ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

A flat neural network architecture to represent movement primitives with integrated sequencing

Andre Lemme[1] and Jochen Steil[1] *

1- Institute for Cognition and Robotics (CoR-Lab) & Faculty of Technology
Bielefeld University, 33615 Bielefeld - Germany

Abstract. The paper proposes a minimalistic network to learn a set of movement primitives and their sequencing in one single feedforward network. Utilizing an extreme learning machine with output feedback and a simple inhibition mechanism, this approach can sequence movement primitives efficiently with very moderate network size. It can interpolate movement primitives to create new motions. This work thus demonstrates that an unspecific single hidden layer, that is a flat representation is sufficient to efficiently compose complex sequences, a task which usually requires hierarchy, multiple timescales and multi-level control mechanisms.

1 Introduction

Movement primitives (MPs) in form of dynamical attractor systems are a common paradigm to encode motion skills in robotics, whereas recent research has turned to investigate libraries of MPs. In this context, it is discussed, how to sequence movement primitives (MPs) to compose complex trajectories, usually independently from the question on how to represent the MPs themselves. Any such approach needs a minimum control logic for sequencing MPs (i) to index the MPs in a larger architecture and (ii) to parametrize the execution by the starting point and the goal. It could e.g. be realized either with brute force by memorizing respective data or by learning a respective sequence with any method for sequence learning (see [1]). For instance, Kulic et al. store the sequential activation of specific MPs in [2], while Rhodes et al. [3] organize short sequences as task specific chunks as cognitively adequate units. A respective neural architecture was proposed in [4] in form of continuous-time recurrent neural networks (CTRNNs) to model context-sensitive selection of MPs dependent on sensory data. These approaches assume that the MPs are given and are neurally sequenced. In [5], the stochastic continuous-time recurrent neural network (S-CTRNN) is introduced for learning trajectories by extracting the stochastic structures hidden in demonstrated training data. This more holistic approach allows for learning and smooth blending of multiple time series in one single network, but the parameterization of the learned time series is encoded in context parameters, which are not human readable.

In this paper, an ELM (extreme learning machine) neural network with output feedback represents both the MPs and the temporal sequence of MPs in a single flat representation, where the MPs themselves are represented through the parameters of a dynamic movement primitive (DMP) [6, 7]. Additionally, this shared representation is exploited to interpolate learned MPs.

*This research has received funding from them EC's FP7/2007-2013 – Cognitive Systems, Interaction, Robotics – under grant agreement No 248311 - AMARSi.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

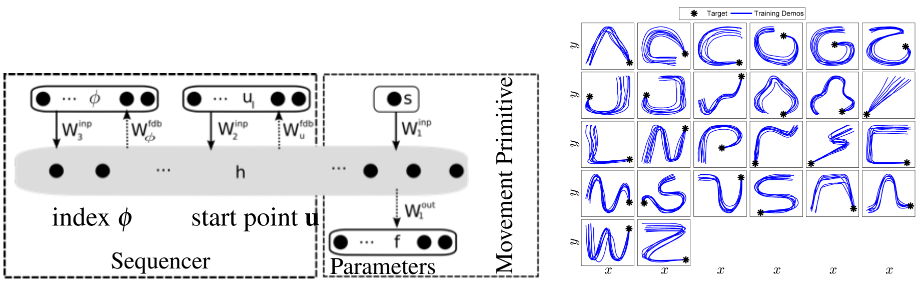


Fig. 1: Left: A flat representation of a sequence of starting points $\mathbf{u}_l$ and indices ϕ together with the MP itself. Right: LASA data set of handwriting motions for training.

2 A flat neural learning architecture for movement generation

Representation of movement primitives: We use a variation of the original DMP paradigm [6] as proposed in [7], which is invariant to the relative position of the movements's start and goal position. It comprises a second order *spring-damper* dynamics $\tau\ddot{\mathbf{u}}_t = K(\mathbf{g}-\mathbf{u}_t) - D\dot{\mathbf{u}}_t - Ks(\mathbf{g}-\mathbf{u}_0) + K\mathbf{f}(s)$, where K, D are stiffness and damping constants with $D = 2\sqrt{K}$ to generate a critical damped system [7]. It is coupled with a canonical system: $\dot{s}_t = -\frac{1}{M_{\mathrm{mp}}}$, where the specified number of time steps M_{mp} determines the motion generation time. The spring-damper part attracts the trajectory to the goal $\mathbf{g}$, while the perturbation term(“force”, $\mathbf{f}(s)$) is learned from demonstration to model the motion's shape. Global asymptotic stability is ensured if the perturbation $\mathbf{f}$ becomes zero at the end of the movement, which results in a linear convergence to the goal point. The training data is given by recording $\mathbf{u}_t, \dot{\mathbf{u}}_t, \ddot{\mathbf{u}}_t$ from example motion

