Optimizing Extreme Learning Machines via Ridge Regression and Batch Intrinsic Plasticity

Klaus Neumann[a,*], Jochen J. Steil[a]

[a]*Research Institute for Cognition and Robotics (CoR-Lab)*
Faculty of Technology, Bielefeld University
Universitätsstr. 25, 33615 Bielefeld, Germany

Abstract

Extreme learning machines are randomly initialized single-hidden layer feed-forward neural networks where the training is restricted to the output weights in order to achieve fast learning with good performance. This contribution shows how batch intrinsic plasticity, a novel and efficient scheme for input specific tuning of non-linear transfer functions, and ridge regression can be combined to optimize extreme learning machines without searching for a suitable hidden layer size. We show that our scheme achieves excellent performance on a number of standard regression tasks and regression applications from robotics.

Keywords: Neural network, learning, extreme learning machine, batch intrinsic plasticity, ridge regression, regularization.

1. Introduction

Huang et al. introduce to use the extreme learning machine (ELM) [1] which is a learning scheme based on random projections. It is appealing because of the high efficiency, conceptual simplicity and the good learning results (a survey on ELM techniques can be found in [2]). As opposed to random projections for dimensionality reduction, which have been considered in [3, 4], it is characteristic for the ELM to use high-dimensional projections, which often actually increase the feature dimensionality. The differences of the models are mentioned in [5]. The relation between the ELM approach and earlier proposed feedforward random projection methods is further discussed in [6, 7].

Despite the apparent simplicity of the ELM approach, its task performance strongly depends on the random initialization and the size of the hidden layer. A good performance is usually achieved by manually tuning the ELM parameters

*Corresponding author
Email address: `kneumann@cor-lab.uni-bielefeld.de` (Klaus Neumann)

to a task-suitable regime which can lead to good generalization results (e.g. see [8, 9]) but in a clearly heuristic way.

Since the ELM is based on the empirical risk minimization principle [10], it leads to overfitting of the data, in particular if the task does not comprise many training samples. To match this challenge, variations to the ELM have recently been developed that are based on the idea to change the hidden layer size or to regularize the output weights in order to improve the generalization ability.

Different ways to determine a suitable hidden layer size were already introduced and benchmarked: Huang shows in [11] that the universal function approximation ability of the ELM is preserved when incrementally updating the size of the hidden layer, in a model named incremental extreme learning machine (I-ELM). Related ideas to improve ELMs are the optimally pruned extreme learning machine (OP-ELM) [12] reducing the hidden layer size and the error minimized extreme learning machine (EM-ELM) [13] which aims at increasing the hidden layer size, optionally in groups of neurons.

Methods based on the introduction of an additional weight decay term in the error function are the regularized extreme learning machine (RELM) [14] and the recently developed Tikhonov regularized optimally pruned extreme learning machine (TROP-ELM) [15], while the TROP-ELM approach combines the idea of regularization with the OP-ELM. Both use Tikhonov regularization [16] to control the norm of the output weights, a technique that is also called ridge regression (RR). The method is well known and established also in other domains of neural network research [17] and machine learning in general [18].

However, the random initialization (RND) - i.e. the input weights and the biases of the non-linear transfer functions - have a great influence on the task performance as well. Methods only controlling the network size and the output matrix are insufficient in tuning the neurons to a good regime, where the encoding is optimal. Additional methods to improve the encoding by adapting the neurons' parameters therefore have a great potential for a better performing ELM. The experiments in this contribution show that saturated and almost constant hidden layer neurons lead to an encoding where the prediction capability is poor. This motivates techniques to tune the output distributions of hidden layer neurons in a regime where the encoding is suited for good generalization.

To achieve this goal, we are inspired by a biologically plausible mechanism first introduced by Triesch [19] under the notion of intrinsic plasticity (IP). It is shown in [20], that IP enhances the encoding in recurrent neural networks. The output is forced by IP to approximate exponential distributions. This maximizes the network's information transmission, caused by the high entropy of the distribution. Inspired by IP, recently, a novel method called batch intrinsic plasticity (BIP) [21] was introduced to optimize ELMs. BIP is used for pretraining and adapts the activation function of the hidden layer neurons analytically by a pseudo inverse technique such that desired output distributions are achieved. This makes BIP highly efficient.

This contribution shows that ELMs pretrained by BIP have a significantly better generalization ability than a randomly initialized ELM when trained with RR without tuning the size of the hidden layer on different real world tasks. The approach is also compared with respect to performance and computational time with other state of the art optimization techniques for ELMs.

2. The ELM, Regularization and Batch Intrinsic Plasticity

The ELM consists of three layers: $\mathbf{u} \in \mathbb{R}^I$ denotes 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 $\mathbf{W}^{\text{in}} \in \mathbb{R}^{R\times I}$, while the read-out matrix $\mathbf{W}^{\text{out}} \in \mathbb{R}^{O\times R}$ comprises the read-out weights. When denoting the weights from the input layer to a specific hidden layer neuron i with $\mathbf{W}^{\text{in}}_i \in \mathbb{R}^{1\times I}$, the ELM scheme then becomes

$$\mathbf{y} = \mathbf{W}^{\text{out}}\mathbf{h} = \mathbf{W}^{\text{out}} \left(\dots, f\left(a_i \mathbf{W}^{\text{in}}_i \mathbf{u} + b_i\right), \dots\right)^T \quad , \tag{1}$$

where h denotes the actual state of the hidden layer, a_i is the slope and b_i the bias of the i-th hidden layer neuron. Traditionally, the ELM is created by randomly initializing the input matrix, the slopes a_i and the biases b_i $(i = 1, \dots R)$ in the - typically sigmoid - activation function. Usually the slopes are set to one.

