RELIABLE INTEGRATION OF CONTINUOUS CONSTRAINTS INTO EXTREME LEARNING MACHINES

KLAUS NEUMANN*

Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University
Universitätsstraße 25, 33615 Bielefeld, Germany

kneumann@cor-lab.de

MATTHIAS ROLF

Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University
Universitätsstraße 25, 33615 Bielefeld, Germany

mrolf@cor-lab.de

JOCHEN JAKOB STEIL

Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University
Universitätsstraße 25, 33615 Bielefeld, Germany

jsteil@cor-lab.de

The application of machine learning methods in the engineering of intelligent technical systems often requires the integration of continuous constraints like positivity, monotonicity, or bounded curvature in the learned function to guarantee a reliable performance. We show that the extreme learning machine is particularly well suited for this task. Constraints involving arbitrary derivatives of the learned function are effectively implemented through quadratic optimization because the learned function is linear in its parameters, and derivatives can be derived analytically. We further provide a constructive approach to verify that discretely sampled constraints are generalized to continuous regions and show how local violations of the constraint can be rectified by iterative relearning. We demonstrate the approach on a practical and challenging control problem from robotics, illustrating also how the proposed method enables learning from few data samples if additional prior knowledge about the problem is available.

Keywords: extreme learning machine, neural network, prior knowledge, continuous constraints, regression.

1. Introduction

In recent years, machine learning has matured to a point where the application of learning methods in the engineering of intelligent technical systems increasingly comes into focus of applied research. A prominent example is the german leading edge cluster "Intelligent Technical Systems", which implements a dedicated project

*corresponding author

to apply self-optimization and learning methods in technical production systems and the intelligent mechatronical devices of tomorrow.

In this application domain, machine learning solutions face the challenge to guarantee certain properties of the learned functions to assure safe and reliable operation of the technical system while capitalizing on the full power of data-driven modeling methods. In automatic control, for example, constraints like maximum energy of the control signal, monotonicity or smoothness are often required to ensure safe operation of the plant [1]. This is prior knowledge in the sense that certain desired relations between inputs and outputs are known in advance, i.e. constraints must hold on the learned function [2]. The acceptance of learning algorithms in such domains depends on a reliable integration of these constraints into the learner, where "reliable" can refer to a high probability, or even a mathematical proof, of success. This task is not trivial because many machine learning algorithms gain their very power from the universal approximation capabilities. In real applications data are typically noisy, outliers occur and there is always a risk of overfitting corrupted data. A constraint that is present in the ideal function, from which the training data are sampled, may therefore be violated by the learned function.

Approaches that use a-priori model selection can guarantee certain properties or limits of the model complexity. Yet, model selection strategies are entirely specific for these constraints and can not easily be found for complex constraints. The issue of incorporation becomes even more complex due to the large variety of such constraints. Essentially, the reliable integration of prior knowledge in form of constraints can be separated into two main aspects: (i) the integration of the constraint into the learning algorithm and (ii) the mathematical verification that the constraints hold in the learned function. Both (i) and (ii) are connected if the learning algorithm directly guarantees the constraint. However, we can also first tackle (i) separately and then ex-post verify that the outcome is the desired one. In this paper, we follow the latter approach.

The incorporation of specific kinds of prior knowledge was tackled before in several machine learning models. Many of those deal with the embedding of very specific kinds of prior knowledge like monotonicity [3,4] or minimum/maximum output control [1]. Also the prior knowledge integration into ELMs was focused earlier in [5] for only a limited number of cases. More general approach are offered by [6,2,8] but without focus on the ensured implementation of constraints.

This contribution shows that the particular form of the extreme learning machine (Huang et al. [9,10,25], see [23] for recent theoretical results) allows for an efficient and flexible incorporation of continuous constraints in the learned function. To this aim, output learning is organized as solving a quadratic constraint optimization problem, i.e. learning refers to minimization of the standard square error under additional linear constraints. It turns out that all constraints which can be expressed as linear inequalities involving arbitrary derivatives of the learned functions can be incorporated. This is possible because partial differentiation of a function which is implemented by an ELM, can directly be performed due to the special form of the

ELM with its fixed input weights. The key idea then is to iterate steps (i)-learning and (ii)-verification. The first step is to learn a function based on a training data set and linear inequalities for the constraints and then (ii) efficiently verify, where constraints are still violated by the learned function within continuous regions of the input space. Then a smart sampling strategy is applied to incrementally add more inequalities for the constraints and (i) is repeated, starting from the previous solution as initial point.

Many different mechanisms were developed in order to improve ELMs without focus on prior knowledge. An interesting idea to improve ELMs is to decrease the size of the hidden layer - the optimally pruned extreme learning machine (OPELM) [21]. The learning results are improved by pruning the OPELM using a leave-one-out criterion and a ranking of the neurons. [22], e.g., describes ensemble strategies for ELMs based on entropy measures. Such methods are complementary to the idea of incorporating prior knowledge and can therefore be used in parallel in order to optimize the network architecture and the encoding in the hidden layer.

