Batch Intrinsic Plasticity for Extreme Learning Machines

Klaus Neumann and Jochen J. Steil

Research Institute for Cognition and Robotics (CoR-Lab)
Faculty of Technology, Bielefeld University
Universitätsstr. 25, 33615 Bielefeld, Germany
{kneumann,jsteil}@cor-lab.uni-bielefeld.de,www.cor-lab.de

Abstract Extreme learning machines are 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. The success of learning strongly depends on the random parameter initialization. To overcome the problem of unsuited initialization ranges, a novel and efficient pretraining method to adapt extreme learning machines task-specific is presented. The pretraining aims at desired output distributions of the hidden neurons. It leads to better performance and less dependence on the size of the hidden layer.

Keywords: extreme learning machine, pretraining, neural network, learning, intrinsic plasticity, batch, regression

1 Introduction

In [1], Huang proposes the extreme learning machine (ELM) which is an efficient learning algorithm based on random projections. Its task performance depends on the size of the hidden layer and the initialization ranges of the parameters. A good performance is usually achieved by manually tuning these parameters to a task-suitable regime.

Although, recently some improvements to the ELM have been developed, that are based on the idea to change the hidden layer size, an automatic and efficient task-specific optimization method for ELMs is still missing.

Feng presents a method which adds random neurons to the ELM - the error minimized extreme learning machine (EMELM) [2]. Whereas recomputation of the pseudo inverse is necessary, the computational time for solving the regression task is reduced to a minimum by using fast update rules derived in the original paper. Another idea to improve ELMs is to decrease the size of the hidden layer - the optimally pruned extreme learning machine (OPELM) [3]. The OPELM method starts with a large hidden layer and a ranking of the neurons. The learning results are improved by pruning the OPELM using a leave-one-out criterion. There is no need to specify the size of the hidden layer in advance without knowledge of the task complexity by using these methods. However, the results still strongly depend on the random initialization - i.e. the biases and

input weights. Methods controlling the network size are insufficient in tuning the neurons to a good regime, where the encoding is optimal.

It is shown in [4], that a biologically inspired online learning rule called intrinsic plasticity (IP) published by Triesch in [5] is able to enhance the encoding in recurrent neural networks. The output is forced by IP to produce exponential distributions. This maximizes the network's information transmission, caused by the high entropy of the distribution. Inspired by IP, we propose a novel method to pretrain ELMs, which also aims on achieving desired output distributions. In contrast to IP, the pretraining works in batch fashion by creating imaginary targets and will therefore be called batch intrinsic plasticity (BIP). The method adapts the hidden layer analytically by a pseudo inverse technique instead of performing a computationally expensive gradient-descent. This idea makes BIP highly efficient.

The following experiments show that the new method leads to better results for randomly initialized ELMs. In particular the generalization ability of the networks is improved significantly.

2 Extreme Learning Machine

The ELM consists of three different layers: $u \in \mathbb{R}^{I\times 1}$ collects the input, $h \in \mathbb{R}^{R\times 1}$ the hidden, and $\hat{y} \in \mathbb{R}^{O\times 1}$ the output neurons. The input is connected to the hidden layer through the input matrix $W^{\text{in}} \in \mathbb{R}^{R\times I}$, while the read-out matrix $W^{\text{out}} \in \mathbb{R}^{O\times R}$ contains the read-out weights. The ELM as it is proposed by Huang 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. When denoting the weights from the input layer to a specific hidden layer neuron i with $W_i^{\text{in}} \in \mathbb{R}^{1\times I}$, the ELM scheme then becomes

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

2.1 Supervised Read-Out Learning by Ridge Regression

Supervised learning for ELMs is restricted to the read-out weights W^{out}. In order to infer a desired input-output mapping from a set of N_{tr} training samples $(u(k), y(k))$ with $k = 1 \dots N_{\text{tr}}$, the read-out weights W^{out} are adapted such that the mean square error for the training set is minimized:

$$E = \frac{1}{N_{\text{tr}}} \sum_{k=1}^{N_{\text{tr}}} ||y(k) - \hat{y}(k)||^2 \rightarrow \min \quad . \tag{2}$$