2.1. Supervised Read-Out Learning for Extreme Learning Machines

Supervised learning for ELMs is restricted to the read-out weights $\mathbf{W}^{\text{out}}$. In order to infer a desired input-output mapping from a set of N_{tr} training samples $(\mathbf{u}(k), \mathbf{t}(k))$ with $k = 1 \dots N_{\text{tr}}$ and simultaneously punishing large weight values in the read-out matrix, the read-out weights $\mathbf{W}^{\text{out}}$ are adapted such that the sum of the mean square error and a regularization term for the training set is minimized:

$$E = \frac{1}{N_{\text{tr}}} \sum_{k=1}^{N_{\text{tr}}} ||\mathbf{t}(k) - \mathbf{y}(k)||^2 + \varepsilon ||\mathbf{W}^{\text{out}}||^2 \rightarrow \min \quad . \tag{2}$$

To achieve this, the paper focuses on batch training and uses a standard ridge regression method introduced for ELMs as regularized extreme learning machine (RELM) in [14]. The network's states $\mathbf{h}(k)$ belonging to the inputs $\mathbf{u}(k)$ as well as the desired output targets $\mathbf{t}(k)$ are collected in a state matrix $\mathbf{H} = (\mathbf{h}(1) \dots \mathbf{h}(N_{\text{tr}})) \in \mathbb{R}^{R\times N_{\text{tr}}}$ and a target matrix $\mathbf{Y} = (\mathbf{y}(1) \dots \mathbf{y}(N_{\text{tr}})) \in \mathbb{R}^{O\times N_{\text{tr}}}$. The optimal read-out weights are then determined by the least squares solution

$$\mathbf{W}^{\text{out}} = \mathbf{Y}\mathbf{H}^T \left(\mathbf{H}\mathbf{H}^T + \varepsilon \mathbf{1}\right)^{-1} , \tag{3}$$

where the factor $\varepsilon \geq 0$ was identified by Tikhonov in [16] as output regularization strength. This least squares solution converges to the pseudo inverse for small values of ε.

2.2. Intrinsic Plasticity

BIP is inspired by a learning rule called intrinsic plasticity (IP) which was developed by Triesch in 2004 [19] as a model for homeostatic plasticity for analog neurons with parameterized Fermi-functions $f(x) = (1 + \exp(-ax - b))^{-1}$. The goal is to optimize the information transmission of a single neuron strictly local and online by adaptation of slope a and bias b of the Fermi-function such that the neuron's output h becomes exponentially distributed. IP-learning can be derived by minimizing the difference $D(f_h, f_{\text{exp}})$ between the output f_h and an exponential distribution f_{exp}, quantized by the Kullback-Leibler-divergence (KLD) [22]:

$$D(f_h, f_{\text{exp}}) = \int\limits_{\Omega} f_h(h) \log\left(\frac{f_h(h)}{f_{\text{exp}}(h)}\right) = -H(h) + \frac{1}{\mu}\mathbb{E}(h) + \log(\mu) \ , \qquad (4)$$

where $H(h)$ denotes the entropy and $\mathbb{E}(h)$ the expectation value of the output distribution. In fact, minimization of $D(F_h, F_{\text{exp}})$ in Eq. (4) for a fixed $\mathbb{E}(h)$ is equivalent to entropy maximization of the output distribution. For small mean values, i.e. $\mu \approx 0.2$, the neuron is forced to respond strongly only for a few input stimuli. The following online update equations for slope and bias - scaled by the step-width η_{IP} - are obtained:

$$\Delta a = \frac{\eta_{\text{IP}}}{a} + x\Delta b \qquad \Delta b = \eta_{\text{IP}}\left(1 - \left(2 + \frac{1}{\mu}\right)h + \frac{1}{\mu}h^2\right) \ . \qquad (5)$$

The only quantities used to update the neuron's non-linear transfer function are x, the synaptic sum arriving at the neuron, the firing rate h and its squared activity h^2.

2.3. Batch Intrinsic Plasticity

The task performance of an ELM strongly depends on the random initialization of the input matrix and the biases. Without expert-tuning by means of additional task knowledge, a random initialization can lead to the problem of saturated or constant neurons. This can be avoided by finding activation functions which are in a favorable regime. Since IP is an online rule it is too computationally expensive for ELMs. Thus, we use BIP - a highly efficient batch version of IP - to adapt the activation function such that certain output distributions are achieved.

Only the inputs $\mathbf{u} = (\mathbf{u}(1), \mathbf{u}(2) \dots \mathbf{u}(N_{\text{tr}})) \in \mathbb{R}^{I \times N_{\text{tr}}}$ are used for optimization. The goal is to adapt slope a_i and bias b_i of the activation function such that the desired distribution f_{des} for the neuron's outputs $h_i(k) = f(a_i s_i(k) + b_i)$ is realized.

