Rank-Revealing Orthogonal Decomposition in Extreme Learning Machine Design

Jacek Kabziński(✉)

Lodz University of Technology, Stefanowskiego 18/22, Lodz, Poland
jacek.kabzinski@p.lodz.pl

Abstract. Extreme Learning Machine (ELM), a neural network technique used for regression problems, may be considered as a nonlinear transformation (from the training input domain into the output space of hidden neurons) which provides the basis for linear mean square (LMS) regression problem. The conditioning of this problem is the important factor influencing ELM implementation and accuracy. It is demonstrated that rank-revealing orthogonal decomposition techniques can be used to identify neurons causing collinearity among LMS regression basis. Such neurons may be eliminated or modified to increase the numerical rank of the matrix which is pseudo-inverted while solving LMS regression.

Keywords: Neural networks modelling · Extreme learning machine
Nonlinear systems

1 Introduction

An Extreme Learning Machine (ELM) [1, 2] – a neural network with one fixed hidden layer and adjustable output weights - is able to solve complicated regression or classification problems. In this paper, application of ELMs for modeling multivariable, nonlinear functions with batch data processing is considered. The main ideas behind the standard ELM approach are that: the weights and biases of the hidden nodes are generated at random, without 'seeing the data', and are not adjusted, so 'training' means that the output weights are determined analytically, solving a linear mean square (LMS) problem. Therefore, the training is reduced to one step and the training time is very short comparing to iterative training.

The numerical round-off errors of linear mean square regression are the main reasons for ELMs' modeling errors and are strictly connected with the number of neurons in the hidden layer. When the number of hidden layer nodes is small, the ELM may not be able to transform the input into the feature space effectively and the approximation error may be unacceptably large. When the number of hidden layer nodes is large, it increases the computation complexity, may lead to an ill-conditioned LMS regression problem and may even result in overfitting of the ELM. The necessity of improving the numerical properties of ELM was noticed in several recent publications [3–6]. Neurons pruning techniques were proposed in [7, 8] and incremental learning was used in [9, 10]. Both methods try to get the optimal number of hidden layer nodes. But, with every change of hidden layer nodes, the output weights need to

V. Kůrková et al. (Eds.): ICANN 2018, LNCS 11140, pp. 3–13, 2018.
https://doi.org/10.1007/978-3-030-01421-6_1

be recalculated, so these techniques considerably increase the computation complexity of ELM. In [11], the method called orthogonal projections to latent structures, which is a combination of orthogonal signal correction and partial least squares, is proposed, but it still leads to a tedious iterative procedure. Complicated methods of probability distribution optimization are proposed in [12].

The main contribution of this paper is to show that rank-revealing transformations, known since the previous century, are effective tools to indicate neurons responsible for numerical collinearity among the LSM regression basis. Such "non-contributing" neurons may be eliminated or modified so that they are useful in the approximation. In any case, the final basis for LMS regression is orthogonal and the output weights may be obtained by solving well-conditioned LMS problem.

The standard ELM is described in Sect. 2. Instead of the most popular random generation of weights, the application of low discrepancy sequences (LDS) [13, 14] is considered. Rank-Revealing Orthogonal Decomposition is introduced and applied in Sect. 3, while the proposed neuron modification procedure is presented in Sect. 4. The paper ends with numerical experiments and conclusions.

2 Basic Extreme Learning Machine

The standard Extreme Learning Machine applied for modelling (regression) problems may be considered as a combination of a nonlinear mapping from the input space into the feature space and a linear least-squares regression. The training data for a n-input ELM form a batch of N samples:

$$\{(x_i, t_i), \quad x_i \in R^n,\ t_i \in R,\ i = 1, \ldots, N\}, \tag{1}$$

where x_i denote the inputs and t_i denote the desired outputs, which form the target (column) vector $T = \begin{bmatrix} t_1 & \cdots & t_N \end{bmatrix}^T$. It is assumed that each input is normalized to the interval [0,1].