2. Embedding Prior Knowledge via Sampling into ELMs

The classical extreme learning machine (ELM) approach [9] consists of three layers: $\mathbf{x} \in \mathbb{R}^I$ collects the input, $\mathbf{h} \in \mathbb{R}^R$ the hidden, and $\mathbf{y} \in \mathbb{R}^O$ the output neurons. The input is connected to the hidden layer through the input matrix $W^{\mathbf{inp}} \in \mathbb{R}^{R\times I}$, while the read-out is given by $W^{\mathbf{out}} \in \mathbb{R}^{O\times R}$. The calculation for the ith output neuron for input $\mathbf{x}$ reduces to:

$$y_i(\mathbf{x}) = \sum_j W^{\mathbf{out}}_{ij} f(a_j \sum_k W^{\mathbf{inp}}_{jk} x_k + b_j) = \sum_j W^{\mathbf{out}}_{ij} h_j(\mathbf{x}) \ , \qquad (1)$$

where a_j, b_j are slope and bias of the jth neuron parameterizing the component-wise applied Fermi-function $f(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$. In the experiments, we optimize the choice of the slopes a and the biases b through pretraining by batch intrinsic plasticity (BIP) [11] after random initialization. The learning algorithm is fed by a data set $D = (X, T) = (\mathbf{x}^i, \mathbf{t}^i)_{i=1\dots N_{\mathrm{tr}}}$ consisting of N_{tr} training samples. The hidden layer states obtained for inputs X are harvested in the matrix $H(X) = (\mathbf{h}(\mathbf{x}^1), \dots, \mathbf{h}(\mathbf{x}^{N_{\mathrm{tr}}}))$ and the corresponding targets in the matrix T, respectively.

Many constraints on a desired function can be expressed by bounding on the outputs $\mathbf{y}(\cdot)$ (see Eq. (1)) or its partial derivatives. Hence, we generally refer to a constraint $L(W^{\mathbf{out}}, \mathbf{u})$ for input $\mathbf{u}$ w.r.t. the read-out parameters $W^{\mathbf{out}}$ as a linear combination of partial derivatives with parameters γ_i and bound c of the form:

$$L(W^{\mathbf{out}}, \mathbf{u}) = \sum_i \gamma_i D^{\mathbf{m}^i} y_i(\mathbf{u}) - c = \sum_i \gamma_i W^{\mathbf{out}}_i \cdot D^{\mathbf{m}^i} \mathbf{h}(\mathbf{u}) - c. \qquad (2)$$

$D^{\mathbf{m}^i} = \partial^M / \partial u_{m^i_1} \dots u_{m^i_M}$ is the component-wise differential operator, whereas the vector $\mathbf{m}^i = (m^i_1 \dots m^i_M) \in [1, I]^M$ defines the input dimensions with regard to which the differentiations are carried out. Interestingly, Eq. (2) shows that the constraint can be rephrased in terms of the hidden state which is linear in the learning parameters $W^{\mathbf{out}}$ defining a linear inequality for each discrete sample $\mathbf{u}$.

If a sampling set $U = (\mathbf{u}^1, \dots, \mathbf{u}^{N_\mathrm{s}})$ comprising of N_s discrete inputs is given, incorporation of these constraints into the function learned by the ELM is phrased as solving a quadratic programming [12,13] optimizing the read-out weights $W^{\mathbf{out}}$ subject to a sytem of linear inequalities $L(W^{\mathbf{out}}, U) \leq 0$:

$$W^{\mathbf{out}} = \arg\min_{W}(\|W \cdot H(X) - T\|^2 + \alpha\|W\|^2), \tag{3}$$

$$\text{subject to: } L(W, U) \leq 0. \tag{4}$$

However, to make this approach practical also the multi-dimensional differentiation has to be carried out. Fortunately, the special form of the ELM allows to compute arbitrary analytical partial derivatives of the different output components analytically as:

$$\begin{aligned}\frac{\partial^M y_i(\mathbf{u})}{\partial u_{m_1} \dots u_{m_M}} &= \sum_j W_{ij}^{\mathbf{out}} \frac{\partial^M h_j(\mathbf{u})}{\partial u_{m_1} \dots u_{m_M}} \\ &= \sum_j W_{ij}^{\mathbf{out}} f^{(M)}(a_j \sum_k W_{jk}^{\mathbf{inp}} u_k + b_j) \cdot a_j^M W_{jm_1}^{\mathbf{inp}} \dots W_{jm_M}^{\mathbf{inp}}, \end{aligned} \tag{5}$$

where $f^{(M)}$ denotes the Mth derivative of f. Note, that the output of the network can be interpreted as 0th derivative when choosing $M = 0$. Solving the quadratic program now guarantees satisfaction of the given constraints w.r.t. the discrete inputs $\mathbf{u}$, which is already useful in many applications.

3. From Discrete to Continuous Constraints

The next step is to target constraints in a continuous compact region $\mathcal{S}$ of the input space, i.e. to generalize the point-wise constraints to a continuous region, where discrete inputs $\mathbf{u}^i \in \mathcal{S}$ are regarded as discrete samples of the continuous constraint.

In principle, no finite number of discrete samples $\mathbf{u}^i$ can implement the constraint and at the same time guarantee its generalization to hold in the continuous region without additional verification effort. In fact, the quadratic program only guarantees the satisfaction of the continuous constraint in the points $\mathbf{u}^i$. But we expect that the generalization ability and the implicit smoothness of the used network will enable an implementation of the constraint in the whole region $\mathcal{S}$ with only a limited number of discrete samples. We can therefore expect that sampling is sufficient for generalization, but then need to verify ex-post that the constraint $L(\mathbf{u}) = L(W^{\mathbf{out}}, \mathbf{u}) \leq 0$ holds for all $\mathbf{u} \in \mathcal{S}$ (the read-out matrix is omitted in the following sections for notational simplicity).

It is impossible to verify the reliability on a given constraint in closed form, because the universal approximation capability of the ELM [10] implies that the learned function can in principle be arbitrarily complex. We therefore provide an algorithm to verify the fulfillment of the constraint which is based on a worst case analysis of the Taylor approximation of the learned function.

The algorithm's result $P(L, \mathbf{y}, \mathcal{S}, \varepsilon)$ w.r.t. the constraint L, reliability ε, and the ELM's output $\mathbf{y}$ is *true* for region $\mathcal{S}$ if and only if the constraint L is satisfied by the function $\mathbf{y}(\cdot)$ in region $\mathcal{S}$ with reliability ε.

