Neurocomputing 000 (2017) 1–9

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Modelling of parametrized processes via regression in the model space of neural networks

Witali Aswolinskiy[a,*], René Felix Reinhart[b], Jochen Jakob Steil[c]

[a] Research Institute for Cognition and Robotics - CoR-Lab, Universitätsstraße 25, 33615 Bielefeld, Germany
[b] Fraunhofer Research Institution for Mechatronic Systems Design IEM, Zukunftsmeile 1, 33102 Paderborn, Germany
[c] Technische Universität Braunschweig, Institute for Robotics and Process Control, Mühlenpfordtstr. 23, 38106 Braunschweig, Germany

ARTICLE INFO

Article history:
Received 8 July 2016
Revised 14 October 2016
Accepted 21 December 2016
Available online xxx

Keywords:
Time series modelling
Model space
Reservoir computing
Extreme learning machine
Echo state network

ABSTRACT

We consider the modelling of parametrized processes, where the goal is to model the process for new parameter value combinations. We compare the classical regression approach to a modular approach based on regression in the model space: First, for each process parametrization a model is learned. Second, a mapping from process parameters to model parameters is learned. We evaluate both approaches on two synthetic and two real-world data sets and show the advantages of the regression in the model space.

1. Introduction

Many processes in nature and technology depend on the environment or the context. For example, the water freezing process depends on the initial water temperature (Aristotle–Mpemba effect). More generally, a chemical process may depend on temperature and pressure, or a mechanical process may depend on the applied forces and material properties. For instance, the dynamics of robotic manipulator motions can be considered as a process parametrized by the load the robot is carrying. Models of such parametrized processes are important in many applications, e.g. for optimization and control of production processes.

We distinguish between process parameters $\boldsymbol{p}$, which do not change during the process, e.g. load mass, and the process inputs $\boldsymbol{x}$, e.g. time. The constancy of the process parameters during the process separates our definition from the contextual features defined in [1,2]. The process parameters provide a context or environment for the process and shape the process output. They are defined over a continuous domain, e.g. masses, temperatures, etc., otherwise specialized models for each discrete process parameter configuration could be learned. We target the generalization of process models to novel process parameters.

The classical, data-driven modelling approach for parametrized processes is to train a regressor using the combination of process parameters and process inputs as model inputs (see Fig. 1 (top left)). If, for example, the process inputs are a robot motion trajectory and the process parameter is the load the robot is carrying, then a single model input would consist of a trajectory step combined with the mass of the load. Since the load does not change during the execution of the trajectory, the mass value would be constant. After a change of the load, a model input would consist of the new mass value combined with the trajectory value. This model input encoding leads to an increased training data demand to cover the higher-dimensional input space spanned by the combination of process parameters and process inputs.

In contrast to this monolithic approach, we propose a modular approach, which separates the learning of process parameters and process inputs utilizing learning in the model space [3]. First, for each process parameter combination, we learn a model for the process given the process input. This step yields specialized model parameters for the respective process parameters. Second, we learn a mapping from process parameters to the learned model parameters—a map from the process parameter space to the space of process models. Hence the term *regression in the model space*. We denote the lower-level process models as *specialist models*, and the higher-level regression model from process parameter space to model parameter space as *generalist model* in this paper (see Fig. 1 (top right)). By decoupling the learning of specialist and generalist models, the dimensionality of the input data for both models is smaller and thus better generalization from fewer samples can be achieved.

* Corresponding author.
E-mail address: waswolinskiy@cor-lab.uni-bielefeld.de (W. Aswolinskiy).

http://dx.doi.org/10.1016/j.neucom.2016.12.086
0925-2312/

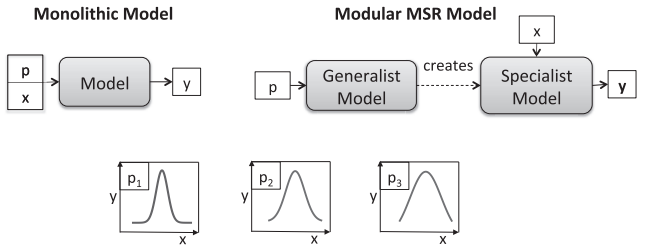

Fig. 1. Modelling of parametrized processes with a monolithic (top left) and a modular architecture (top right). On the bottom three exemplary instantiations of a parametrized process are depicted: The process output y is a function of the process input $\boldsymbol{x}$ and depends on the process parametrization $\boldsymbol{p}_i$.

