# Neurally imprinted vector fields with data-driven learned Lyapunov function

Andre Lemme[1], Klaus Neumann[1] and Jochen J. Steil[1]

Nonlinear dynamical systems are a promising representation to learn complex robot movements. Besides their undoubted modeling power, it is of major importance that such systems work in a stable manner. We use a neural learning scheme that estimates stable dynamical systems from demonstrations based on a two-stage process: first, a data-driven Lyapunov function candidate is estimated. Second, stability is incorporated by means of a novel method to respect local constraints in the neural learning.

We consider trajectory data that are driven by time independent vector fields: $\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}) \;, \mathbf{x} \in \Omega$ , where a state variable $\mathbf{x}(t) \in \Omega \subseteq \mathbb{R}^d$ at time $t \in \mathbb{R}$ defines a state trajectory. It is assumed that the vector field $\mathbf{v} : \Omega \to \Omega$ is nonlinear, continuous, and continuously differentiable with a single asymptotically stable point attractor $\mathbf{x}^*$ with $\mathbf{v}(\mathbf{x}^*) = 0$ in $\Omega$. The limit of each trajectory in $\Omega$ thus satisfies: $\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}^* : \forall \mathbf{x}(0) \in \Omega$ .

The key question is how to learn $\mathbf{v}$ as a function of $\mathbf{x}$ by using demonstrations for training and ensure its asymptotic stability at target $\mathbf{u}^*$ in $\Omega$. The estimate is denoted by $\hat{\mathbf{v}}$ in the following. The evolution of motion can then be computed by numerical integration of $\dot{\mathbf{x}} = \hat{\mathbf{v}}(\mathbf{x})$, where $\mathbf{x}(0) \in \Omega$ denotes the starting point of the motion.

Consider a single hidden layer feed-forward neural network for the estimation of $\mathbf{v}$: $\mathbf{x} \in \mathbb{R}^d$ denotes the input, $\mathbf{h} \in \mathbb{R}^R$ the hidden, and $\hat{\mathbf{v}} \in \mathbb{R}^d$ the output neurons. The input is connected to the hidden layer by the input matrix $W^{\mathbf{inp}} \in \mathbb{R}^{R\times d}$. The read-out matrix is given by $W^{\mathbf{out}} \in \mathbb{R}^{d\times R}$. For input $\mathbf{x}$, the output of the $i$th neuron is thus given by:

$$\hat{v}_i(\mathbf{x}) = \sum_{j=1}^{R} W^{\mathbf{out}}_{ij} f(\sum_{n=1}^{d} W^{\mathbf{inp}}_{jn} x_n + b_j) \;, \tag{1}$$

where $i = 1,\ldots,d$ and $b_j$ is the bias for neuron $j$. $f(x) = 1/(1+\exp(-x))$ denotes the activation function applied to each neuron in the hidden layer. The components of the input matrix and the biases are drawn from a random distribution and remain fixed after initialization. This network architecture is called extreme learning machine (ELM) [1].

Let $D = (\mathbf{x}(k), \mathbf{v}(k)) : k = 1\ldots N_{\mathrm{tr}}$ be the data set for training where $N_{\mathrm{tr}}$ is the number of samples in $D$. Supervised learning for ELMs is restricted to the read-out weights $W^{\mathbf{out}}$ and is usually done by ridge regression in a computationally cheap fashion:

$$W^{\mathbf{out}} = \left(HH^T + \varepsilon I\right)^{-1} HV^T \;, \tag{2}$$

where $H = (\mathbf{h}(\mathbf{x}(1)),\ldots,\mathbf{h}(\mathbf{x}(N_{\mathrm{tr}}))) : k = 1\ldots N_{\mathrm{tr}}$ is a matrix harvesting the hidden states for each input $\mathbf{x}(k)$ in the training data set, $I$ is the identity matrix, and $\varepsilon > 0$ is the regularization parameter. The plain ELM learning scheme it not well suited for learning from trajectories because it will lead to unstable dynamical systems, but we show that this network architecture is particularly well suited for incorporation of stability constraints in learning.