Reliability Verification. As a first step, a second order Taylor approximation of the constraint L in the input region $\mathcal{S}$ and its corresponding remainder are calculated. The Taylor approximation in $\mathbf{u}^0 \in \mathcal{S}$ is given as:

$$\begin{aligned} L(\mathbf{u}) &= T(\mathbf{u}, \mathbf{u}^0) + rem(\mathbf{u}, \mathbf{u}^0) && (6)\\ &= C + J^T(\mathbf{u} - \mathbf{u}^0) + \frac{1}{2}(\mathbf{u} - \mathbf{u}^0)^T H(\mathbf{u} - \mathbf{u}^0) + rem(\mathbf{u}, \mathbf{u}^0) \ , && (7) \end{aligned}$$

where $C = L(\mathbf{u}^0)$ is the constant term, $J = \nabla L(\mathbf{u})\big|_{\mathbf{u}^0}$ is the Jacobian, $H = (\nabla\nabla^T)L(\mathbf{u})\big|_{\mathbf{u}^0}$ is the Hessian matrix evaluated at point $\mathbf{u}^0$, and $rem(\mathbf{u}, \mathbf{u}^0)$ is the remainder term. We compute an upper bound for the remainder:

$$rem := \sum_i \gamma_i \sum_j W_{ij}^{\mathbf{out}} M_j \frac{|\underline{u}_j - \overline{u}_j|^3}{6} \ , \tag{8}$$

where i is the dimension of the constrained output and γ_i are the coefficients defined as in Eq. (2). The used variables are:

$$\underline{u}_j = \min_{\mathbf{u}\in\mathcal{S}} W_j^{\mathbf{inp}}\mathbf{u} \ , \ \overline{u}_j = \max_{\mathbf{u}\in\mathcal{S}} W_j^{\mathbf{inp}}\mathbf{u} \ , \text{ and} \tag{9}$$

$$M_j = \max_{u_j \in [\underline{u}_j, \overline{u}_j]} \left[f^{(M+3)}(a_j u_j + b_j) \cdot a_j^{M+3} W_{jm_1}^{\mathbf{inp}} \dots W_{jm_M}^{\mathbf{inp}} \right] \ . \tag{10}$$

The maximization steps in (9) can be performed by evaluating the vertices of $\mathcal{S}$ if the region $\mathcal{S}$ is a convex polyhedron, e.g. a regular hypercube. The maximization in Eq. (10) uses that f can be differentiated analytically if we use the Fermi-function which is one-dimensional w.r.t. its argument - which is only feasible due to the simple and elegant form of the ELM. The remainder of the Taylor approximation $rem(\mathbf{u}, \mathbf{u}^0)$ in Eq. (7) is bounded from above by rem in Eq. (8) in region $\mathcal{S}$:

$$rem \geq rem(\mathbf{u}, \mathbf{u}^0) : \forall \mathbf{u} \in \mathcal{S}. \tag{11}$$

Since T (see Eq. (7)) is a polynomial of second order, it is possible to find its global maximum and minimum in $\mathcal{S}$ analytically. In order to find a worst case approximation of the learned function, the following is tested and a recursive step is performed:

$$P(\mathcal{S}) = \begin{cases} 1 & : \text{if } \max\limits_{\mathbf{u}\in\mathcal{S}} \left[T(\mathbf{u}, \mathbf{u}^0)\right] < \varepsilon - rem \\ 0 & : \text{if } \min\limits_{\mathbf{u}\in\mathcal{S}} \left[T(\mathbf{u}, \mathbf{u}^0)\right] > \varepsilon + rem \\ \bigwedge_i P(\mathcal{S}_i) & : \text{otherwise} \end{cases} \tag{12}$$

where $\mathbf{u}^0 \in \mathcal{S}$ is the base of the Taylor approximation and $\mathcal{S} = \bigcup_i \mathcal{S}_i$ is divided into the pairwise disjoint subregions $\mathcal{S}_i$.

A schematic view of the decision process in Eq. (12) is given in Fig. 1. The figure shows a one-dimensional mapping with constraint $L \leq \varepsilon$ and three subregions ($\mathcal{S}_{i-1}$, $\mathcal{S}_i$, and $\mathcal{S}_{i+1}$) where the reliability is tested. The center of each region $\mathbf{u}^0$ defines the respective Taylor polynomial and the remainder estimate *rem* of L. The regions show the possible cases of Eq. (12): (i) the maximum of the Taylor polynomial $\max\left[T(\mathbf{u},\mathbf{u}^0)\right]$ calculated for $\mathbf{u} \in \mathcal{S}_{i-1}$ is below $\varepsilon - rem$ and therefore $P = 1$ - the constraint is fulfilled; (ii) the minimum of the Taylor polynomial $\max\left[T(\mathbf{u},\mathbf{u}^0)\right]$ in region $\mathcal{S}_{i+1}$ is above $\varepsilon + rem$ and therefore $P = 0$ - there exists at least one point in this region where the constraint is violated; and (iii) it is not clear whether the constraint in region $\mathcal{S}_i$ is satisfied or not - a division into smaller subregions is necessary.