Learning in the model space was previously mainly applied to time series classification [3,4]. Our contribution is the extension of the approach to *regression* in the model space, which we initially presented in [5] and applied later to robotics in [6]. Here, we extend the analysis of model space regression to the usage of Echo State Networks (ESNs, [7]) as *specialist models* and analyse the interpolation and extrapolation capabilities of model space regression on a more extensive set of examples. A conceptually similar approach was used in robotics for learning of parameterizable motor skills based on dynamic motion primitives [8–10], where mappings from task parameters to parameters of motion primitive models are learned. While the work in [10] argues for a non-modular learning of parameterized motor skills due to a more compact representation, we here show that the modular regression in the model space is beneficial with respect to the demand for fewer training data.

An earlier approach related to regression in the model space is context-dependent neural networks [2]. In contrast to the work in [2], which used iterative schemes to train context-dependent network weights, we enable efficient, one-shot learning techniques by rigorously separating the training of specialized process models for a particular context and the training of the generalist models. We achieve this using either Echo State Networks (ESNs, [7]) or Extreme Learning Machines (ELMs, [11]) as *specialist models*. Some light has been shed on the advantageous properties of the model space of ESNs for time series classification in [4,12].

The remainder of this paper is structured as follows. In Section 2, we briefly describe ELMs and ESNs and then outline in Section 3, how to accomplish regression in the model space of ELMs and ESNs. In Section 4, we apply our approach to two synthetic and two real-world examples and compare the results to the classical monolithic approach. The paper closes with a discussion and some concluding remarks.

2. Echo State Networks (ESN) and Extreme Learning Machines (ELM)

We first review the ESN and ELM network architectures before we explain how to apply regression in the model space using these models in the next section.

ESNs and ELMs are neural networks, which comprise three layers: An input layer $\boldsymbol{u} \in \mathbb{R}^I$, a hidden layer $\boldsymbol{h} \in \mathbb{R}^N$ with N hidden neurons, and a linear output or readout layer $\boldsymbol{y} \in \mathbb{R}^O$.

In ESNs the hidden layer is a reservoir of recurrently connected neurons, which provide a non-linear fading memory of the inputs. The reservoir states $\boldsymbol{h}$ and the readouts $\boldsymbol{y}$ are updated according to

$$\boldsymbol{h}(k) = (1-\lambda)\boldsymbol{h}(k-1) + \lambda a(\boldsymbol{W}^{rec}\boldsymbol{h}(k-1) + \boldsymbol{W}^{in}\boldsymbol{u}(k) + \boldsymbol{b}) \quad (1)$$

$$\boldsymbol{y}(k) = \boldsymbol{W}^{out}\boldsymbol{h}(k), \quad (2)$$

where $\lambda \in (0, 1]$ is the leaking rate, $a(\cdot)$ the activation function applied element-wise to the neuron inputs, e.g. hyperbolic tangent or logistic, $\boldsymbol{b} \in \mathbb{R}^N$ the neuron biases, $\boldsymbol{W}^{rec} \in \mathbb{R}^{N\times N}$ the recurrent weight matrix, $\boldsymbol{W}^{in} \in \mathbb{R}^{N\times I}$ the weight matrix from the inputs to the reservoir neurons and $\boldsymbol{W}^{out} \in \mathbb{R}^{O\times N}$ the weight matrix from the reservoir neurons to the readouts $\boldsymbol{y}$. $\boldsymbol{W}^{in}$, $\boldsymbol{W}^{rec}$ and $\boldsymbol{b}$ are initialized randomly and remain fixed. $\boldsymbol{W}^{rec}$ is scaled to fulfil the Echo State Property (ESP, [7]). The ESP is typically achieved by scaling the spectral radius of $\boldsymbol{W}^{rec}$ to be smaller than one. The scaling of $\boldsymbol{W}^{in}$ is task-dependent and has strong influence on the network performance.