$$\mathbf{f}(s) = -(\mathbf{g}-\mathbf{u}_t) + \frac{D}{K}\dot{\mathbf{u}}_t + s(\mathbf{g}-\mathbf{u}_0) + \frac{\tau}{K}\ddot{\mathbf{u}}_t. \tag{1}$$

In previous work, the function f has been approximated as a weighted sum of Gaussian basis functions. We here propose to use an ELM to learn $\mathbf{f}(s)$. Without loss of generality, the learning of MPs parameters is performed in a normalized space, where the goal point is always the origin. This is beneficial as only the initial start point relative to the goal point $\mathbf{u}_l$ of the demonstrations is required as to be stored for each MP.

Extreme learning machine with output feedback: The proposed architecture is illustrated in Fig. 1. The DMP force term $\mathbf{f}(s)$ (*how*), the starting points $\mathbf{u}_l$ (*from where*) and indices ϕ (*which MP*) are learned by the network, the latter two are fed back to the network to close the loop and realizes the recurrent sequencer. The following ELM network dynamics arise:

$$\mathbf{h}(k) = \sigma(W_1^{\mathbf{inp}} s(k) + W_2^{\mathbf{inp}} \mathbf{u}_l + \mathbf{W}_3^{inp} \phi + b), \tag{2}$$

$$\hat{\mathbf{f}}(k) = W_1^{\mathbf{out}} \mathbf{H}(k), \tag{3}$$

where $\mathbf{h} \in \mathbb{R}^R$ gives the hidden state and $\phi \in \mathbb{R}^I$, $\mathbf{u}_l \in \mathbb{R}^d$ denote the input of the network. These inputs are respectively connected to the hidden layer through the input matrices $\mathbf{W}_3^{inp} \in \mathbb{R}^{R\times I}$ and $\mathbf{W}_2^{inp} \in \mathbb{R}^{R\times d}$, where I is the expected number of MPs stored in this network and d is the dimension of the normalized MP space. The bias of the hidden layer neurons is denoted by b and all input weight matrices remain fixed after random initialization. The activation function $\sigma(x) = 1/(1+\exp(-x))$ is applied to each neuron in the hidden layer, which is connected to the output by $\mathbf{W}_1^{out} \in \mathbb{R}^{d\times R}$ and gives the approximation of the perturbation of the transformation system given in Eq. (1). At the start of the movement generation the input is $s = 1$ and decreases according to $\dot{s_t} = -\frac{1}{M_{\mathrm{mp}}}$ to $s = 0$, which corresponds to the end of the motion.

Learning an activation sequence: The task of the sequencer (Fig. 1) is to activate a number of MPs in a defined way. First, a MP needs to be found in the library, which requires an index that uniquely identifies the chosen MP. Note that the index does not need any semantics and in the following a one-of-K-coding vector ϕ is used. Second, the initial conditions of the MP need to set the start point $\mathbf{u}_l$, while the goal was normalized at the origin and is implicitly given.

The representation of a complex movement as a sequence of MPs thus can be modeled by the ELM with output feedback through the sequencer part in Fig. 1 with just two simple additional conditions that could easily be realized neurally through inhibitors. The initial condition is set as identification and start point of the first MP. Then, during the movement generation i.e. $s > 0$ the input $(\Phi(k), \mathbf{u}_l(k))$ stays fixed, which effectively cuts off the recurrence in the sequencer part, i.e. Eq. (5) and Eq. (6) below are not applied. This parametrizes the particular MP, due to the unique impact of the sequencer input on the global representation, that is $\mathbf{f}(s)$ is generated by the network and the DMP spring-damper equation is integrated over time.