An intrinsic feature of Taylor approximations is that the quality of the estimated approximation is the best close to the approximation point $\mathbf{u}^0$ (locality feature): the smaller the region $\mathcal{S}$, the better the estimation. This separation is performed until convergence of the algorithm, where we can stop early whenever the constraint is

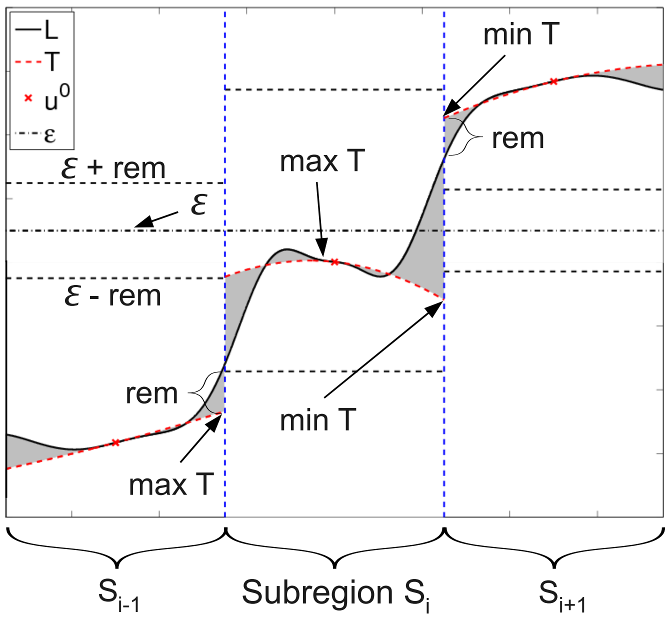

Figure 1. One-dimensional illustration of the reliability verification of a constraint L with a Taylor polynomial of second order in a subregion $\mathcal{S}_i$.

definitely not fulfilled in some subregion. For $\varepsilon > 0$ we ε-fulfill the constraint in the sense that a violation of size ε is tolerated, very similar to the ε-sensitive loss functions frequently employed in support vector regression. In engineering applications, we also might set $\varepsilon < 0$ to guarantee some safety margin, e.g. to balance numerical errors.

4. Iterative Sampling and Re-Learning

This section demonstrates how the interplay between learning and verification works. A synthetic regression task is used as an illustrative example where the maximal output of an $\mathbb{R}^2 \to \mathbb{R}$ map is supposed to be bounded despite that the training samples are violating the constraint.

The Learning Procedure. As a first step ($k = 0$), the network is initialized randomly and trained without any constraints (the sample matrix U^k is empty in the beginning): $U^0 = \emptyset$ - in this case learning can be done in computationally cheap fashion by the use of ridge regression (RR). The region $\mathcal{S}$ is initialized without separation into subregions: $\{\mathcal{S}_0\} = \{\mathcal{S}\}$

In the iterative step k, a uniform rejection sampling in the indifferent/violating regions $\mathcal{S}_i$ ($P(\mathcal{S}_i) \neq 1$) is performed N_{re} times. Samples violating the constraint are collected: $\hat{U} = \{\hat{\mathbf{u}} \in \{\mathcal{S}_i\} : L(\hat{\mathbf{u}}) > 0\}$. The rejection sampling stops if N_{C} samples $\hat{\mathbf{u}}$ are found or the number of trials N_{re} is exceeded. If less than N_{C} samples $\hat{U}$ were found in $\{\mathcal{S}_i\}$, the reliability is verified by algorithm P for one further separation of the regions $\{\mathcal{S}_i\}$.

The sample set U^k in step k is merged with the set $\hat{U}$ to constitute the new sample set U^{k+1} for step $k+1$: $U^{k+1} = U^k \cup \hat{U}$. The obtained set of samples is then used for training of the ELM. The sampling algorithm stops if $P(\mathcal{S}) = 1$ or performs a rejection sampling on the subregions $\mathcal{S}_i^k$ where $P(\mathcal{S}_i^k) = 0$ then starting with iteration step $k+1$. A pseudo code of the learning procedure is provided by Alg. 1.

Furthermore, it is shown in the following section that this sampling strategy is highly effective - only few samples are needed to incorporate the constraint reliably in comparison to random sampling.

Illustrative Example. An ELM with $R = 100$ hidden layer neurons is supposed to learn an $\mathbb{R}^2 \to \mathbb{R}$ map of four two-dimensional Gaussian distributions. The centers of the single distributions are $\mu_{1\ldots4} = (\pm 0.5, \pm 0.5)^T$ and the respective variance of the distributions are $\sigma_{1\ldots4} = \frac{1}{5}$. $N_{\text{tr}} = 500$ samples are used for training, $N_{\text{te}} = 500$ for testing, and $N_{\text{re}} = 10^4$ for rejection sampling. Also Gaussian noise with an amplitude of 0.1 is added to the training data. The network was trained by BIP with $\mu_{\text{BIP}} = 0.2$ beforehand (according to [11]). The regularization parameter $\alpha = 10^{-4}$ was obtained by line search. Simultaneously, an artificial constraint is supposed to be satisfied by the network after learning: the maximal output of the network is restricted to $y(\mathbf{x}) \leq 0.4 = c$ for all $\mathbf{x} \in \mathcal{S} = [-1, 1]^2$, where c is the

Algorithm 1: Constraint Learning with Sampling Strategy

```
initialize ELM randomly
require ELM $\mathbf{y}(\cdot)$, data set $D$, constraint $L$, region $\mathcal{S}$, reliability $\varepsilon$, sample
increment $N_{\mathrm{C}}$, max. rejection trials $N_{\mathrm{re}}$, and learning parameters $\alpha$ and $\mu$
train ELM with BIP and then with RR: $\mu, \alpha \rightarrow$ new ELM $\mathbf{y}(\cdot)$
set $k = 0$, $U^k = \emptyset$, $\{\mathcal{S}^0\} = \{\mathcal{S}\}$
repeat
  repeat
    build $\hat{U} = \{\hat{\mathbf{u}} \in \{\mathcal{S}_i\} : L(\hat{\mathbf{u}}) > 0 \ \& \ P(\mathcal{S}_i) \neq 1\}$ with $N_{\mathrm{C}}$ samples ($N_{\mathrm{re}}$ trials)
    if rejection sampling found $N_{\mathrm{C}}$ samples
      merge sample matrices: $U^{k+1} = U^k \cup \hat{U}$
      learning with quadratic program: $D, U^{k+1}, \alpha \rightarrow$ new ELM $\mathbf{y}(\cdot)$
      increase $k \leftarrow k + 1$
    else
      break
    end if
  end repeat
  collect $p = P(L, y, \mathcal{S}, \varepsilon)$ and subregions $\{\mathcal{S}_i\}$ where $P(\mathcal{S}_i) \neq 1$
until p is true
return $\mathbf{y}(\cdot)$
```