In order to stabilize the dynamical systems estimate $\hat{\mathbf{v}}$, we recall the conditions for asymptotic stability of arbitrary dynamical systems found by Lyapunov: a dynamical system is asymptotically stable at fixed-point $\mathbf{x}^* \in \Omega$ in the compact and positive invariant region $\Omega \subset \mathbb{R}^d$ if there exists a continuous and continuously differentiable function $L : \Omega \to \mathbb{R}$ which satisfies the following conditions:

$$\begin{array}{llll} (\mathbf{i}) & L(\mathbf{x}^*) = 0 & (\mathbf{ii}) & L(\mathbf{x}) > 0 : \forall \mathbf{x} \in \Omega, \mathbf{x} \neq \mathbf{x}^* \\ (\mathbf{iii}) & \dot{L}(\mathbf{x}^*) = 0 & (\mathbf{iv}) & \dot{L}(\mathbf{x}) < 0 : \forall \mathbf{x} \in \Omega, \mathbf{x} \neq \mathbf{x}^* \;. \end{array} \tag{3}$$

We assume that we have a function $L$ that satisfies condition (**i**)-(**iii**) and call it: *Lyapunov candidate*. Such a function can be used to obtain a learning algorithm that also satisfies condition (**iv**) w.r.t. the estimated dynamics $\hat{\mathbf{v}}$. We therefore re-write condition (**iv**) by using the ELM learner defined in Eq. (1):

$$\dot{L}(\mathbf{x}) = \sum_{i=1}^{d} (\nabla_{\mathbf{x}} L(\mathbf{x}))_i \cdot \sum_{k=1}^{R} W^{\mathbf{out}}_{ij} \cdot f(\sum_{k=1}^{d} W^{\mathbf{inp}}_{jk} x_k + b_j) < 0 \;. \tag{4}$$

Note that $\dot{L}$ is linear in the output parameters $W^{\mathbf{out}}$ irrespective of the form of the Lyapunov function $L$. Eq. (4) defines a linear inequality constraint $\dot{L}(\mathbf{u}) < 0$ on the read-out parameters $W^{\mathbf{out}}$ for a given point $\mathbf{u} \in \Omega$ which can be implemented by quadratic programming [2], [3]. The training of the read-out weights $W^{\mathbf{out}}$ given by Eq. (2) is rephrased as a quadratic program subject to constraints given by $L$:

$$\begin{gathered} W^{\mathbf{out}} = \arg\min_{W} (\|W \cdot H(X) - V\|^2 + \varepsilon_{\mathrm{RR}} \|W\|^2) \\ \text{subject to: } \dot{L}(U) < 0 \;, \end{gathered} \tag{5}$$

where the matrix $V$ contains the corresponding target velocities of the demonstrations and $U = \{\mathbf{u}(1),\ldots,\mathbf{u}(N_{\mathrm{s}})\}$ is the matrix collecting the discrete samples.

In the following we introduce a sampling strategy in order to minimize the number of samples needed for generalization of the local constraints towards the continuous region.

*This work was not supported by any organization

[1]The authors are with the Research Institute for Cognition and Robotics (CoR-Lab), Faculty of Technology, Bielefeld University. Universitätsstr. 25, 33615 Bielefeld, Germany. kneumann,alemme,jsteil)@cor-lab.de

The data set $D$ for training and the region $\mathscr{S}$ where the constraints are supposed to be implemented are assumed to be given. As a first step ($k = 0$), the network is initialized randomly and trained without any constraints (i.e. the sample matrix $U^k = U^0 = \emptyset$ is empty). In this case learning can be accomplished by ridge regression - the standard learning scheme for ELMs, see Eq. (2). In the next step, $N_\mathrm{C}$ samples $\hat{U} = \{\hat{\mathbf{u}}^1, \hat{\mathbf{u}}^2, \ldots, \hat{\mathbf{u}}^{N_\mathrm{C}}\}$ are randomly drawn from a uniform distribution in $\Omega$. Afterwards, the number of samples $\nu$ fulfilling (**iv**) of Lyapunov's conditions of asymptotic stability is determined. The sampling algorithm stops if more then $p$ percent of this samples fulfill the constraint. Otherwise, the most violating sample $\hat{\mathbf{u}}$ of condition (**iv**) is added to the sample pool: $U^{k+1} = U^k \cup \hat{\mathbf{u}}$. The obtained set of samples is then used for training. A pseudo code of the learning procedure is provided in Alg. 1.