When the particular MP is converged and the input s is zero, then the following update of network dynamics are performed:

$$\mathbf{h}(k+1) = \sigma(\mathbf{W}_2^{inp}\mathbf{u}_l(k) + \mathbf{W}_3^{inp}\phi(k) + \mathbf{b}), \tag{4}$$

$$\phi(k+1) = \mathbf{W}_\phi^{fdb}\mathbf{h}(k), \tag{5}$$

$$\mathbf{u}_l(k+1) = \mathbf{W}_u^{fdb}\mathbf{h}(k), \tag{6}$$

The hidden layer is fed back to the inputs by $\mathbf{W}_\phi^{fdb} \in \mathbb{R}^{I\times R}$ and $\mathbf{W}_u^{fdb} \in \mathbb{R}^{d\times R}$. This effectively recalls the next MP index and initial conditions in the learned sequence.

3 Learning methodology

The training data is specified by at least one normalized demonstration (w.r.t. the goal point) to record a set of $\mathbf{u}_t, \dot{\mathbf{u}}_t, \ddot{\mathbf{u}_t}$ and the respective index ϕ and start point $\mathbf{u}_l$. The input s is given by a decreasing linear function from $s = 1$ to $s = 0$ in M_{mp} steps, where M_{mp} is the number of samples in the demonstrated trajectory. The input $(\phi, \mathbf{u}_l)$ stays fixed during the learning of the DMP perturbation terms. The hidden layer states are obtained from applying $s(j)$ as input and harvesting the hidden states in the matrix $\mathbf{H}(S(j)) = (\mathbf{h}(s(1)), \ldots, \mathbf{h}(s(M_{\mathrm{mp}})))$, where $j \in [1 \ldots M_{\mathrm{mp}}]$ is the current step. The corresponding targets are given by $\mathbf{T}(j) = (f(s(1)), \ldots, f(s(M_{\mathrm{mp}})))$, where f is given by Eq. (1). The sequence of of MPs is learned by closing the loop (e.i. Eq. (5) and Eq. (6) are applied). The input at iteration k in the sequence is given by $\mathbf{x}(k) = (\mathbf{u}_l(k), \phi(k))$ and

the corresponding target $\mathbf{t}(k) = (\mathbf{u}_l(k+1), \phi(k+1))$. If M_{tr} is the length of the training sequence, then a sequence is given by $\mathbf{S} = (\mathbf{X}, \mathbf{T}) = (\mathbf{x}(n), \mathbf{t}(n)) : n = 1 \dots M_{\text{tr}}$.
Efficient online learning: We use a version of recursive least squares (OS-ELM [8]) to incrementally learn or refine MPs if new training data is available. To learn the first MP with $k = 0$ (the first demonstration needs to include at least as many $s-$steps as the number of hidden neurons), we initialize $\mathbf{W}_0^{out} = \mathbf{P}_0\mathbf{H}_0^T\mathbf{T}_0, \mathbf{P}_0 = (\mathbf{H}_0^T\mathbf{H}_0 + \varepsilon\mathbb{1})^{-1}$, where ε is the regression parameter and $\mathbb{1}$ is the identity matrix. Further primitives are added incrementally:

$$\mathbf{P}_{k+1} = \mathbf{P}_k\mathbf{H}_{k+1}^T(\mathbb{1} + \mathbf{H}_{k+1}\mathbf{P}_k\mathbf{H}_{k+1}^T)^{-1}\mathbf{H}_{k+1}\mathbf{P}_k, \tag{7}$$

$$\mathbf{W}_{k+1}^{out} = \mathbf{W}_k^{out} + \mathbf{P}_{k+1}\mathbf{H}_{k+1}^T(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\mathbf{T}_k). \tag{8}$$

4 Composition of complex trajectories

Recall activation sequence: We start by evaluating the performance of the sequencer part, which means to sequence the ϕs in the right order with the correct starting points $\mathbf{u}_l$ of the motion relative to the end point. An update of the sequencing part (see Fig. 1) according to Eq. (4) generates a new starting point to parametrize a new MP and an index to select the respective MP, if the input is $s = 0$. An example of a reproduced learned initial point sequence is given in Fig. 2. Here we used $\varepsilon_{\text{Seq}} = 1$ for adapting the feedback weights $\mathbf{W}_*^{fdb}$. Note that no preprocessing of the starting points (see Fig. 2(circles)) is necessary. After learning the network generates a mean starting point (see Fig. 2(cross)) corresponding to the primitive indexed by ϕs (Fig. 2(numbers)).