output bound. Note, that the output of the network is interpreted as 0th derivative according to Eq. (5) and Eq. (2) and is thus consistent with the proposed framework when choosing $M = 0$. In each step, the algorithm adds $N_{\mathrm{C}} = 3$ constraints to the learning until no violation can be assessed anymore. The reliability for the discrete constraints is set to $\varepsilon = 0.01$. A cell is divided along its longest side into two equal parts, if the reliability was not verified in the respective step. This defines the structure of the subregions.

Fig. 2 summarizes the results of the task. The first row shows the outputs of the network in the first, fourth and the last (12th) iteration of the learning, while the second row visualizes the corresponding outcome of the reliability verification. In the plots of the second row: green boxes show regions where the constraint is verified, red boxes where the constraint is violated and crosses mark samples in U obtained by the iteration process. Obviously, the constraint is violated before the iteration starts - see Fig. 2 (left column). After some iterations of the algorithm, the maximal output of the network in $\mathcal{S}$ shrinks due to the addition of samples of the constraint. The right hand side of the figure demonstrates that the constraint is fulfilled after twelve iterations, e.g. on basis of 36 discrete $\mathbf{u}_i$, sampling the constraint. The error for the respective iteration-steps are: $E^1_{\mathrm{tr}} = 0.087$, $E^1_{\mathrm{te}} = 0.090$, $E^4_{\mathrm{tr}} = 0.134$, $E^4_{\mathrm{te}} = 0.136$, $E^{12}_{\mathrm{tr}} = 0.160$ and $E^{12}_{\mathrm{te}} = 0.160$. The trade-off between the satisfaction of the constraint and the learning of the data set clearly shows-up in the errors which become bigger in each step of the iteration. Note that the samples are only placed where the mapping produces high output values. Moreover, the cells produced by the recursive step in Eq. (12) are small in the region where the network's output y is close to

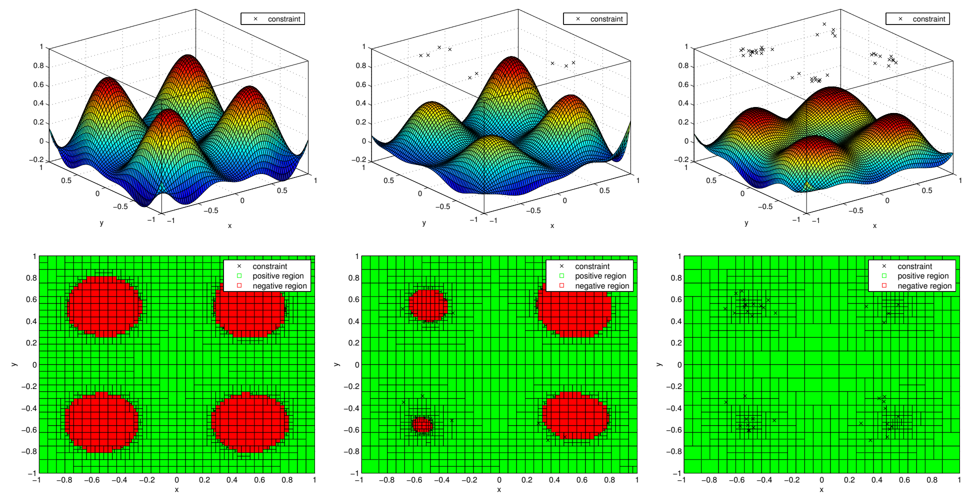

Figure 2. Iteration of the learning algorithm. Green boxes show regions where the constraint is verified, red boxes where the constraint is violated and crosses mark samples in U obtained by the iteration process. The maximal output bound is strongly violated by the unconstrained learner (left). The output of the network shrinks in each step of the iteration (center). The constraint is satisfied and proofed (right).

the upper bound c. This is due to the fact that the remainder rem becomes big in comparison to the difference of network output y and the output bound $c = 0.4$. Fig. 3 shows the results for two different sampling strategies. The plot visualizes the relative violation $\frac{\bar{C}}{N_S}$ by testing the fulfillment of the constraint at $N_S = 10^6$ randomly drawn and uniformly distributed discrete points in $\mathcal{S} = [-1, 1]^2$ - where $\bar{C}$ is the number of points in S where the constraint is not satisfied. It demonstrates that a sampling strategy as proposed in this contribution is superior to a random sampling.

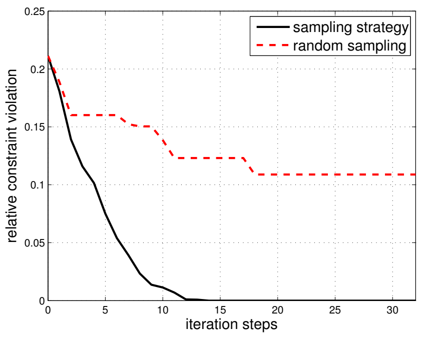

Figure 3. Proposed sampling strategy vs. random sampling.

