-wise collected pattern signal values and $\boldsymbol{X}$ the corresponding reservoir activations. α is the regularization strength.

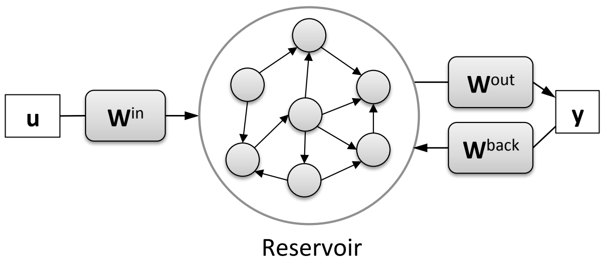

Fig. 2: Echo State Network with input $\boldsymbol{u}$ and output $\boldsymbol{y}$, which is fed back to the reservoir.

During testing (pattern generation) the output $y(k) = \hat{d}(k+1)$ serves as an estimation of the next pattern step and is fed back into the reservoir. In parameterized pattern generation, the input $\boldsymbol{u}$ corresponds to the control parameter, e.g. the sine frequency for a sine wave generator.

3 Parameterized pattern generation via regression in the model space

The training of the MSR architecture is depicted in Fig. 3. It consists of two steps: First, for each pattern, an ESN is trained using teacher forcing. The ESNs are trained independently, but share the same reservoir parameters $\boldsymbol{W}^{rec}$ and $\boldsymbol{W}^{back}$ in order to create a coherent model space. Second, an Extreme Learning Machine (ELM, [5]) is trained as *generalist* to map the control parameters to

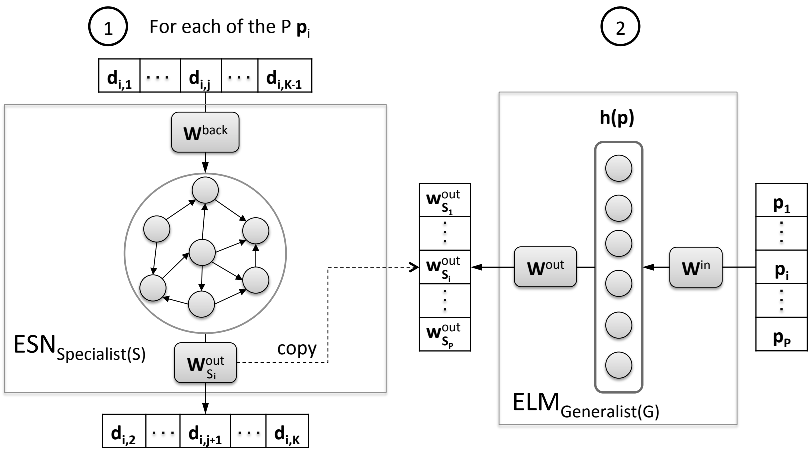
1
For each of the P $\mathbf{p}_i$
$\mathbf{d}_{i,1}$ ⋯ $\mathbf{d}_{i,j}$ ⋯ $\mathbf{d}_{i,K-1}$
$\mathbf{W}^{back}$
$\text{ESN}_{\text{Specialist(S)}}$
$\mathbf{W}^{out}_{S_i}$
copy
$\mathbf{d}_{i,2}$ ⋯ $\mathbf{d}_{i,j+1}$ ⋯ $\mathbf{d}_{i,K}$
2
h(p)
$\mathbf{W}^{out}_{S_1}$
$\mathbf{W}^{out}_{S_i}$
$\mathbf{W}^{out}_{S_P}$
$\mathbf{W}^{out}$
$\mathbf{W}^{in}$
$\mathbf{p}_1$
$\mathbf{p}_i$
$\mathbf{p}_P$
$\text{ELM}_{\text{Generalist(G)}}$

Fig. 3: Training with Model Space Regression (MSR). In the first step, for each of the P control parameters $\boldsymbol{p}_i$ a *specialist* generator is trained using the corresponding pattern outputs $\boldsymbol{d}_{i,j}$, where $j = 1, \ldots, K_i$ is the pattern sample index. In the second step, the *generalist* ELM is trained to map the the control parameters $\boldsymbol{p}_i$ to the ESN readout weights $\boldsymbol{W}^{out}_{S_i}$.

the trained readout weights of the ESNs. The ELM is a two-layer feed-forward network with a random hidden layer and a linear readout layer trained by ridge regression. We chose here the ELM for its simplicity and fast training time - other non-linear regressors like multilayer perceptrons could be used too.

During testing, the *generator* creates from a control parameter value the ESN readout weights. A new ESN is created with the reservoir shared by all ESNs during training and the created readout weights. Then, the created ESN is run autonomously in a feedback loop.

4 Results

We tested the classical ESN pattern generator and our MSR approach on several synthetic datasets. As testing scheme we used leave-one-out-cross-validation (LOOCV), where in P folds, $P-1$ patterns were used for training and the remaining patterns for testing. That is, the trained system, given the control parameter value, had to produce the corresponding pattern from a zero-state.