To implement BIP, a linear regression problem is formulated, where random virtual targets $\mathbf{t} = (t_1, t_2 \dots t_{N_{\text{tr}}})^T$ are drawn in ascending order $t_1 < \dots < t_{N_{\text{tr}}}$ from the desired output distribution f_{des}. Fig. 1 illustrates how BIP works: the targets $\mathbf{t}$ and the input stimuli $\mathbf{s}_i$ are combined to build data pairs inducing a regression problem. The synaptic sum arriving at neuron i when stimulated by

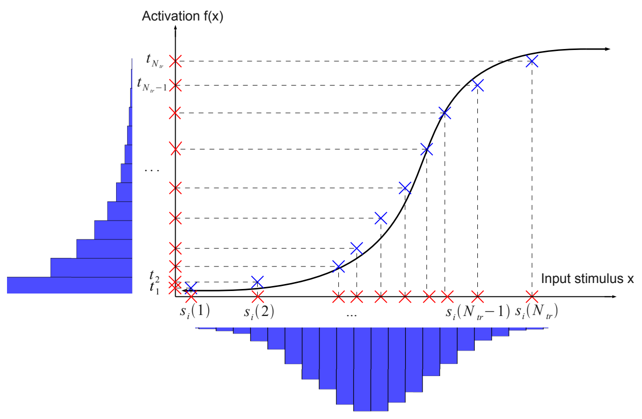

Figure 1: Batch intrinsic plasticity formulated as a regression problem.

training sample k is given by $s_i(k) = \mathbf{W}_i^{\text{in}}\mathbf{u}(k)$ and collected in $\mathbf{s}_i = \mathbf{W}_i^{\text{in}}\mathbf{u}$. Since the stimuli need to be mapped onto the right targets, a rearrangement of the stimuli in ascending order $s_i(1) < \cdots < s_i(N_{\text{tr}})$ is done by sorting $\mathbf{s}_i \leftarrow \text{sort}(\mathbf{s}_i)$. This is necessary because a monotonically increasing activation function f is used to map all incoming training stimuli on the right targets and infer the desired distribution f_{des} for the neuron's output. Defining the data matrix $\Phi(\mathbf{s}_i) = \left(\mathbf{s}_i^T, (1 \ldots 1)^T\right)$ and the parameter vector $\mathbf{v}_i = (a_i, b_i)^T$ learning for the i-th neuron is formulated as a linear and over-determined regression problem, where the outputs are mapped onto the targets $h_i(k) \approx t_k$:

$$\|\Phi(\mathbf{s}_i) \cdot \mathbf{v}_i - \mathbf{f}^{-1}(\mathbf{t})\| \to \min \quad . \tag{6}$$

The solution for the optimal slope a_i and bias b_i is obtained by computation of the Moore-Penrose pseudo inverse [23]:

$$\mathbf{v}_i = (a_i, b_i)^T = \Phi^\dagger(\mathbf{s}_i) \cdot \mathbf{f}^{-1}(\mathbf{t}) \quad . \tag{7}$$

Typically Fermi and tanh functions are used as activation functions. The algorithm is always terminating, since the pseudo inverse always exists. It is also important that the virtual targets are in $[0, 1]$ (if the Fermi function is used). The learning is done in an one shot fashion and summarized in Alg. 1.

The pretraining is of the same order of complexity as the supervised read-out learning, since only the least squares solutions of the linear model Φ have to be calculated. Only the virtual targets t_i used as meta-parameters are necessary to fully define the regression model. Since the method is unsupervised, Alg. 1 has to be recomputed if new input samples are used, e.g. when changing the task. The following sections 2.4 and 3 investigate on the behavior of BIP when stimulated with different inputs.

Algorithm 1: batch intrinsic plasticity (BIP) for ELMs

```
**initialize** ELM randomly
**require** get inputs $\mathbf{u} = (\mathbf{u}(1), \mathbf{u}(2) \dots \mathbf{u}(N_{\text{tr}}))^T$
**for all** hidden neurons $i$ **do**
    get stimuli $\mathbf{s}_i = \mathbf{W}_i^{\text{in}} \cdot \mathbf{u}$
    draw targets $\mathbf{t} = (t_1, t_2 \dots t_{N_{\text{tr}}})^T$ from desired distribution $f_{\text{des}}$
    sort targets $\mathbf{t} \leftarrow \text{sort}(\mathbf{t})$ and stimuli $\mathbf{s}_i \leftarrow \text{sort}(\mathbf{s}_i)$
    construct $\Phi(\mathbf{s}_i) = \left(\mathbf{s}_i^T, (1 \dots 1)^T\right)$
    calculate (pseudo-)inverse $(a_i, b_i)^T = \mathbf{v}_i = \Phi(\mathbf{s}_i)^\dagger \cdot \mathbf{f}^{-1}(\mathbf{t})$
**end for**
**return** $\mathbf{v} = (v_1, v_2 \dots v_R)^T$
**incoporate** collected slopes and biases in $\mathbf{v}$ into ELM
```

2.4. Shaping Output Distributions via Batch Intrinsic Plasticity

To illustrate the behavior of the BIP-learning, a single-neuron model with different fixed input distributions f_s is considered. $N_{\text{tr}} = 50$ samples are used for training and $N_{\text{te}} = 1000$ samples for testing - both drawn from f_{s}.

Three different input and output distributions are taken into account: $f_{\text{des}} = f_{\text{s}} = \exp(\text{onential})$, norm(al), and uni(form). The moments of the distributions are: $\mu(\exp) = 0.2$, $\sigma(\exp) = 0.2$, $\mu(\text{norm}) = 0.5$, $\sigma(\text{norm}) = 0.1$, $\mu(\text{uni}) = 0.5$, and $\sigma(\text{uni}) = 0.3$. These values are used for the experiments collected in Tab. 1.

