Neurally Imprinted Stable Vector Fields

Andre Lemme, Klaus Neumann, Felix Reinhart and Jochen Steil

Institute for Cognition and Robotics (CoR-Lab) & Faculty of Technology
Bielefeld University, Universitätsstr. 25, 33615 Bielefeld - Germany

Abstract. We present a novel learning scheme to imprint stable vector fields into Extreme Learning Machines (ELMs). The networks represent movements, where asymptotic stability is incorporated through constraints derived from Lyapunov stability theory. We show that our approach successfully performs stable and smooth point-to-point movements learned from human handwriting movements.

1 Introduction

Vector fields are a common representation in different applications and can be used for instance to encode quantitative flow visualization, optical flow in computer vision and force fields in motor control.

In this context, approximating vector fields from sparse data is a typical scenario for learning algorithms. In [1], a superposition of irrotational basis fields is used to approximate a variety of vector patterns, where it is assumed that the data originate from a potential function. In [2], a combination of neural networks is used to reconstruct vector fields, where prior knowledge of inherent properties of vector fields is used to enhance the accuracy. Both approaches learn from sparse data that is uncorrelated in time and space using prior knowledge.

In case of learning from data that is correlated in time, like movement trajectories, different prior knowledge is required to guarantee the reliability of the estimate. An approach suitable for learning vector fields from trajectories is the Stable Estimator of Dynamical Systems approach (SEDS, [3]). It is based on a mixture of Gaussian functions where global asymptotic stability is ensured by fulfilling a specific Lyapunov function. In recent work [4], a superposition of two neural networks is used for movement generation. The stability is addressed by training the outputs of one network to implement the motion and the outputs of the other to generate dynamics towards this very motion. However, this approach can not guarantee the stability of the motion.

We propose a novel learning method for representing vector fields based on the Extreme Learning Machine (ELM, [5]) approach, which features efficient supervised learning. The main contribution is to devise a learning scheme for the ELM approach that incorporates stability constraints. These constraints are derived from a parameterized quadratic Lyapunov function, to learn vector fields which enforce stable movement generation. Several mechanisms have been developed in order to improve the performance of ELMs without focus on incorporating constraints. One idea to improve ELMs is to decrease the size of the hidden layer - the Optimally Pruned Extreme Learning Machine (OP-ELM, [6]). Those methods are orthogonal to the proposed technique and can thus be used in parallel.

2 Neural Networks for Movement Generation

The dynamical system that we consider for movement generation is autonomous and of first order, mapping positions $\mathbf{x}$ to vectors $\mathbf{v}$, which are integrated over time t:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \cdot \mathbf{v}(\mathbf{x}^t) \ , \tag{1}$$

where Δt is a time constant for discretization of the continuous dynamics. The initial state of this dynamical system is denoted by $\mathbf{x}^0$. We consider an ELM as depicted in Fig. 1 to encode vector fields. The ELM is a feed-forward neural network, that comprises three different layers of neurons: $\mathbf{x} \in \mathbb{R}^I$ denotes the input, $\mathbf{h} \in \mathbb{R}^R$ the hidden, and $\mathbf{v} \in \mathbb{R}^I$ the output neurons. The input is connected to the hidden layer through the input matrix $W^{\mathbf{inp}} \in \mathbb{R}^{R \times I}$ which remains fixed after random initialization. The read-out matrix is given by $W^{\mathbf{out}} \in \mathbb{R}^{I \times R}$ and is subject to supervised learning. For input $\mathbf{x}^t$ the output of the ith neuron is thus given by:

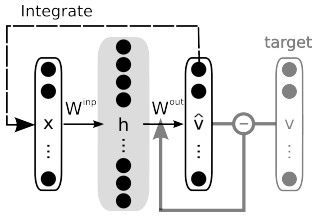

Fig. 1: ELM used in an integration-loop.

$$\hat{v}_i(\mathbf{x}^t) = \sum_{j=1}^{R} W_{ij}^{\mathbf{out}} f(a_j \sum_{n=1}^{I} W_{jn}^{\mathbf{inp}} x_n^t + b_j) \tag{2}$$

where slope a_j and bias b_j parameterize the component-wise Fermi function $f(x) = \frac{1}{1+e^{-x}}$ of the jth neuron in the hidden layer.

3 Implementation of Asymptotic Stability

Learning a vector field from a few training trajectories gives only sparse information of the shape of the entire vector field. Therefore, there is considerable need for generalization to spatial regions where no training data reside. The most important feature in the case of point-to-point movements is to converge to a given target. This target is described as a fixed-point attractor in the vector field. Learning this attractor without prior knowledge is especially hard because the training data comprises only a few training samples that encode the target.