ESNs have several important hyper-parameters, e.g. input scaling and ridge factor, which have severe effect on the performance. Additionally, in MSR also the *generalist* needs tuning. We performed randomized grid parameter search to find good parameters for both approaches. As metric we used the distance between the target and the generated outputs computed via fast dynamic-time-warping [11].

4.1 Sine wave generation

We consider first a sine wave generator modelling $y = a \cdot sin(b \cdot x)$. The goal of the trained generator is to produce a sine wave with the given amplitude a and frequency b. We vary the frequency in the range $[0.2, 0.6]$ with step size 0.25 and the amplitude in the range $[0.5, 2]$ with step size 0.75. For each pattern, 500 steps were used for training and testing. The last 100 steps of the best LOOCV generation results are shown in Fig. 4. While MSR is able to generate a sine wave with a given amplitude and frequency, the classic ESN generator fails to produce the sine wave with the lowest frequency and highest amplitude (cf. Fig. 4 bottom left).

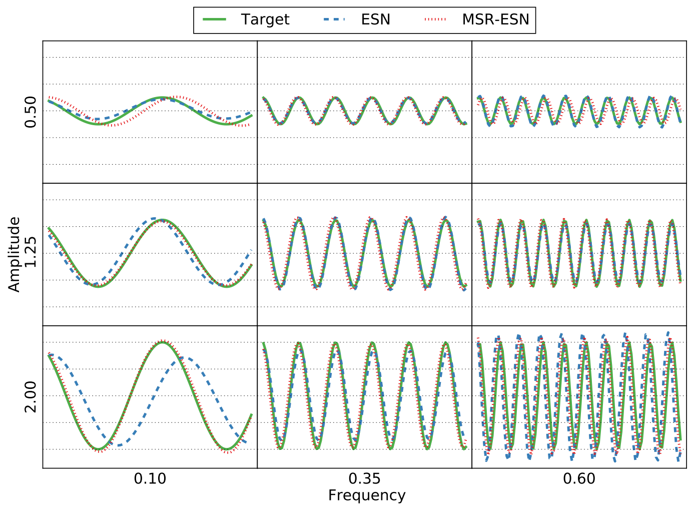

Fig. 4: Sine wave generator with ESN and MSR-ESN. Each cell depicts the LOOCV test generation results over the last 100 generated steps for the denoted frequency and amplitude.

4.2 Skewed figure eight generation

As second task we consider the two-dimensional figure eight pattern:

$$\begin{aligned} y_1 &= sin(x) \\ y_2 &= a \cdot sin(2x - b) + (1 - a) \cdot cos(x + b), \end{aligned}$$

where a controls the shape and b the skewness. When y_2 is plotted over y_1, $(a = 0, b = 0)$ corresponds to a circle and $(a = 1, b = 0)$ to the figure eight. We varied a and b in the range $[0, 1]$ with step size 0.5 and recorded the resulting nine patterns for 300 steps.

MSR produces the target signal with high precision, while the classic ESN shows a relative strong deviation (cf. Fig. 5). A version with a constant b, where only a was varied, posed no problem for either approach.

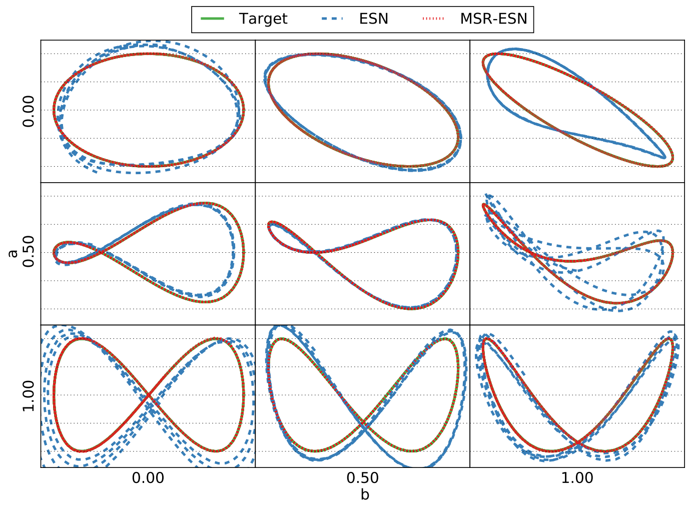

Fig. 5: Figure eight generator with ESN and MSR-ESN. Each cell depicts the LOOCV generation results for the corresponding values of the parameters (a, b).

4.3 Teardrop generation

The teardrop curve is defined as:

$$y_1 = cos(x) \tag{5}$$
$$y_2 = sin(x) \cdot sin^m(0.5x). \tag{6}$$

We varied m uniformly in the range $[2, 10]$ with step size 2 and recorded each pattern for 300 steps. While both ESN and MSR-ESN capture the overall shape, neither is able to create a new curve with precision (cf. Fig. 6).