The nonlinear mapping is performed by a single layer of hidden neurons with infinitely differentiable activation functions. The "projection-based neurons" are used most commonly. Each n-dimensional input is projected by the input layer weights $w_k^T = [w_{k,1} \ldots w_{k,n}], \quad k = 1, \ldots M$ and the bias b_k into the k-th hidden neuron input. Next, a nonlinear transformation h_k, called activation function (AF) of the neuron is applied to obtain the neuron output. The transformation of a batch of N samples by the hidden layer is represented as an $N \times M$ matrix:

$$H = H_{N\times M} = \left[h_i\left(w_i^T x_j + b_i\right)\right]_{\substack{j = 1, \ldots, N \\ i = 1, .., M}}. \tag{2}$$

It is assumed that the number of samples is greater than the number of neurons: $N > M$. The impact of the selected type of AFs on the network performance is limited, and therefore sigmoid AFs remain among the most widely used.

According to the standard approach, the weights and biases are generated at random, using any continuous probability distribution [2]. Using the uniform distribution in $[-1, 1]$ to generate the weights and the biases is the standard procedure. Recently, an application of Low-Discrepancy Sequences (LDS) [13, 14] was proposed to replace the random generation of neurons' parameters. The discrepancy measures the uniformity of a sequence X of N points in the hypercube $P = [0, 1]^n$, and is defined as

$$D_N(X) = \sup_B \left| \frac{N_o(X, B)}{N} - L(B) \right|, \tag{3}$$

where B is any hypercube $[a_1, b_1] \times \ldots \times [a_n, b_n] \subset P$, $N_o(X, B)$ denotes the number of points from X belonging to B and $L(B)$ is the Lebesgue measure (volume) of B [13]. So, low discrepancy means that the number of points in a subset is as proportional as possible to the volume. Numerical procedures to generate various LDSs are offered by popular software packages. For example, easy generation of Halton and Sobol sequences [13, 14] is possible in Matlab. The distance among any LDS and a random set tends to zero if the number of points increases (Fig. 1), so the universal approximation property of a standard (randomly-generated) ELM [15] is generalized to the deterministic case with weights and biases taken from an LDS [16] (Fig. 2).

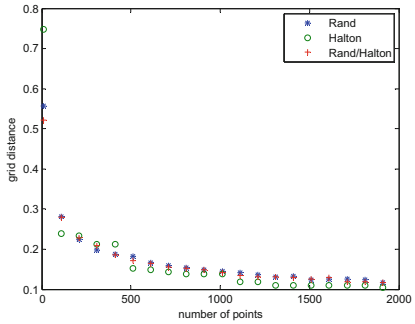

Fig. 1. Maximal distance (mean in 20 experiments) to the nearest neighbour: * - inside the random set, o – inside the Halton set, + from the Halton set to the random set. Points are generated in the 3-dimensional cube.

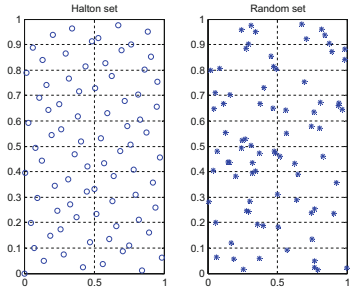

Fig. 2. 80 points generated from Halton sequence and from uniform distribution in 2 dimensions.

Hence, deterministic creation of neurons' weights and biases from an LDS is an interesting alternative and allows to describe features of an ELM without repeating numerous experiments.