5. Application in Automatic Control: Bionic Handling Assistant

Finally, a real world application from engineering is presented, i.e. the control of the bionic handling assistant (BHA) shown in Fig. 4. The BHA is a pneumatically actuated, award-winning [14] robotic platform [15,16] manufactured by Festo. The actuation principle with several segments of continuous parallel components has gathered increasing interest in robotics research over the last decade [17,18]. It belongs to a new class of soft and lightweight robots with pneumatic chambers which allows for a safe physical interaction between robots and humans. The downside of its biologically inspired design is that hardly any analytic models are available for control, which qualifies learning as an essential tool for its application. It is therefore a prototypical engineering application where learning is the method of choice, constraints are known from the physical plant, but data sampling is very expensive and must be done with great efficiency.

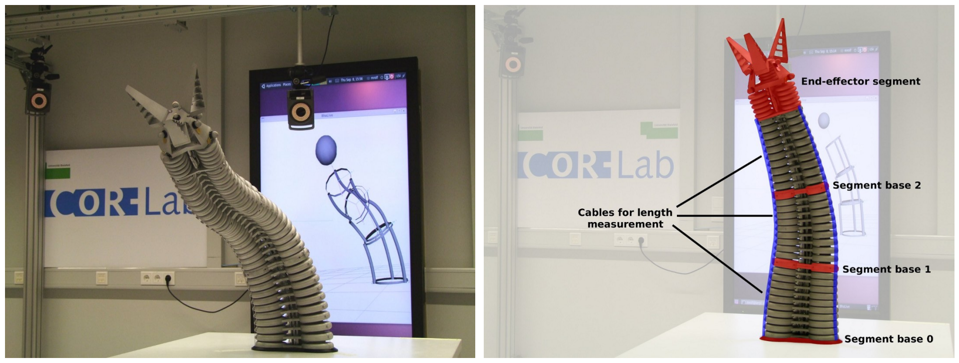

Figure 4. The bionic handling assistant BHA (left). The segments and respective length sensors of the BHA (right).

The most fundamental control capability necessary to use the BHA is to control the actuator lengths. The actuation itself operates with pneumatics, which is not sufficient for a reliable positioning of the robot: air pressure only describes a force, and friction with the addition of physical hysteresis effects cause different outcomes for the same pressure. In order to avoid this problem, the BHA has cable potentiometers (see Fig. 4 (right)) that allow to sense the outer length of the actuators providing geometric information about the robot shape. Controlling these lengths can, in principle, be done with standard schemes like proportional-integral-derivative-control (PID). The fundamental problem is that these approaches rely on quick and reliable feedback from the robot, while the BHA only provides very delayed and noisy feedback due to its pneumatic actuation. Consequently, the control would need to be applied with very low gains, which corresponds to only slow movements. At this point, learning can largely help to leverage the opportunities provided by the BHA. We improve the control of the actuator lengths by learning the *inverse mapping* from some desired actuator length to the pressure necessary to reach it in a mechanical equilibrium. These models allow a direct estimation of a reasonable control signal that can be applied on the robot very aggressively, without waiting for delayed feedback.

It is known in advance that the ground-truth behavior *per axis is strictly monotonous*, since higher pressure in one actuator physically leads to an extension of the actuator itself. Formulated as a constraint, it is necessary for the learner to reflect this behavior in order to be applicable in a closed control loop without leading to an amplification of errors.

The main difficulty for the learning is that training data are expensive and noisy. For each ground truth point of training data, some pressure needs to be applied on the real robot. This pressure must be active until the physical deformations of the robot have reached a mechanical equilibrium, which can take up to 20 seconds for a single data point. During this physical equilibration process, the visco-elastic properties of the elastic material result in slightly different equilibrium lengths every time. This process can be seen as highly inhomogeneous noise and requires multiple repetitions of the same experiment to obtain a reasonable basis for data driven learning. Furthermore, the high amount of noise sometimes causes the learner to violate the constraint of strictly monotonous behavior, if this prior knowledge is not explicitly incorporated in form of constraints. Note that the constraints, which can easily be expressed through the derivatives, also serve to keep the number of training samples small. Where no direct data are available, constraints quasi substitute actual data and can be generated artificially from prior knowledge. This is the key for faithful interpolation between sparse data points. This is desired because the samples recorded from the real robotic system are expensive in many respects.

Data Set. In order to learn the inverse models for length control we consider the three main segments (see Fig. 4 (right)) in isolation, each of which comprises three actuators, without a significant loss of accuracy. Hence, the goal for each segment is to learn a model that maps the desired length (meter) of the three actuators to the three necessary pressures (milli-bar).

For each segment, we explored the pressure space by applying pressures between minimum and maximum value in five equidistant steps. This results in a pressure grid comprising $5 \times 5 \times 5 = 125$ samples. For each pressure, the resulting combination of three lengths was recorded. In order to deal with the inherent variation due to the visco-elastic material we repeated this process five times with different traversal orderings, so that 625 samples per segment are available for learning. The minimum and maximum pressures, and the resulting length ranges are collected in Tab. 1. The grid for the applied pressures of segment one is illustrated in Fig. 5 (left). The corresponding length values recorded on the robot are shown in Fig. 5 (right). The data are clearly non-linear, with strong interactions of components and has huge gaps in the middle part of the target data, for which generalization is critical. All this makes the task evidently difficult to learn.

An appropriate error measure is to compute the per-axis average-deviation from

Table 1. Properties of the BHA data sets for the different segments.