4.4 More complex tasks

We also experimented with a parameterized multiple superimposed oscillator (P-MSO) in it's simplest form: $y = sin(a \cdot x) + sin(b \cdot x)$, and varied (a, b) in the range $[0.1, 0.6]$. We were, however, unable to train either architecture successfully. This is not surprising, considering that a (non-parameterized) MSO is not an easy task for ESNs and requires additional measures to solve (cf. [12]).

The ability of an ESN to generate each pattern places a natural limit on what can be learned - if an ESN can not learn a single pattern, than it will not

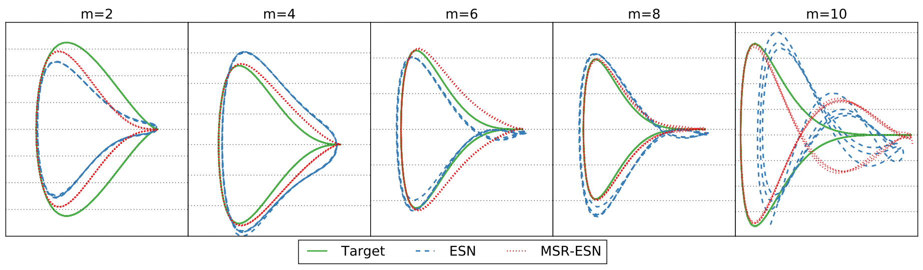

Fig. 6: Teardrop generator with ESN and MSR-ESN. Each cell depicts the LOOCV generation results for the corresponding value of the parameter m.

be possible to generate an ESN which can, or train an ESN to produce multiple patterns.

5 Discussion

The training of a single network to generate different patterns presents two challenges. First, the number of patterns that the network can learn is inherently limited. Similar patterns, as in the case of parameterized patterns, might require less memory, but might also interfere with each other in the state space because of their similarity. Second, the network must be able to change it's output according to the control parameter - it has to be able to reach the attractor corresponding to the control parameter pattern from any state. Both challenges were tackled recently by Jaeger's Conceptors [8]. However, the conceptors were used for morphing between different patterns, and not to learn parameterized patterns - it is unclear, how the conceptor concept can be extended to learn to generate patterns for new control parameter values.

MSR bypasses both challenges by creating networks tailored to each pattern. The basic assumption is, that similar control parameter values result in similar sequences and that the *generalist* can learn this relationship.

6 Conclusion

In this paper we introduced regression in the model space of ESNs for parameterized pattern generation. In contrast to other approaches, where a single ESN is trained to generate different patterns, in our modular approach for each pattern a specialist ESN (more precise: a readout) is created. The specialist ESN then autonomously generates the pattern. An evaluation on several synthetic datasets showed that MSR-ESN can generate parameterized patterns with a higher precision than a single ESN.

The successful application to the synthetic datasets shows that for some tasks the readout weights can be expressed as a function of the control parameters and learned from few examples. Further research is required to assess whether the recurrent network weights can also be learned from the control parameters and to extend the approach to more complex tasks.

Acknowledgments. This project is funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster Competition "it's OWL" (intelligent technical systems OstWestfalenLippe) and managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the contents of this publication.

References

1. Aswolinskiy, W., Reinhart, F., Steil, J.: Modelling parameterized processes via regression in the model space. In: European Symposium on Artificial Neural Networks (ESANN) (2016)
2. Aswolinskiy, W., Reinhart, F., Steil, J.: Time series classification in reservoir- and model-space: a comparison. In: Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR) (2016), accepted
3. Chen, H., Tino, P., Rodan, A., Yao, X.: Learning in the model space for cognitive fault diagnosis. IEEE Trans. on Neural Networks and Learning Systems 25(1), 124–136 (2014)
4. Chen, H., Tang, F., Tino, P., Yao, X.: Model-based kernel for efficient time series analysis. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 392–400 (2013)
5. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: IEEE International Joint Conference on Neural Networks. vol. 2, pp. 985–990 (2004)
6. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. GMD Technical Report 148, 34 (2001)
7. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach. GMD-Forschungszentrum Informationstechnik (2002)
8. Jaeger, H.: Controlling recurrent neural networks by conceptors. arXiv preprint arXiv:1403.3369 (2014)
9. Li, J., Jaeger, H.: Minimal energy control of an esn pattern generator. Jacobs University technical report (26) (2011)
10. Li, J., Waegeman, T., Schrauwen, B., Jaeger, H., et al.: Frequency modulation of large oscillatory neural networks. Biological cybernetics 108(2), 145–157 (2014)
11. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intelligent Data Analysis 11(5), 561–580 (2007)
12. Steil, J.J.: Several ways to solve the mso problem. In: European Symposium on Artificial Neural Networks (ESANN). pp. 489–494 (2007)
13. Wyffels, F., Schrauwen, B.: Design of a central pattern generator using reservoir computing for learning human motion. In: Advanced Technologies for Enhanced Quality of Life, 2009. AT-EQUAL'09. pp. 118–122. IEEE (2009)