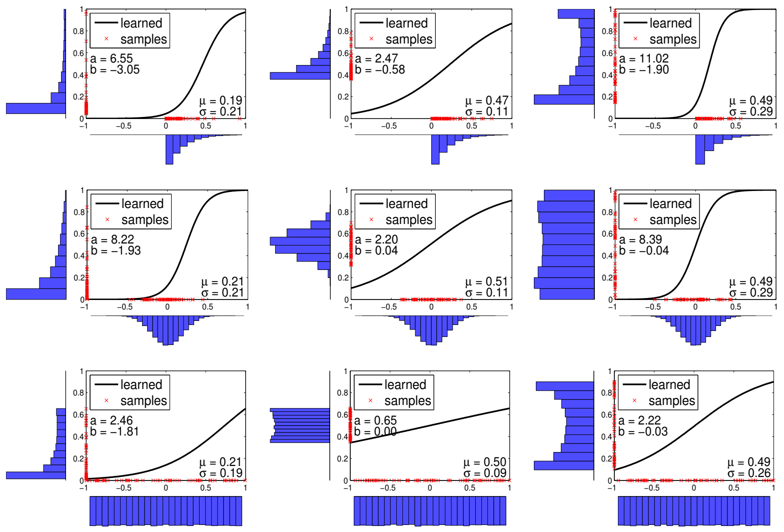

Figure 2: A neuron's activation function adapted by BIP to approximate the output distributions f_{des} while starting from the input distributions f_{s}. The input distribution is varied over the rows, while the output distributions varies column-wise.

We demonstrate the ability of BIP by repeating results from [21] for the sake of completeness of the current paper. Fig. 2 illustrates the result of adapting

μ/σ	exp	norm	uni
exp	0.18±0.02	0.49±0.01	0.46±0.04
	0.21±0.03	0.08±0.01	0.25±0.02
norm	0.20±0.02	0.50±0.01	0.49±0.04
	0.19±0.02	0.09±0.01	0.29±0.01
uni	0.19±0.02	0.49±0.01	0.49±0.03
	0.18±0.02	0.09±0.01	0.31±0.01

Table 1: Fits of output distributions. A cell contains mean and standard deviation of μ and σ.

the neuron's nonlinear transfer function. The input distribution is assigned to the rows of the figure, while the desired output distribution is given column-wise. The incoming training stimuli are visualized by the crosses on the x-axis, while the corresponding targets are displayed on the y-axis. The x-axis shows a histogram of the synthetically created test stimuli while the y-axis shows a histogram of the outputs produced by the learned activation function transforming the inputs. Especially when stimulated with Gaussian input, the neuron is able to achieve the three desired output distributions very accurately - illustrated by the second row in Fig. 2. It is demonstrated in the first column of Fig. 2 that the exponential distribution is approximated for all inputs. However, since the sigmoid activation function has only two degrees of freedom, the match is typically not perfect. The figure shows that large deviations from the optimal output distribution can sometimes be observed.

Further statistics are summarized in Tab. 1. The table shows the results obtained for one neuron which is trained by BIP for 100 trials. After each trial, the mean and the standard deviation of the output distribution are collected which determines the deviation of samples from the desired distribution. The table shows, that μ and σ of the output distribution are always approximated very well with low variance.

3. The Importance of Output Distributions

The following section describes why shaping of output distributions of hidden layer neurons in ELMs is important and demonstrates its influence on the networks' performance.

We analyze the extrapolation and interpolation behavior of the trained ELMs on two robotics tasks where we artificially tune the neurons to undesired regimes. It is additionally investigated how strong the results of BIP depend on the random initialization of the input matrix and the biases.

The hidden layers have $R = 200$ neurons, the parameterized Fermi-function $f(x) = 1/(1 + e^{-ax-b})$ is used as activation function and the desired output used to produce the targets for the BIP learning is the exponential distribution $f_{\text{des}} = f_{\text{exp}}$ with a fixed mean $\mu = 0.2$. The networks for the robotics interpolation task had a ridge regression parameter of $\varepsilon_{\text{opt}} = 10^{-4}$ while the networks for the extrapolation task had a ridge regression parameter of $\varepsilon_{\text{opt}} = 10^{-5}$. These

RR parameters where identified as parameters leading to the lowest test error by line search in the range $\varepsilon \in [10^{-9}, 10^{-8}, \ldots, 10^{3}]$.

The task is to learn the observed inverse kinematics mapping between joint and task space of a redundant six degrees of freedom (DOF) robot arm shown in Fig. 3 and Fig. 4. $N_{\mathrm{tr}} = 100$ (RoboInter) and $N_{\mathrm{tr}} = 200$ (RoboExtra) training samples and $N_{\mathrm{tr}} = 1000$ (RoboInter) and $N_{\mathrm{tr}} = 2000$ (RoboExtra) test samples are generated by projecting a task trajectory specified in Cartesian end-effector coordinates into the joint space of the robot arm by means of the analytically calculated inverse kinematics function $F : \mathbb{U} \rightarrow \mathbb{Y}$, where $\mathbb{U}$ is the task and $\mathbb{Y}$ the joint space. For each task space input $(u_1(k) \ldots u_6(k))^T$ containing the end-effector position and orientation the six-dim target vector $(y_1(k) \ldots y_6(k))^T$ is computed and additionally corrupted with Gaussian-noise ($\sigma_N = 0.1$). The generated trajectory forms an eight for the robotics interpolation task - see Fig. 3 - and four circles for the extrapolation task where only the second and third circle were used for training and the first and the fourth cycle for testing - see Fig. 4.