Segment	Max. Press.	Min. Press.	Max. Len.	Min. Len.	# Samples	# Trials
1	800 mbar	0 mbar	0.32 m	0.16 m	625	5
2	1000 mbar	0 mbar	0.33 m	0.15 m	625	5
3	1200 mbar	0 mbar	0.29 m	0.16 m	625	5

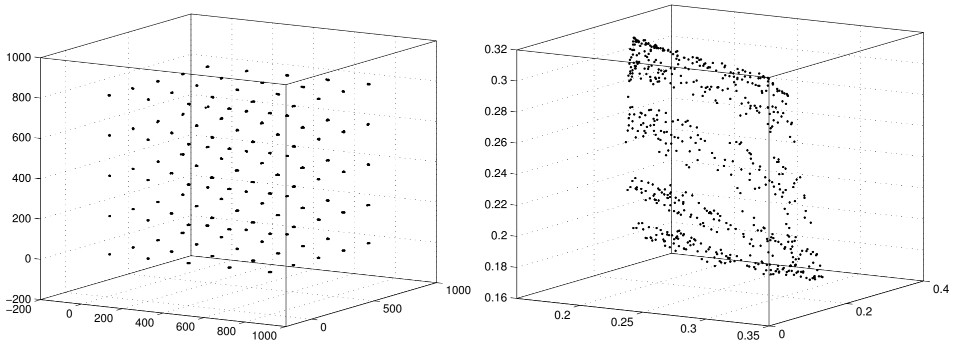

Figure 5. Data set for segment 1. Pressure grid with five samples per dimension (left) and the corresponding length values (right).

the measured ground truth value:

$$E = \frac{1}{N}\sum_i \frac{1}{D}\sum_d \|\text{pressure}_d(i) - y_d(\text{length}(i))\| \ , \tag{13}$$

where N is the number of samples in the evaluated data set and $D = 3$ is the input and output dimensionality.

As mentioned before, the ground-truth behavior per axis is supposed to be strictly monotonous for each segment. Rephrased in mathematical terms, the learner y therefore needs to fulfill the following properties:

$$L_d(\mathbf{u}) = -\frac{\partial}{\partial u_d} y_d(\mathbf{u}) < 0 : \forall u \in \mathcal{S} \ , \tag{14}$$

where $d = 1, 2, 3$ is the output dimension, u is an input for the respective segment, and $\mathcal{S}$ is the three-dimensional input space defined by the minimal and maximal length of the segment (see Tab. 1). In order to rephrase this definition in terms of Eq. (2), we set $\mathbf{m}^i = i$, $\gamma_i = 1$ and $c = 0$.

Results. The performance of three different constrained models is evaluated on the BHA data. We tested a linear model (LM), a partially monotonic multi-layer perceptron (PMMLP) [3] and the ELM model - introduced in this paper.

The ELMs used in the experiments have $R = 300$ hidden layer neurons and are optimized by BIP with $\mu_{\text{BIP}} = 0.2$. The regularization parameters $\alpha = 10^{-5}$, 10^{-3}, and 10^{-2} are obtained by line search for the segments 1,2, and 3 respectively. The input space of the ELM where the constraint is implemented is defined by the morphology of the BHA and its physical restrictions stated in Tab. 1. $N_{\mathrm{re}} = 10^6$ samples are used for rejection sampling. Additionally, a lower bound of $c = 100$ mbar/meter (defined with some tolerance $\varepsilon = 100$ mbar/meter) is implemented. Only networks where the algorithm terminates successfully were taken into account in the experiments.

The results are obtained by cross-validation over the five trials measured by the error function in Eq. (13). For each fold four trials are used for training and one trial is used for testing the generalization ability of the models. Afterwards, the errors are averaged over the five folds. We tested 100 different network initializations and state the training and test errors of the best performing models in Tab. 2. The mapping

Table 2. Best training/test errors for the different models out of 100 initializations.

Segment	LM	PMMLP	ELM
1	50.1/54.0	39.84/46.24	**30.77/39.76**
2	64.5/69.0	47.69/56.19	**36.29/49.77**
3	60.9/68.2	50.03/60.40	**43.80/58.23**

ability of the LM is too poor to capture enough structure encoded in the BHA data - the errors are large. The PMMLP performs better than the LM since it induces non-linearity. Unfortunately, a tuning of the PMMLP needs to be done manually; its performance is depending on the experience of the tuning expert. The mapping of the PMMLP can be further optimized by use of more sophisticated tuning methods regarding the number of neurons, but a specific tuning remains difficult since the performance of the model is highly depending on the initialization of the weights. The ELM - in contrast - performs significantly better although it was provided with the same number of parameters in the hidden layer.

In order to show the quality of the produced mapping by the PMMLP and ELM, also the mean and standard deviation over the different initializations are stated in Tab. 3. In order to make the results comparable, both networks are provided with the same number of neurons. The mean performance of the ELM is close to the

Table 3. Mean training errors $\pm$ standard deviation of the training errors / mean test errors $\pm$ standard deviation of the test errors for 100 initializations.

Segment	PMMLP	ELM
1	43.60±2.09/49.96±1.91	**31.01±0.11/39.92±0.10**
2	51.04±1.93/59.39±1.86	**36.61±0.15/50.13±0.14**
3	55.31±3.20/65.78±3.05	**43.89±0.06/58.40±0.09**

best performance. The standard deviation of the errors is very low in comparison to the PMMLP. The main reason for this is the use of BIP and ridge regression producing a task-specific representation in the hidden layer irrespective of the random initialization and additionally regularizing the read-out layer which was rigorously analyzed in [11,24].

The experiments show that the ELM is outperforming the other models significantly. One essential reason is that the two competing models (LM, PMMLP) fulfill the constraint globally by means of model selection, although only a dimension-wise monotony is required. This overly conservative model selection of LM and PMMLP reduces their approximation capability too much. However, to the best of the authors' knowledge there is no simple way to implement the desired partial constraint through a model selection mechanism, which underscores the flexibility of the presented approach. The effort of quadratic optimization and verification of the learned ELM function pays off, because the local implementation of the continuous constraint (if enforced only in the previously defined region $\mathcal{S}$ of the input space) leaves more degrees of freedom in the model for approximation of the actual non-linear mapping.