In ELMs the hidden layer is feed-forward and the output is computed by

$$\boldsymbol{y}(\boldsymbol{u}) = \boldsymbol{W}^{out} a(\boldsymbol{W}^{in}\boldsymbol{u} + \boldsymbol{b}). \quad (3)$$

The ELM has the universal approximation capability [13].

In both methods, i.e. ELMs and ESNs, the readout weights $\boldsymbol{W}^{out}$ are learned with ridge regression:

$$\begin{aligned}\boldsymbol{W}^{out} &= \arg\min_{\boldsymbol{W}}\left(\|\boldsymbol{H}\boldsymbol{W}^T - \boldsymbol{T}\|^2 + \alpha\|\boldsymbol{W}\|^2\right) \\ &= \left(\boldsymbol{H}^T\boldsymbol{H} + \alpha\boldsymbol{I}\right)^{-1}\boldsymbol{H}^T\boldsymbol{T},\end{aligned} \quad (4)$$

where $\boldsymbol{H}$ are the collected neuron activations, $\boldsymbol{T}$ the target values and $\alpha \geq 0$ the regularization strength. The ELM and ESN network structure is depicted in Fig. 2.

ELMs and ESNs belong to the category of random projection techniques, which are based on the idea that a combination of random features can lead to a sufficiently rich encoding of the inputs. Since only the output weights are trained, efficient one-shot learning e.g. by linear regression can be used instead of gradient descent. Compared to neural networks trained with backpropagation, usually a higher number of neurons is used to provide a sufficiently rich combination of features and to increase the memory in case of ESNs.

3. Model Space Regression (MSR)

For the learning of parametrized process models, we present and compare two approaches: The classical, monolithic approach and a modular approach using the concept of regression in the model space.

In the monolithic approach, the process inputs $\boldsymbol{x} \in \mathbb{R}^A$ and the process parameters $\boldsymbol{p} \in \mathbb{R}^B$ are combined, which results in the input space $I \in \mathbb{R}^{A+B}$. The monolithic process model $f : I \rightarrow \mathbb{R}^O$ is then learned traditionally using both process inputs and process parameters as model inputs (see Fig. 1 (left)).

In Model Space Regression (MSR), the modelling is made in two steps: First, a *generalist* model creates for a given process parameter combination $\boldsymbol{p}$ a *specialist* model. For ELM or ESN specialist models, the *generalist* model generates readout matrices $\boldsymbol{W}^{out}(\boldsymbol{p})$ for the desired process parameters $\boldsymbol{p}$. Second, the *specialist* model computes from the process input the process output. Since $\boldsymbol{x}$ and $\boldsymbol{p}$ are processed separately in MSR, the input spaces $I_{Generalist} \in \mathbb{R}^A$ for the generalist and $I_{Specialist} \in \mathbb{R}^B$ for the specialist are smaller than in the monolithic approach. Fig. 1 (right) depicts the modular approach with regression in the model space.

In this paper, we use ELMs and ESNs as specialist models for MSR. For the generalist models, we either use an ELM or linear (ridge) regression. For the monolithic modelling and the *specialists* in MSR, we use ELMs for stateless processes and ESNs for state-dependent processes. In state-dependent processes, the output depends not only on the current input, but also on the previous inputs. In this case, the dynamic reservoir of the ESN provides the necessary short-term memory.

While training of the monolithic models proceeds as in classical supervised learning, the training of the modular approach proceeds in two phases:

Please cite this article as: W. Aswolinskiy et al., Modelling of parametrized processes via regression in the model space of neural networks, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2016.12.086