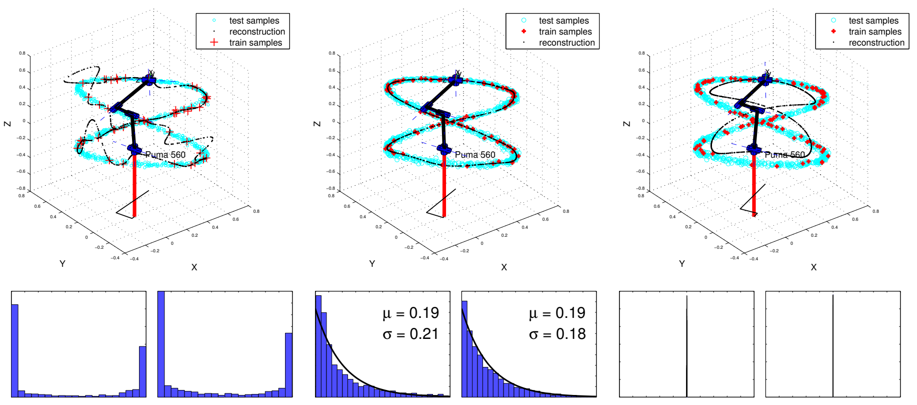

Figure 3: The effect of saturated (left) and almost constant hidden layer neurons (right) on the robotics interpolation task (RoboInter). BIP produces desired output distributions favorable for generalization (center).

Fig. 3 and Fig. 4 show the results for ELMs which hidden layer neurons are tuned into a specific regime in order to clearly extract the effect of BIP. Fig. 3 is separated into two rows. The first row illustrates the robotics interpolation task. The second row shows the output distributions of two hidden layer neurons for the three different examples respectively. Fig. 3 (left) demonstrates a situation where the neurons are saturated. This was achieved by drawing the input matrix weights and biases uniformly from $[-10, 10]$. The first two images in the second row show that the neurons in this case have an almost binary coding. The robot arm is supposed to follow an eight-like trajectory which is not approximated accurately. The generalization ability of the ELM is poor, strong overfitting occurs. Fig. 3 (right) shows the effect of hidden layer neurons with almost constant activation functions producing peak-like output distributions. The input weights and biases are drawn from $[-0.1, 0.1]$. Therefore, the complexity of the network is to low and the mapping can not be approximated.

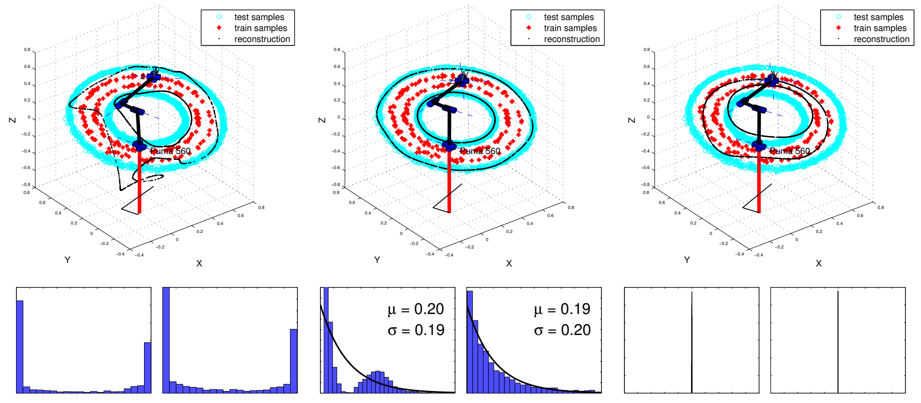

Figure 4: BIP trained ELMs produce better mappings for the robotics extrapolation task (RoboExtra).

Fig. 3 (center) illustrates the mapping results for an ELM which was trained by BIP after the random initialization in a saturated or constant regime. The hidden layer neurons approximate exponential distributions with fixed mean $\mu = 0.2$ illustrated by the third and the fourth histogram in the second row of Fig. 3 - the network performs well. The shaping of the output distribution clearly improves the generalization ability of the network.

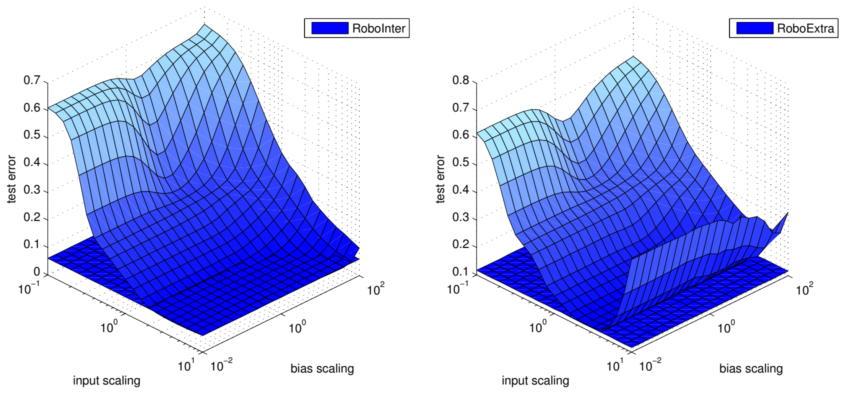

Figure 5: Performance of RELMs on the test set measured as RMSE for different input matrix and bias scalings.