In order to stabilize the dynamical system induced by the network, we recall the conditions for asymptotic stability of arbitrary dynamical systems defined by Lyapunov: a dynamical system is asymptotically stable at fixed-point $\mathbf{x}^* \in A$ in the compact and positive invariant region $A \subset \mathbb{R}^I$ if there exists a continuous and continuously differentiable function $L : A \to \mathbb{R}$

$$\textbf{(i)} \ \ L(\mathbf{x}^*) = 0 \ , \qquad \textbf{(ii)} \ \ L(\mathbf{x}) > 0 : \forall \mathbf{x} \in A, \mathbf{x} \neq \mathbf{x}^* \ , \tag{3}$$

$$\textbf{(iii)} \ \ \dot{L}(\mathbf{x}^*) = 0 \ , \qquad \textbf{(iv)} \ \ \dot{L}(\mathbf{x}) < 0 : \forall \mathbf{x} \in A, \mathbf{x} \neq \mathbf{x}^* \ . \tag{4}$$

We assume that the function L satisfies condition (**i**)-(**iii**). In order to obtain a learning algorithm for $W^{\mathbf{out}}$ that also respects condition (**iv**) of the Lyapunov function L, we analyze this condition by taking the time derivative of L:

$$\dot{L}(\mathbf{x}) = \frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{x}) = (\nabla_{\mathbf{x}}L(\mathbf{x}))^T \cdot \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = (\nabla_{\mathbf{x}}L(\mathbf{x}))^T \cdot \hat{\mathbf{v}} \tag{5}$$

$$= \sum_{i=1}^{I}(\nabla_{\mathbf{x}}L(\mathbf{x}))_i \cdot \sum_{k=1}^{R} W_{ij}^{\mathbf{out}} \cdot f(a_j \sum_{k=1}^{I} W_{jk}^{\mathrm{inp}} x_k + b_j) < 0 \ . \tag{6}$$

Note that $\dot{L}$ is linear in the output parameters $W^{\mathbf{out}}$ irrespective of the form of the Lyapunov function L. For a given point $\mathbf{u} \in A$, Eq. (5) and Eq. (6) define a linear constraint $\dot{L}(\mathbf{u}) < 0$ on the read-out parameters $W^{\mathbf{out}}$, which is implemented by a quadratic programming scheme introduced for ELMs in [7]. Whereas it is shown in [7] that a well-chosen sampling of points $U = (\mathbf{u}(1), \ldots, \mathbf{u}(N_{\mathrm{u}})) : \mathbf{u} \in \mathbb{R}^I$, is sufficient to generalize the incorporated discrete constraints to continuous regions in a reliable way. The read-out weights $W^{\mathbf{out}}$ are trained by solving the quadratic program with weight regularization:

$$W^{\mathbf{out}} = \arg\min_{W}(\|W \cdot H(X) - V\|^2 + \varepsilon\|W\|^2), \text{ subj. to: } \dot{L}(U) < 0 \ , \tag{7}$$

where the matrix $H(X) = (\mathbf{h}(\mathbf{x}(1)), \ldots, \mathbf{h}(\mathbf{x}(N_{\mathrm{tr}})))$ collects the hidden layer states obtained from a given data set $D = (X, V) = (\mathbf{x}(k), \mathbf{v}(k)) : k = 1 \ldots N_{\mathrm{tr}}$ for inputs X and the corresponding output vectors V and where ε is the regularization parameter.

4 Learning Stable Point-to-Point Movements

In order to analyze the impact of the stabilization mechanism introduced in Sec. 2, we perform movements learned from human-demonstrated handwriting movements (benchmark data set [8]). The used data is composed of three S-like trajectories with 250 samples each and the end-point located at the origin (see Fig. 2). The ELM is initialized with $R = 100$ neurons in the hidden layer. The slopes a_i are initialized with ones, biases b_i and components of $W^{\mathbf{inp}}$ are initialized randomly drawn from the uniform distribution on $[-1, 1]$. The regularization parameter is $\varepsilon = 10^{-5}$. The constraint set U consists of $N_{\mathrm{u}} = 2500$ samples drawn from the uniform distribution on the set $[-0.5, 2.1] \times [-1.1, 2.5]$, which covers the relevant region of the task space. Consider the Lyapunov function $L(\mathbf{x})$ of the following form:

$$L(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \cdot G^{*T}G^* \cdot (\mathbf{x} - \mathbf{x}^*) \ . \tag{8}$$