The paper focuses on batch training and uses a standard linear ridge regression method to control the size of the output weights. This is different to the approach in the original ELM paper where the pseudo inverse is used. The generalization ability of the networks is improved by that technique. The network's states $h(k)$ belonging to the inputs $u(k)$ as well as the desired output targets $y(k)$

are collected in a state matrix $H = (h(1) \dots h(N_{\text{tr}}))^T \in \mathbb{R}^{N_{\text{tr}} \times R}$ and a target matrix $Y = (y(1) \dots y(N_{\text{tr}}))^T \in \mathbb{R}^{N_{\text{tr}} \times O}$. The optimal read-out weights are then determined by the least squares solution

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

where the factor $\varepsilon \geq 0$ was identified by Tikhonov in [6] as output regularization strength.

2.2 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, almost linear or constant neurons. This can be avoided by finding activation functions which are in a favorable regime. Thus, we introduce a novel method to adapt activation functions such that certain output distributions are achieved. An invertible activation function and a random number generator which produces numbers drawn from the desired distribution are assumed.

Only the inputs $u = (u(1), u(2) \dots u(N_{\text{tr}})) \in \mathbb{R}^{I \times N_{\text{tr}}}$ stimulating the network 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. The synaptic sum arriving at neuron i is given by $s_i(k) = W_i^{\text{in}} u(k)$ and collected in $s_i = W_i^{\text{in}} u$.

Therefore, a linear regression problem is formulated, where random targets $t = (t_1, t_2 \dots t_{N_{\text{tr}}})^T$ are drawn in ascending order $t_1 < \dots < t_{N_{\text{tr}}}$ from the desired output distribution. Since the stimuli need to be mapped onto the right targets, a rearrangement of the stimuli in ascending order $s_i(1) < \dots < s_i(N_{\text{tr}})$ is done by sorting $s_i \leftarrow \text{sort}(s_i)$. 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. The model $\Phi(s_i) = \left(s_i^T, (1 \dots 1)^T\right)$ and the parameter vector $v_i = (a_i, b_i)^T$ are built to reduce the learning for the i-th neuron to a linear and over-determined regression problem, where the outputs are mapped onto the targets $h_i(k) \approx t_k$:

$$\|\Phi(s_i) \cdot v_i - f^{-1}(t)\| \rightarrow \min \quad . \tag{4}$$

The solution for the optimal slope a_i and bias b_i is obtained by computation of the Moore-Penrose pseudo inverse [7]:

$$v_i = (a_i, b_i)^T = \Phi^{\dagger}(s_i) \cdot f^{-1}(t) \quad . \tag{5}$$

Typically Fermi and tangens hyperbolicus functions are used as activation functions. The learning is done in one-shot fashion and summarized in Alg. 1.

The pretraining is of the same order of complexity than the supervised read-out learning, since only the least squares solutions of the linear model Φ have to be calculated. In the experiments, the pretraining and the supervised learning showed no significant difference in the computational time.

Algorithm 1 batch intrinsic plasticity (BIP)

Require: get inputs $u = (u(1), u(2) \dots u(N_{\text{tr}}))^T$
  for all hidden neurons i **do**
    get stimuli $s_i = W_i^{\text{in}} \cdot u$
    draw targets $t = (t_1, t_2 \dots t_{N_{\text{tr}}})^T$ from desired distribution f_{des}
    sort targets $t \leftarrow \text{sort}(t)$ and stimuli $s_i \leftarrow \text{sort}(s_i)$
    build $\Phi(s) = \left(s_i^T, (1 \dots 1)^T\right)$
    calculate (pseudo-)inverse $(a_i, b_i)^T = v_i = \Phi(s_i)^\dagger \cdot f^{-1}(t)$
  end for
  return $v = (v_1, v_2 \dots v_R)^T$

3 Results

In Sect. 3.1 the impact of BIP-learning is considered and single-neuron behavior is illustrated for different input and desired output distributions. Sect. 3.2 demonstrates the performance of the ELMs after pretraining on a robotics task. Sect. 3.3 shows that the performance is less dependent on the size of the hidden layer after pretraining the ELMs with BIP on the Abalone task from the UCI machine learning repository [8] and compares the method to other state of the art models.