Fig. 4 shows the same as Fig. 3 for the robotics extrapolation task. The generalization ability in the case where saturated hidden layer neurons are provided in the hidden layer is even worse than in the robotics interpolation task. In this task, BIP optimizes the encoding of the neurons such that the mapping is approximated very well. Note, that the desired distribution is not always produced with high accuracy (see third plot in the second row of Fig. 4). However, mean and standard deviation are approximated very well.

These examples demonstrate that the shaping of the output distributions for the hidden layer neurons separates into two effects: weights which are too large or too small for the given task are scaled and the encoding in the hidden layer is sparsified. The combination of these effects leads to a good generalization ability.

Fig. 5 shows how the test error (RMSE) changes when the input matrix W^{in} and the biases b are scaled with a factor. After scaling the input matrix and the biases, BIP-training is applied. Fig. 5 (left) illustrates the results for the robotics interpolation and Fig. 5 (right) for the robotics extrapolation task. The test error for the randomly initialized ELM highly depends on the initialization while the networks trained with BIP perform very well independently of the initialization. BIP compensates the scaling of the incoming stimuli such that a good generalization capability occurs indicated by a flat error surface below the one for the randomly initialized ELMs.

4. Batch Intrinsic Plasticity for Benchmark Tasks

The following section describes experiments focusing on the impact of BIP and RR on the task performance of different real world tasks. The input matrix components W^{in}_{ij}, the biases b_i for a respective ELM are drawn from the uniform distribution in $[-1, 1]$. The Fermi-function $f(x) = 1/(1 + e^{-a_i x - b_i})$ is again used as activation function for the networks. The desired output producing the targets for the BIP learning is the exponential distribution $f_{\text{des}} = f_{\text{exp}}$ with a fixed mean μ separately drawn for each neuron from $\mu \in [0.05, 1]$. BIP is applied after the random initialization of the networks as pretraining method. The results are averaged over 20 different network initializations for each data set. The statistics over the networks is contained in the following tables. The optimal RR parameter and network size is determined by a line search. All experiments have been applied on a x86_64 linux machine with at least 4 GB of memory and a 2+ GHz intel processor in matlab R2010a.

All data sets are regression tasks from the UCI machine learning repository - detailed information can be found in [24]. The separation of the data sets into training and test data is done as specified in the UCI repository for the respective tasks (see Table 2). The input data has been normalized to $[-1, 1]$, while the output data was normalized into the range $[0, 1]$.

The results of the learning for three of the tasks are illustrated in Fig. 6. The first row shows the development of the test error for growing hidden layer size when trained with linear regression. The randomly initialized ELMs strongly overfit the training data when the network becomes large. This effect is significantly reduced when pretrained with BIP. Whereas the effect is present for all data sets, we show only three for illustration in Fig. 6.

The second row in Fig. 6 displays the development of the test error when changing the ridge regression parameter for networks with $R = 500$ hidden

Task	Abbreviation	Attributes	# Training	# Test
Abalone	Ab	8	2000	2177
CaliforniaHousing	Ca	8	8000	12640
CensusHouse8L	Ce	8	10000	12784
DeltaElevators	De	6	4000	5517
ComputerActivity	Co	12	4000	4192
Elevators	El	18	8725	7847

Table 2: Specification of the 6 used regression data sets from the UCI machine learning repository [24].

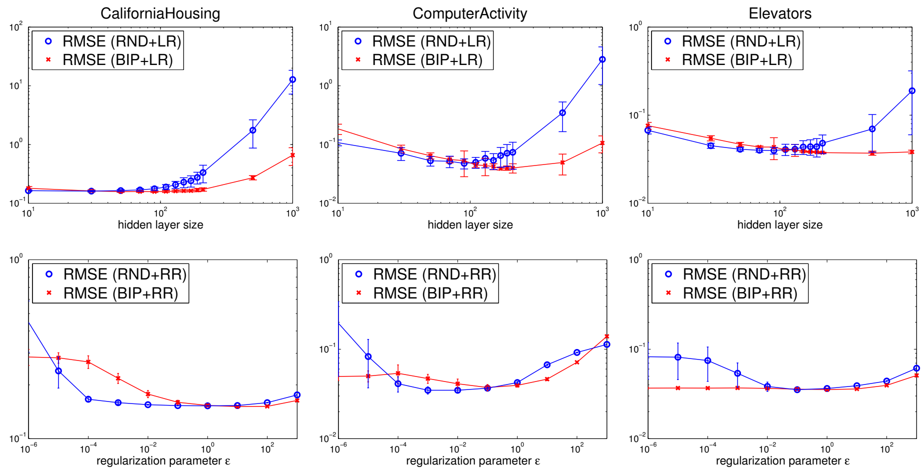

Figure 6: RMSE development on the test set for growing hidden layer size and different ridge regression parameters.

neurons. The networks' show a typical generalization behavior where the models first overfit the data, reach an optimal value and then degenerate for too strong regularization. The BIP-pretrained ELMs perform better than the randomly initialized networks in a large range of output regularization parameters and are by far less sensitive to its choice.