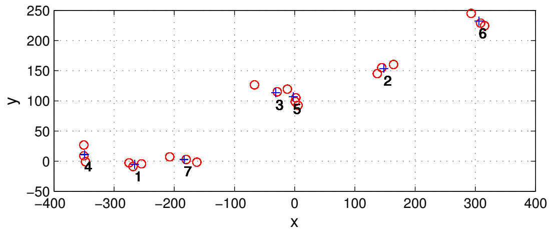
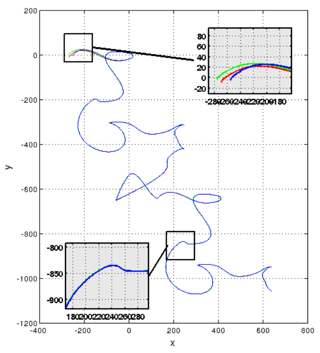


Fig. 2: Reproduction of the learned sequence of starting points (in mm). Circles mark the starting points of the training data and the cross mark the reproduced starting point of the MP indexed by $1-7$.

Network capacity: For further evaluation, we trained a single network (Fig. 1) to represent all $M = 20$ motions in the LASA dataset (Fig. 1) together with a respective sequencing, where each of the motions is specified by $N = 3$ demonstrations. After learning, the reproduction performance of the motion generation with respect to the size of the hidden layer is measured, which is one critical parameter of the overall approach. The order in which the motion patterns are presented is randomized. The reproduction performance of the monolithic approach over $k = 10$ network initialization for the hidden layer size of $R \in \{40, \dots, 200\}$ is evaluated. The input matrices and biases b_i are initialized randomly from uniform distributions in $[-10, 10]$ and $[-1, 1]$ respectively. The DMP transformation system is initialized with $K = 200$. The reproduction error is measured by the point wise root mean square error (RMSE) over all motions learned by the network: $RMSE = \frac{1}{N}\sum_N \sqrt{\frac{1}{N_{tr}}\sum_i^{N_{tr}} ||u_n(i) - \hat{u}_n(i)||^2}$, where $\mathbf{u}$ is the position at time-step i and $\hat{\mathbf{u}}$ is the corresponding reproduction. For this experiment the ϕ index of the desired primitive is set manually in the input layer. From Fig. 3 we obtain that the



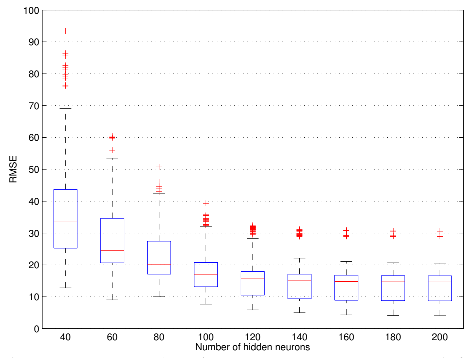


Fig. 3: Left: Sequencing seven MPs (Fig. 4) repeated twice. Three runs started from different initial points (zoomed). Right: Error for learning of $M = 20$ motion patterns with different hidden layer sizes.

performance is robust with respect to the number of neurons in the hidden layer for the very modest number of at least 140 neurons.

Combined sequencing and motion generation: In Fig. 3 (left), a complex trajectory is shown which is composed of seven MPs trained in one network together with the proper activation sequence of these MPs. The network can be exploited with a minimal overhead of additional "control". While executing the MP, the feedback from the network to the sequencing part is inhibited i.e. Eq. (5) and Eq. (6) are not applied, until the MP is finished. After the motion generation (i.e. $s = 0$) the Eq. (5) and Eq. (6) are applied and a new ϕ and u_l can be recalled from the network.

Now the next motion primitive can start, as selected through the newly updated index. The complex movement is repeated three times with the three different initial positions (see Fig. 3, zoomed area upper right corner). The overall sequence structure repeats itself, because a cyclic loop was trained for the sequencer. Note that after the first primitive ('sine wave'), the next primitive ('Sshape') all three repetitions of the complex motions are aligned (see Fig. 3, zoomed area bottom left corner), which is due to the generalization capability of the DMP to converge back to its initial form.