3.1 Batch Intrinsic Plasticity and Single Neuron Behavior

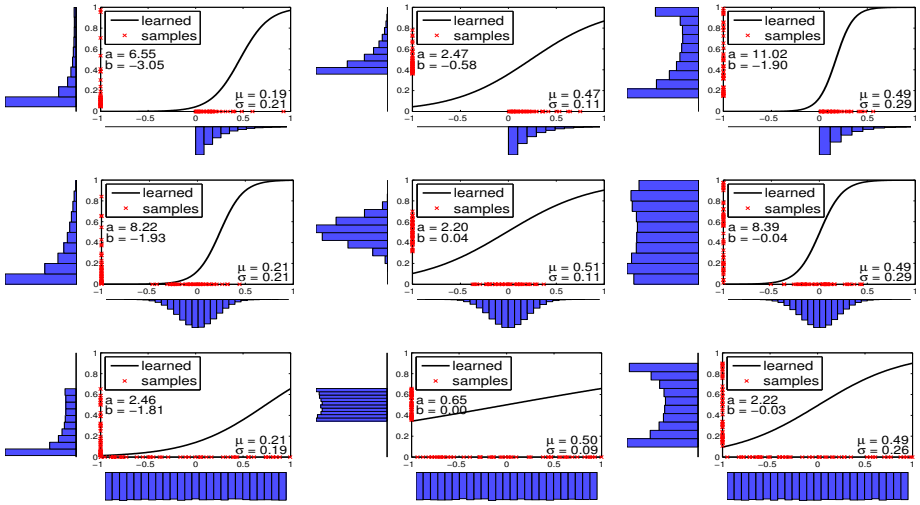

Figure 1. A neuron's activation function adapted by BIP to approximate the output distributions f_{des} while starting from the input distributions f_{s}. The input distribution is varied over the rows, while the output distributions varies column-wise.

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 are used for testing - both drawn from f_{s}.

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

Fig. 1 illustrates the result of adapting the neuron's nonlinear transfer function. The input distribution is assigned to the rows of the figure, while the desired output distribution is assigned column-wise. The incoming training stimuli are visualized by the crosses on the x-axis, while the corresponding targets are 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. 1. It is demonstrated in the first column of Fig. 1 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.

Table 1. Fits of output distributions. A cell contains mean and standard deviation of the χ^2-value, μ and σ.

$\chi^2/\mu/\sigma$	exp	norm	uni
exp	0.49±0.36	1.04±1.04	1.83±0.37
	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	0.08±0.06	0.05±0.04	0.27±0.11
	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	0.27±0.11	0.25±0.09	1.14±0.13
	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 2. Test errors on the robotics task. Comparison of randomly initialized and BIP-pretrained ELMs.

rnd BIP	ld(ε)=-15	−12	−9
R=50	.062±.003	.062±.003	.060±.002
	.062±.004	.063±.004	.059±.002
100	.094±.034	.093±.032	.077±.017
	.073±.014	.072±.013	.061±.002
150	.149±.076	.148±.076	.107±.042
	.073±.013	.073±.013	.062±.003
200	.229±.160	.227±.158	.153±.085
	.075±.015	.075±.015	.062±.003

Further statistics are summarized in Tab. 1. The table shows a neuron which is trained by BIP for 100 trials. After each trial, the mean and the standard deviation of the output distribution are collected as well as the χ^2-value over 100 trials which determines the deviation of samples from the desired probability distribution. The χ^2-value is given by $\chi^2 = \sum_{i=1}^{\#\text{bins}} \frac{(O_i - E_i)^2}{E_i}$, where $\#\text{bins} = 20$ is the number of bins equidistantly distributed in the interval $[0, 1]$. E_i is the analytically given value of the integral in the i-th bin-range, and O_i is the observed value divided by the number of test samples $N_{\text{te}} = 1000$. The table

shows, that μ and σ of the output distribution are always approximated very well with low variance.