The results for the best performing ELMs are summarized in Tab. 3 and Tab. 4. Tab. 3 contains the performance results on the test sets for randomly initialized ELMs and BIP-trained ELMs with linear regression. The results are given for the best obtained hidden layer size which was varied in $R \in [5, 10, \ldots, 500, 1000]$. The BIP-pretrained ELMs perform better than the purely random initialized networks for the majority of the tested regression tasks and lead to stable results with low variance. The results also show that the BIP-pretrained networks can handle larger networks without overfitting the data. The change in the hidden layer encoding shows a significant impact on the networks' performance.

Since large networks have a good generalization ability after training with

Task	BIP	R	RND	R
Ab	0.0753 ± 0.0012	35	$\mathbf{0.0750 \pm 0.0009}$	35
Ca	$\mathbf{0.1577 \pm 0.0051}$	90	0.1587 ± 0.0034	25
Ce	$\mathbf{0.0657 \pm 0.0009}$	210	0.0675 ± 0.0019	110
De	0.0539 ± 0.0003	130	$\mathbf{0.0529 \pm 0.0002}$	70
Co	$\mathbf{0.0390 \pm 0.0022}$	170	0.0477 ± 0.0094	90
El	$\mathbf{0.0371 \pm 0.0020}$	500	0.0397 ± 0.0035	90

Table 3: The results (RMSE) of ELMs pretrained with BIP for real world regression tasks in comparison to randomly initialized ELMs. The networks are stated with the optimal hidden layer size.

BIP, ELMs with a $R = 500$ neuron hidden layer are used for further experiments. We expect the BIP networks in this case to perform better than the random networks after supervised readout training with RR, since the previous experiments show that BIP improves the generalization ability significantly for large networks. We state the test error results for the best performing networks with the corresponding output regularization strength ε which was found in $\varepsilon \in [10^{-9}, 10^{-8}, \ldots, 10^{3}]$ in Tab. 4.

Task	BIP	$\log \varepsilon$	RND	$\log \varepsilon$
Ab	$\mathbf{0.0734 \pm 0.0002}$	1	0.0739 ± 0.0001	-1
Ca	$\mathbf{0.1492 \pm 0.0013}$	2	0.1525 ± 0.0006	0
Ce	$\mathbf{0.0628 \pm 0.0002}$	0	0.0640 ± 0.0006	-3
De	$\mathbf{0.0526 \pm 0.0001}$	1	0.0527 ± 0.0002	-2
Co	$\mathbf{0.0342 \pm 0.0010}$	0	0.0350 ± 0.0017	-2
El	$\mathbf{0.0357 \pm 0.0003}$	0	0.0357 ± 0.0010	-1

Table 4: BIP-pretrained and randomly initialized ELMs trained with RR. The networks are stated with the optimal RR parameter.

In order to compare the results obtained for the BIP-pretrained ELMs with state of the art optimization techniques for ELMs, we summarize the performance results from other methods in Table 5. The performance results for the RELM, EI-ELM, I-ELM and CI-ELM where taken from [14, 7, 11, 25] respectively.

It is shown that the BIP-pretrained ELMs additionally trained with RR can perform better than other state of the art models. However, the optimization is done in a complementary way, which is important to note. In such a case, it is possible to combine those methods with BIP. Interestingly, the RELM performs best on almost all of the tasks, which is not fully consistent with the results obtained in this paper.

Table 6 shows the computational speed of the learning for one network. BIP takes only a third of the computational time, while the read-out learning is computationally more costly. The BIP algorithm only needs to be performed only once for each network and data set, while the read-out learning needs to

Task	RELM	EI-ELM	I-ELM	CI-ELM
Ab	0.076 ± 0.006	0.082 ± 0.002	0.092 ± 0.005	0.083 ± 0.003
Ca	0.117 ± 0.003	0.149 ± 0.002	0.168 ± 0.005	0.155 ± 0.005
Ce	0.036 ± 0.002	0.083 ± 0.001	0.092 ± 0.867	0.087 ± 0.002
De	0.018 ± 0.003	0.058 ± 0.003	0.074 ± 0.013	0.060 ± 0.007
Co	0.019 ± 0.003	0.094 ± 0.003	0.120 ± 0.013	-

Table 5: Mean performance and standard deviation of other state of the art methods on the respective regression tasks used to benchmark BIP pretrained ELM with RR.

be recomputed when a different regularization parameter is used. BIP in its present form is computationally efficient and does not slow down the learning significantly when compared to the improvement of performance.

Task	BIP	RR	sum (s)
Ab	3.0222 (27%)	8.2164 (73%)	11.2386
Ca	7.7488 (29%)	19.264 (71%)	27.0128
Ce	9.4519 (32%)	19.909 (68%)	29.3609
De	4.7233 (30%)	10.903 (70%)	15.6263
Co	4.5979 (26%)	12.907 (74%)	17.5049
El	2.1856 (17%)	10.454 (83%)	12.6396

Table 6: Computational time consumption for the BIP-pretraining (second column), the readout learning via RR (third column), and the sum of both. The values are given in seconds. The percentage indicates how the computational time is divided by the learning methods.

The results demonstrate that the best performance of the BIP networks additionally trained with RR in comparison to the randomly initialized networks is not only due to RR or BIP. The combination of RR and BIP leads to significantly better performing networks with a low variance in the test error for all tasks used in the experiments. In addition, the BIP trained networks have higher optimal RR parameters. This is due to the fact that the features provided by the hidden layer are more task suited. Less large output weights are needed to produce a good mapping. This is desired cause it makes the networks robust against noise in the hidden layer state.