This Lyapunov function L will be employed for learning to implement asymptotic stability at the known fixed-point $\mathbf{x}^*$. Note that (**i**) - (**iii**) describes the general form of a Lyapunov function and are fulfilled if $G^{*T}G^*$ is positive definite. If a

data set D is given, G^* is chosen as:

$$G^* = \arg\min_{G \in \mathcal{G}} M_G \quad \text{with} \quad M_G = \sum_{k=1}^{N_{\text{tr}}} \Theta \left[\underbrace{(\mathbf{x}(k) - \mathbf{x}^*)^T \cdot G^T G}_{(\nabla_{\mathbf{x}} L(\mathbf{x}(\mathbf{k})))^T} \cdot \mathbf{v}(k) \right] , \qquad (9)$$

where $\mathcal{G} := \{G \in \mathbb{R}^{I \times I} : \lambda_i \in [\alpha, 1],\ \lambda_i \text{ is eigenvalue of } G\}$, α is a small and positive scalar, and Θ is the ramp function. The minimization operator in the left part of Eq. (9) can be formulated as a nonlinear program. We use successive quadratic programming based on quasi-Newton methods for optimization [9]. The function M measures the violation of condition (**iv**) by the training data with respect to G. The term $\Theta(\cdot)$ guarantees that only those samples $(\mathbf{x}(k), \mathbf{v}(k))$ are counted in M where the scalar product between $G \cdot (\mathbf{x}(k) - \mathbf{x}^*)$ and $G \cdot \mathbf{v}(k)$ is positive. To prevent an infinite stretching of $G^{*T} G^*$ through minimization of M we restrict the eigenvalues λ_i of G^* to $\alpha \leq \lambda_i \leq 1, \forall i = 1 \ldots I$. We set $\alpha = 0.1$ in the experiments.

Fig. 2 (left) illustrates an example of unstable estimation of a nonlinear dynamical system by an ELM trained without the usage of explicit stability constraints. In the areas close to the demonstrations, the trajectories converge

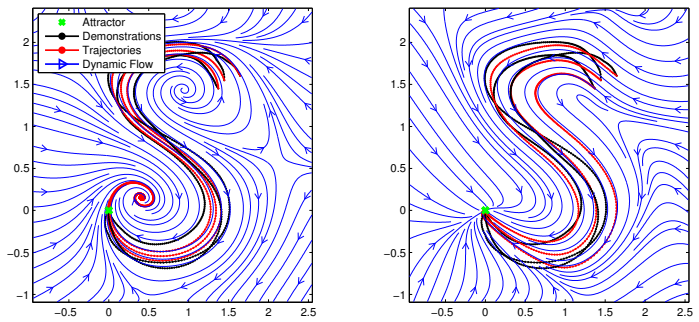

Fig. 2: The impact of the incorporation of asymptotic stability into the learning. Visualized dynamics of a network trained without stability constraints (left) and the same network trained with constraints for stabilization (right).

to an attractor next to the target. In other regions of the space, they either converge to spurious attractors or diverge. In contrast, Fig. 2 (right) shows the same network setup but trained with the stabilization method. The generated trajectories converge to the target, because the learning process enforces asymptotic stability. This ensures that the target is reached when starting from any point in the workspace. In the following we show an evaluation of the new stability mechanism, using two performance measures. The first measure is the root mean square error $E_{\text{tr}} = \sqrt{\frac{1}{N_{\text{tr}}} \sum_k \|\mathbf{v}(k) - \hat{\mathbf{v}}(\mathbf{x}(k))\|^2}$ evaluated on the training data, which quantifies the ability to approximate the training data. The second measure quantifies the stability of the dynamical system. For this measure we

chose $N_\mathrm{s} = 100$ starting points uniformly drawn from $[-0.5, 2.1] \times [-1.1, 2.5]$. The starting points are used to perform movements with $N_\mathrm{max} = 1000$ steps with step size $\Delta_t = 0.1$; the resulting states $\mathbf{x}(N_\mathrm{max})$ are recorded. If the end-point $\mathbf{x}(N_\mathrm{max})$ of the reproduced trajectory is in the vicinity of the desired attractor $\mathbf{x}^* = \mathbf{0}$, the end-point is recognized as converged (i.e. $\|\mathbf{x}(N_\mathrm{max}) - \mathbf{x}^*\| < \delta = 1$), otherwise as diverged. The distance $\mathrm{dist} = \|\mathbf{x}(N_\mathrm{max}) - \mathbf{x}^*\|$ from the converged points to the attractor and the number of converged points S are stored. All results are averaged over $N_\mathrm{ni} = 10$ different network initializations.