3.2 Robotics Regression Task

In the following two sections the experiments are described, where the networks' input matrix components W_{ij}^{in} and the biases b_i are drawn from a uniform distribution in the interval $[-10, 10]$ while the slopes a_i are set to unity. In the experiments, the Fermi-function $f(x) = 1/(1 + \exp(-x))$ is used as activation function and the desired output is the exponential distribution $f_{\text{des}} = f_{\text{exp}}$ with a fixed mean $\mu = 0.2$. It was already shown that this choice of desired output distribution can lead to an improvement of the generalization ability [4].

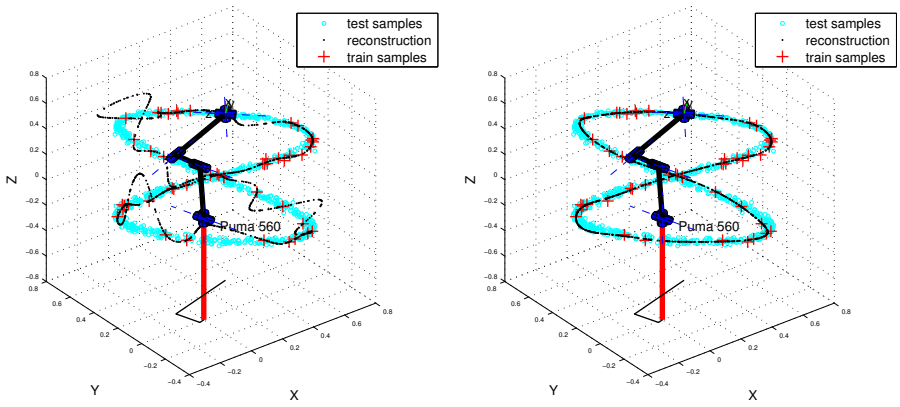

Figure 2. Robotics task for ELMs with: $R = 150$ and $\varepsilon = 10^{-12}$. Left: performance of the randomly initialized ELM. Right: performance of the ELM which was first trained with the BIP method.

The network models are applied 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. 2. $N_{\text{tr}} = 100$ training 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 $F : \mathbb{U} \to \mathbb{Y}$, where $\mathbb{U}$ is the task and $\mathbb{Y}$ the joint space. For each task space input $(u_1(k) \dots u_6(k))^T$ containing the end-effector position and orientation the six-dim target vector $(y_1(k) \dots y_6(k))^T$ is computed and additionally corrupted with Gaussian-noise ($\sigma_N = 0.1$). The generated trajectory forms an eight - see Fig. 2. The left plot images the learned inverse kinematics for a randomly initialized ELM, which apparently overfits the data. The right plot shows the result of the supervised learning for an ELM which was first trained with BIP. The learned part of the inverse kinematics is approximated very well.

Additionally, $N_{\text{te}} = 1000$ test samples are created to verify the generalization capability for different hidden layer sizes R and output regularization strengths ε. The results of the experiments are summarized in Tab. 2 and done for 10 different ELMs and 10 different data sets for each cell. The results show that the ELMs trained with BIP perform significantly better than the randomly initialized networks over the whole range of the parameters. Even ELMs with a big hidden layer and low output regularization (e.g. with $R = 200$, $\varepsilon = 10^{-15}$) do not tend to overfit the data after BIP-pretraining. Also the variance in the performance is much less after pretraining, a robust solution from the learning can be guaranteed.