**Algorithm 1** Sampling Strategy

```
Require: data set $D$, region $\mathscr{S}$, counter $k = 0$, sample pool
  $U^k = \emptyset$, and ELM $\hat{\mathbf{v}}$ trained with $D$
  repeat
    draw samples $\hat{U} = \{\hat{\mathbf{u}}^1, \hat{\mathbf{u}}^2, \ldots, \hat{\mathbf{u}}^{N_\mathrm{C}}\}$
    $\nu$ = no. of samples in $\hat{U}$ fulfilling (iv)
    $U^{k+1} = U^k \cup \arg\max_{\mathbf{u} \in \hat{U}} \dot{L}(\mathbf{u})$
    train ELM with $D$ and $U^{k+1}$
  until $p > \frac{\nu}{N_\mathrm{C}}$
```

The application of standard candidates, i.e. quadratic Lyapunov candidates visualized in Fig. 1 (left) is limited if the tasks demands an appropriate complexity. In theory, much more complex functions are possible and also desired, see Fig. 1 (right), but there is no constructive or analytic way to derive such a candidate function directly therefore we utilize the methodology introduced in [4] to approximate the Lyapunov candidate by another extreme learning machine.

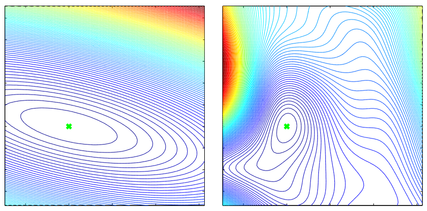

Fig. 1. Level sets of two different Lyapunov candidates $L = x^T P x$ and $L = ELM$.

In order to prevent a violation of the training data $D$ by a given Lyapunov candidate $L$ we use the following functional which defines the measure of violation:

$$M(L) = \frac{1}{N_\mathrm{ds}} \sum_{i=1}^{N_\mathrm{traj}} \sum_{k=1}^{N^i} \Theta \left[ (\nabla L(\mathbf{x}^i(k)))^T \cdot \mathbf{v}^i(k) \right] \quad , \qquad (6)$$

only those samples $(\mathbf{x}^i(k), \mathbf{v}^i(k))$ are counted in $M$ where the scalar product between $\nabla L(\mathbf{x}^i(k))$ and $\mathbf{v}^i(k)$ is positive and thus violating Lyapunov's condition (**iv**).

The proposed method to generate a Lyapunov candidate is to use an ELM architecture which defines a scalar function $L_\mathrm{ELM} : \mathbb{R}^d \to \mathbb{R}$. Note, that this ELM also contains a three-layered and random projection structure with only one output neuron and read-out matrix $W^\mathbf{out} \in \mathbb{R}^R$. The main goal is to minimize the violation of the training data measured by Eq. (6) by making the negative gradient of this function follow the training data closely.

For more details on how to approximate the Lyapunove candidates please see [4], [5].

## ACKNOWLEDGMENT

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] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[2] P. Gill, W. Murray, and M. Wright, *Practical Optimization*. Academic Press, 1981.

[3] T. F. Coleman and Y. Li, "A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables," *SIAM Journal on Optimization*, vol. 6, no. 4, pp. 1040–1058, 1996.

[4] K. Neumann, A. Lemme, and J. J. Steil, "Neural learning of stable dynamical systems based on data-driven lyapunov candidates," in *Int. Conf. Intelligent Robots and Systems (IROS)*, IEEE. Tokio: IEEE, In Press.

[5] A. Lemme, K. Neumann, F. R. Reinhart, and J. J. Steil, "Neurally imprinted stable vector fields." Bruges: d-facto, 2013, best paper award, pp. 327 – 332.