5. Conclusion

This contribution proposes to use BIP as optimization method for ELMs with a fixed hidden layer size and especially motivates the use of shaping output distributions for hidden layer neurons in an ELM. It is further shown that the additional use of output regularization can lead to good and stable performance over different network initializations. The method is compared to other ELM techniques and the timing of the algorithm is analyzed for real world regression tasks.

BIP is a recently developed and unsupervised scheme to pretrain ELMs. The method is efficient and can therefore be used to initialize the networks input weights and biases without detailed knowledge about the task. It was already shown that the production of sparse codes in the hidden layer of a neural network by producing exponential distributions with low fixed mean can lead to good generalization results [20]. This was done for recurrent neural networks by IP which was introduced by Triesch in [26]. IP is based on computationally expensive gradient descent and therefore unsuited for ELMs. BIP overcomes the problem of slow learning dynamics by formulating a regression problem. The experiments show that the new learning method produces the desired output distributions to some extend and leads to an improvement of the learning for ELMs which are additionally regularized by ridge regression. Especially networks with a large hidden layer have a better generalization ability when pretrained with BIP.

Since every hidden layer neuron is updated separately, BIP is well suited for methods optimizing ELMs by incrementally growing the network without recomputation of the already obtained biases and slopes. BIP complements those methods and could - combined with other optimization methods - lead to even better learning results for ELMs. In addition, every neuron can be trained by a different desired output distribution in order to produce a more diverse encoding in the hidden layer.

Only the desired distribution f_{des} and the inverse of the activation f^{-1} is needed for the method, which points out the high flexibility of the method. The generic formulation might be used to analyze the performance of the method with respect to other desired output distributions and activation functions. This will lead to different codes in the hidden layer and has a huge impact on the network's performance.

[1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings, IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985 – 990, 2004.

[2] G.-B. Huang, D. Wang, and Y. Lan, "Extreme learning machine: A survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[3] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[4] D. S. Broomhead and D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks," *Complex Systems 2*, pp. 321–355, 1988.

[5] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, pp. 576–583, 2008.

[6] L. Wang and C. Wan, "Comments on the extreme learning machine," *Neural Networks, IEEE Transactions on*, vol. 19, no. 8, pp. 1494 –1495, 2008.

[7] G.-B. Huang, "Reply to comments on the extreme learning machine," *Neural Networks, IEEE Transactions on*, vol. 19, no. 8, pp. 1495 –1496, 2008.

[8] 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.

[9] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression ans multi-class classification," *accepted by IEEE Transactions on Systems, Man, and Cybernetics: Part B*, 2011.

[10] V. N. Vapnik, *The nature of statistical learning theory.* New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[11] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[12] Y. Miche, A. Sorjamaa, and A. Lendasse, "OP-ELM: Theory, experiments and a toolbox," in *Artificial Neural Networks - ICANN 2008*, vol. 5163, pp. 145–154, 2008.

[13] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *Transactions on Neural Networks*, vol. 20, pp. 1352–1357, 2009.

[14] W. Deng, Q. Zheng, and L. Chen, "Regularized extreme learning machine," in *IEEE Symposium on Computational Intelligence and Data Mining*, pp. 389–395, 2009.

[15] Y. Miche, M. van Heeswijk, P. Bas, O. Simula, and A. Lendasse, "Trop-elm: A double-regularized elm using lars and tikhonov regularization," *Neurocomputing*, vol. 74, no. 16, pp. 2413–2421, 2011.

[16] A. N. Tikhonov and V. Y. Arsenin, "Solutions of ill-posed problems.," *soviet Math. Dokl.*, no. 4, pp. 1035–1038, 1963.

[17] X. Dutoit, B. Schrauwen, J. M. V. Campenhout, D. Stroobandt, H. V. Brussel, and M. Nuttin, "Pruning and regularisation in reservoir computing: a first insight," in *ESANN*, pp. 1–6, 2008.

[18] C. M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Computation*, vol. 7, no. 1, pp. 108–116, 1995.

[19] J. Triesch, "A gradient rule for the plasticity of a neuron's intrinsic excitability," in *Proc. ICANN*, pp. 65–79, 2005.

[20] J. J. Steil, “Online reservoir adaptation by intrinsic plasticity for backpropagation decorrelation and echo state learning,” *Neural Networks, Special Issue on Echo State and Liquid State networks*, pp. 353–364, 2007.

[21] K. Neumann and J. J. Steil, “Batch intrinsic plasticity for extreme learning machines,” in *Proceedings, International Conference on Artificial Neural Networks*, pp. 339–346, 2011.

[22] S. Kullback and A. Leibler, “On information and sufficiency,” *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[23] R. Penrose, “A generalized inverse for matrices,” in *Mathematical Proceedings of the Cambridge Philosophical Society*, pp. 406–413, 1955.

[24] A. Frank and A. Asuncion, “UCI machine learning repository,” 2010.

[25] G.-B. Huang and L. Chen, “Convex incremental extreme learning machine,” *Neurocomputing*, vol. 70, no. 16-18, pp. 3056–3062, 2007.

[26] J. Triesch, “Synergies beween intrinsic and synaptic plasticity in individual model neurons,” in *NIPS*, 2005.