In Tab. 1 the results of the experiments for networks with and without stabilization for different regularization parameters ε are shown. The precision

	without constraints			with constraints		
$\ln \varepsilon$	dist	S/N_s	E_tr	dist	S/N_s	E_tr
-8	.431±.067	.686±.053	.208±.0006	.049±.014	1	.263±.0036
-6	.431±.018	.813±.031	.218±.0008	.043±.016	1	.283±.0043
-4	.401±.013	.917±.053	.235±.0027	.055±.014	1	.313±.0029
-2	.382±.010	.965±.044	.304±.0028	.020±.012	1	.377±.0026

Tab. 1: Network learning with and without constraints compared for different regularization parameters ε.

is given in the distance (dist) of the end-point to the desired attractor after convergence. The ratio S/N_s shows how many movements generated according to Eq. (1) converged to the desired target. Note that $S/N_\mathrm{s} = 1$ holds for all constrained networks. The results show that the stability ratio increases with growing regularization for networks trained without constraints. This induces a trade-off between stability and accuracy for the unconstrained ELM. For the explicitly stabilized ELMs, the trade-off is resolved, because all networks converge to the target independent of the regularization. Also the target is imprinted with a higher degree of precision (cf. Tab. 1 dist). In Fig. 3 we show movements learned from training shapes (black), generated trajectories (red), and the surrounding vector field (blue). Note that all networks produce stable movements which converge to the given attractor while respecting the training trajectories.

5 Conclusion

In this work we proposed a new learning scheme for Extreme Learning Machines (ELMs) to imprint stable vector fields and incorporate prior knowledge into the learning. Prior knowledge in form of linear stability constraints are derived from Lyapunov stability theory and incorporated efficiently by quadratic programming. In our experiments we considered a quadratic Lyapunov function as a proof of concept, but in general any Lyapunov function can be used. We showed that the new approach is sufficient to encode point-to-point movements accurately and that the trade-off between regularization and stability is resolved.

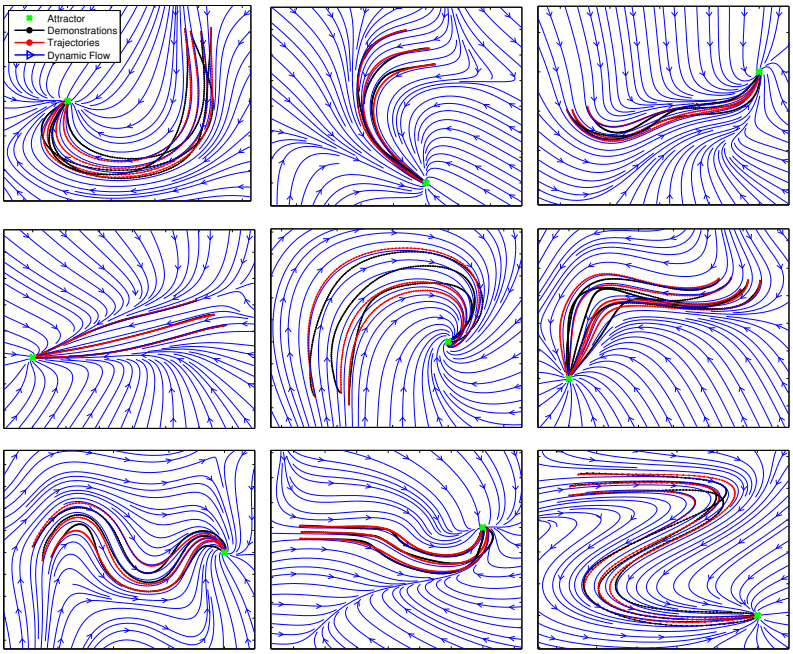

Fig. 3: The figure shows 9 examples from the benchmark data set [8]. The black lines represent the training data and the red lines are the reproduced movements. The learned vector field is depicted in blue.

Acknowledgement: This research is funded by FP7 under GA. No. 248311-AMARSi and the German BMBF within the Intelligent Technical Systems Leading-Edge Cluster

References

[1] F. A. Mussa-Ivaldi. From basis functions to basis fields: vector field approximation from sparse data. *Biological Cybernetics*, 67:479–489, 1992.

[2] Y. Kuroe and H. Kawakami. Vector field approximation by model inclusive learning of neural networks. *Proc. ICANN*, 4668:717–726, 2007.

[3] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[4] F. Reinhart, A. Lemme, and J. Steil. Representation and generalization of bi-manual skills from kinesthetic teaching. *Proc. Humanoids*, pages 560–567, 2012.

[5] Z. Huang and C. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[6] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse. Op-elm: Optimally pruned extreme learning machine. *IEEE Trans. on Neural Networks*, 21(1):158–162, 2010.

[7] K. Neumann and J. Steil. Reliable integration of continuous constraints into extreme learning machines. *Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, In press.

[8] S. Khansari-Zadeh. `http://www.amarsi-project.eu/open-source` [Last ac. 2012-12-05].

[9] M. S. Bazaraaa, H. Sherali, and C. Shetty. *Nonlinear programming: Theory and Algorithms*. Ed. John Wiley & Sons, 2006.