3.3 Abalone Regression Task

In this section, the performance is tested on the well known Abalone task comprising $N_{\text{tr}} = 2000$ samples for training and $N_{\text{te}} = 2177$ for testing. The performance results of some popular optimization techniques (resource allocation network (RAN) [9], minimum resource allocation network (MRAN) [10], incremental extreme learning machine (IELM) [11], and error minimized extreme learning machine (EMELM) [2]) on the Abalone regression task quoted from [2] are given in Tab. 4. 20 BIP-pretrained ELMs are used with different hidden layer sizes R, the results are summarized in Tab. 3. The input was normalized to $[-1, 1]$ and the output to $[0, 1]$, the weights were drawn uniformly from $[-1, 1]$ and linear regression where used for supervised learning as it was done in Feng's work to make the results comparable. Since the mentioned models are focusing

Table 3. Test-RMSEs on Abalone task.

R	40	41	42	43	44
mean	.0748	.0754	.0749	**.0745**	.0756
std	.0005	.0014	.0012	**.0004**	.0020
R	45	46	47	48	49
mean	.0751	.0761	.0747	.0745	.0748
std	.0004	.0014	.0008	.0008	.0005

Table 4. Abalone results, [2].

model	EMELM	IELM
mean	**.0755**	.0920
std	**.0032**	.0046
model	RAN	MRAN
mean	.1183	.0906
std	.0076	.0065

on incremental growth of the hidden layer, which is different to the BIP scheme, a direct comparison seems difficult. However, Tab. 3 shows that the ELMs of size $R = [40, 49]$ perform better in most of the cases than the other models without incrementally searching for good performing networks.

4 Conclusion

This contribution introduces BIP, a novel and unsupervised scheme to pretrain ELMs. Since the algorithm works in batch fashion, it is independent of learning dynamics. It was shown that the new learning method produces the desired

output distributions to some extend and leads to an improvement of the learning for randomly initialized ELMs by task-specific pretraining - no excessive expert-tuning is needed anymore. The method is efficient and can therefore be used to initialize the networks input weights and biases without detailed knowledge about the task. In addition, BIP is compared to other optimization techniques and show that it leads to better and stable results for a specific network size.

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.

Most of the methods used for optimizing ELMs - like the ones mentioned - focus on the size of the hidden layer. BIP complements those methods and could - combined with other optimization methods - lead to even better learning results for ELMs.

References

1. Huang, G.-B., Zhu, Q.-Y., Siew C.-K.: Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In International Joint Conference on Neural Networks (IJCNN'2004), Budapest, Hungary, July 2004.
2. Feng, G., Huang, G.-B., Lin, Q., Gay, R.: Error Minimized Extreme Learning Machine with Growth of Hidden Nodes and Incremental Learning. Trans. Neur. Netw., 20:1352–1357, August 2009.
3. Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C., Lendasse, A.: OP-ELM: Optimally Pruned Extreme Learning Machine. Neural Networks, IEEE Transactions on, 21(1):158–162, 2010.
4. Steil, J.J.: Online Reservoir Adaptation by Intrinsic Plasticity for Backpropagation-Decorrelation and Echo State Learning. Neural Networks, Special Issue on Echo State and Liquid State networks, pages 353–364, 2007.
5. Triesch, J.: Synergies beween Intrinsic and Synaptic Plasticity in Individual Model Neurons. In NIPS, 2005.
6. Tikhonov, A.N., Arsenin, V.Y.: Solutions of Ill-Posed Problems. soviet Math. Dokl., (4):1035–1038, 1963.
7. Penrose, R.: A Generalized Inverse for Matrices. In Mathematical Proceedings of the Cambridge Philosophical Society, pages 406–413, 1955.
8. Frank, A., Asuncion, A.: UCI machine learning repository, 2010.
9. Platt, J.: Resource-Allocating Network for Function Interpolation. Neural Computation, 3(2), 1991.
10. Yingwei, L., Sundararajan, N., Saratchandran, P.: A Sequential Learning Scheme for Function Approximation using Minimal Radial Basis Function Neural Networks. Neural Comput., 9:461–478, February 1997.
11. Huang, G.-B., Chen, L., Siew, C.-K.: Universal Approximation using Incremental Constructive Feedforward Networks with Random Hidden Nodes. Neural Networks, IEEE Transactions on, 17(4):879–892, July 2006.