6. Conclusion

The presented approach shows that the ELM is particularly well suited to include prior knowledge into the learned function by means of adding constraints which are linear functions of arbitrary derivatives. Quadratic optimization can guarantee fulfillment of such constraints after learning in discrete points, whereas the step towards continuous constraints needs additional effort. The ELM's architecture motivates an efficient verification algorithm, which is based on a worst-case approximation of the Taylor expansion of the learned function. To derive the respective bounds is only possible because derivatives and maxima can be analytically determined due to the special form of the ELM. Together with an iterative sampling, this approach provides all the tools to combine the full representational power of the ELM with guarantees on the performance of the learned function, which are crucial in many engineering and automatic control applications. The flexibility to define constraints locally and selectively in continuous regions distinguishes the presented ELM approach from other schemes that for instance guarantee monotonicity by means of a-priori model selection.

The real world example provides both an interesting illustrative application and additional insight on the usefulness of the proposed scheme. It demonstrates a typical case, where data-driven learning is needed because no analytic models are available, but is difficult because sampling can be done only in the real world and is expensive and noisy. Prior knowledge in form of constraints provides additional and necessary information to the learner. It defines how to generalize from training data, even though exhaustive sampling is impossible. The results are very encouraging and actually the learned controller was used in a further learning level to obtain an inverse mapping through exploration on the real robot (^{19}, video ^{20}).

Obviously, the iterative verification and sampling algorithm becomes expensive with increasing output dimensionality of the learned function and will not be practical in very high dimensions. However, in many applications in automatic control, the output is a control signal and will be rather one- or low-dimensional. In such cases, the proposed scheme is very efficient, optimally uses the special architectural setting of the ELM to compute all necessary quantities, and provides guarantees on particular constraints without introducing too much architectural bias. We expect that this treatment can open new application domains in particular for technical systems for data driven learning, where reliability is crucial and traditionally only analytic modeling has been possible.

Acknowledgements

This research and development 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) and managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the contents of this publication.

References

1. E. Hartman, “Training feedforward neural networks with gain constraints”, *Neural Computation*, vol. 12, no. 4, pp. 811–829, 2000.
2. B.-G. Hu, Y. Wang, S. H. Yang, and H. B. Qu, “How to add transparency to artificial neural networks”, *Pattern Recognition and Artificial Intelligence*, vol. 20, no. 1, pp. 72–84, 2007.
3. B. Lang, “Monotonic multi- layer perceptron networks as universal approximators”, *Proc. ICANN*, vol. 3697, pp. 750–750, 2005.
4. H. Daniels and M. Velikova, “Monotone and partially monotone neural networks”, *IEEE Transactions on Neural Networks*, vol. 21, no. 6, pp. 906–917, 2010.
5. F. Han, T.-M. Lok, and M. Lyu, “A new learning algorithm for function approximation incorporating a priori information into extreme learning machine”, in *Advances in Neural Networks*, vol. 3971, pp. 631–636, Springer-Verlag Berlin, 2006.
6. F. Lauer and G. Bloch, “Incorporating prior knowledge in support vector regression”, *Machine Learning*, vol. 70, pp. 89–118, 2008.
7. Y.-J. Qu and B.-G. Hu, “Generalized constraint neural network regression model subject to linear priors”, *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2447–2459, 2011.
8. Z. Sun, Z. Zhang, H. Wang, and M. Jiang, “Cutting plane method for continuously constrained kernel based regression”, *IEEE Transactions on Neural Networks*, vol. 21, no. 2, pp. 238–247, 2010.
9. G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: A new learning scheme of feedforward neural networks”, in *Proc. IJCNN*, (Budapest, Hungary), 2004.
10. G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: Theory and applications”, in *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

11. K. Neumann and J. J. Steil, "Batch intrinsic plasticity for extreme learning machines", in *Proc. ICANN*, vol. 6791, no. 1, pp. 339–346, 2011.
12. 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.
13. P. Gill, W. Murray, and M. Wright, *Practical Optimization.* Academic Press, 1981.
14. "Deutscher Zukunftspreis (German Future-Award)", 2010.
15. K. J. Korane, "Robot imitates nature", *Machine Design*, vol. 82, no. 18, 2010.
16. A. Grzesiak, R. Becker, and A. Verl, "The bionic handling assistant - a success story of additive manufacturing", *Assembly Automation*, vol. 31, no. 4, 2011.
17. M. W. Hannan and I. D. Walker, "Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots", *Journal of Robotic Systems*, vol. 20, no. 2, 2003.
18. C. Laschi, B. Mazzolai, V. Mattoli, M. Cianchetti, and P. Dario, "Design of a biomimetic robotic octopus arm", *Bioinspiration and Biomimetics*, vol. 4, no. 1, 2009.
19. M. Rolf and J. Steil, "Efficient exploratory learning of inverse kinematics on a bionic elephant trunk", *IEEE Trans. NNLS*, 2013. submitted.
20. M. Rolf and J. Steil, "Learning to reach with festo's bionic handling assistant", http://www.youtube.com/watch?v=aGFiMviI0VY.
21. Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally Pruned Extreme Learning Machine", *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 158–162, 2010.
22. J.-H. Zhai, H.-Y. Xu, and X.-Z. Wang, "Dynamic Ensemble Extreme Learning Machine based on Sample Entropy", *Soft Computing*, vol. 16, no. 2, pp. 1493–1502, 2012.
23. X.-Z. Wang, Q.-Y. Shao, Q. Miao, and J.-H. Zhai, "Architecture selection for networks trained with extreme learning machine using localized generalization error model", *Neurocomputing*, vol. 102, no. 1, pp. 3-9, 2013.
24. K. Neumann, J. J. Steil, "Optimizing extreme learning machines via ridge regression and batch intrinsic plasticity", *Neurocomputing (Advances in Extreme Learning Machines, ELM 2011)*, vol. 102, no. 1, pp. 23–30, 2013.
25. G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey", *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.