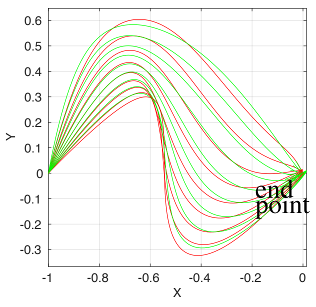


Fig. 4: Two motions interpolated through mixture coefficients at indexing input (red/dark) vs. MP with external computed sum of forces (green/light).

Mixing of movement primitives: Due to the one-of-K coding scheme, only one neuron in the identification vector ϕ is active to represent a single learned MP. A natural generalization, which can be done without further external control logic, is to overlay primitives through co-activation. Then the expectation is that the global representation creates a mixture of the originally learned motions. Fig. 4 shows how two motions are interpolated by applying convex combinations of indexing input ($\phi_1 = (1, 0)$ and $\phi_2 = (0, 1)$) in seven intermediate steps. The network generated motion is blended in a respective additive mixture of two

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

separately trained DMP, which shows that the flat network representation produces by means of internal interpolation highly plausible motion mixtures that resemble externally combined ones. (Further learning parameters: $\varepsilon_{\mathrm{MP}} = 0.5$ i.e. $\mathbf{W}_1^{out}$, size of the hidden layer $R = 100$. All other variables are initialized as described in the previous section.) Both MPs ϕ_1 and ϕ_2 reproduced accurately the training motions and a smooth transition from ϕ_1 or ϕ_2 occurs by simply by changing the input linearly from ϕ_1 to ϕ_2.

5 Very short discussion on the architecture perspective

This paper introduces a flat single-layer neural architecture to learn parametrizations of dynamic movement primitives together with their sequencing in a flexible and efficient way, largely avoiding additional control overhead. From our current experience, there are no obvious scaling limits. But future work needs to further evaluate the capacity of the network to learn different motions, the generalization abilities e.g. to disturbed and multiple sequences, or the interpolation behavior. Still, besides implementing an elegant neural solution to a persistent research question, the presented work strongly questions a number of common notions in behavioral architectures. Seemingly, the motion sequencing task requires explicit hierarchy, multiple timescales and multi-level control mechanisms as present in most related work. But in our model, this is not the case. Time-scales are implicit, control is minimalistic, and a hierarchy is arguably not present: does the MP-representation trigger "top-down" the next initial conditions by releasing the inhibition of the sequencer loop, or does the sequencer "top-down" trigger the next MP ? Maybe more of the "architecture" is in the eye of the beholder than we usually assume. Thus we hope to explore such architectures further in future work, because they are interesting both from a practical and cognitive point of view.

References

[1] R. Amit and M. Matari. Learning movement sequences from demonstration. In *Proc. Int. Conf. on Development and Learning*, pages 203–208, 2002.

[2] D. Kulic, L. Dongheui, C. Ott, and Y. Nakamura. Incremental learning of full body motion primitives for humanoid robots. In *Proc. Int. Conf. on Humanoid Robots*, pages 326–332, 2008.

[3] B. Rhodes, D. Bullock, W. Verwey, B. Averbeck, and M. Page. Learning and production of movement sequences: Behavioral, neurophysiological, and modeling perspectives. *Human movement science*, 23(5):699–746, 2004.

[4] T. Luksch, M. Gienger, M. Mühlig, and T. Yoshiike. Adaptive movement sequences and predictive decisions based on hierarchical dynamical systems. In *Proc. Int. Conf. on Intelligent Robots and Systems*, pages 2082–2088, 2012.

[5] S. Murata, J. Namikawa, H. Arie, S. Sugano, and J. Tani. Learning to reproduce fluctuating time series by inferring their time-dependent stochastic properties: Application in robot learning via tutoring. *Transactions on Autonomous Mental Development*, PP(99):1–1, 2013.

[6] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Proc. Advances in Neural Information Processing Systems 15*, pages 1523–1530. MIT Press, 2003.

[7] D. Park, H. Hoffmann, P. Pastor, and S. Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Proc. Int. Conf. on Humanoid Robots*, pages 91–98, 2008.

[8] N. Liang, G. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *Transactions on Neural Networks*, 17(6):1411–1423, 2006.