The output weights β are found by minimizing the approximation error

$$E_C = \|\beta\|^2 + C\|H\beta - T\|^2, \tag{4}$$

where $C > 0$ is a design parameter added to improve the conditioning of the problem. This approach is called 'Tikhonow regularization' [17]. For $C \to \infty$ the problem becomes equivalent to the minimization of

$$E_\infty = \|H\beta - T\|^2. \tag{5}$$

The output weights, which minimize the regularized criterion (4) are

$$\beta_{Copt} = \left(\frac{1}{C}I + H^T H\right)^{-1} H^T T, \tag{6}$$

while (5) is minimized by

$$\beta_{opt} = H^+ T, \tag{7}$$

where H^+ is the Moore–Penrose generalized inverse of matrix H:

$$H^+ = \left(H^T H\right)^{-1} H^T. \tag{8}$$

3 ELM with Rank-Revealing Orthogonal Decomposition

When a large number of hidden layer neurons is selected, high correlations and multicollinearity always exist among the columns of the hidden layer output matrix H. It may lead to ill-condition of the Moore–Penrose calculation or cause overfitting of the final model. The high condition number of $H^T H$ is the main reason of numerical difficulties in ELM implementation. The Tikhonov regularization is supposed to improve this situation - the coefficient C is selected to decrease the condition number of $\frac{1}{C}I + H^T H$, but unavoidably degrades the approximation accuracy.

The "numerical rank" of a matrix is defined as the number of singular values larger than a certain threshold r.

Rank-Revealing Orthogonal Decomposition, introduced in [18, 19] allows to eliminate multicollinearity among columns of H. Rank-revealing decomposition provides information about the numerical rank of the matrix. For any numerical rank threshold r, the algorithm called RRQR (Rank-Revealing Q-R factorization) allows to represent the column-permuted matrix H as

$$H\begin{bmatrix} P_1 & P_2 \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 & Q_3 \end{bmatrix} \begin{bmatrix} R_1 & R_2 \\ 0 & R_3 \\ 0 & 0 \end{bmatrix}, \tag{9}$$

where P_1, P_2 are permutation matrices, $Q = \begin{bmatrix} Q_1 & Q_2 & Q_3 \end{bmatrix}$ is an orthogonal matrix, R_1, R_3 are upper-triangular matrices and R_1 is a full-numerical-rank matrix with respect

to the threshold r, while maximal singular value of R_3 is not bigger than r. Therefore, after calculation of the rank-revealing QR factorization, the orthogonal matrix

$$Q_1 = HP_1R_1^{-1} \tag{10}$$

may be used to replace H. The multiplication by the permutation matrix P_1 represents selection of the neurons that contribute to the numerical rank. Then, the multiplication by R_1^{-1} provides normalization such that $cond\, Q_1^T Q_1 = 1$. Finally, the optimal output weights are obtained from

$$\beta_{opt} = \left(Q_1^T Q_1\right)^{-1} Q_1^T T. \tag{11}$$

Some neurons are eliminated permanently from the initial set of neurons, hence, the final number of neurons may be smaller than the initially planned. Therefore, the effort to select parameters of excluded neurons is spoiled, but all remaining neurons contribute to the effective approximation. The final form of the network is presented in Fig. 3.

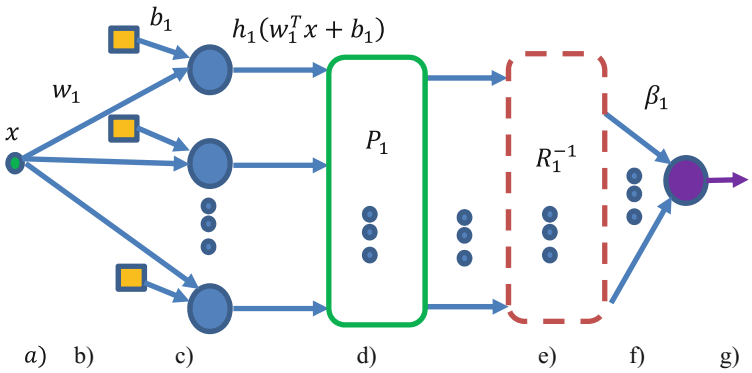

Fig. 3. The modified ELM: (a) input, (b) weights and biases, (c) hidden neurons, (d) elimination of neurons, (e) normalization, (f) output weights, (g) output.

The applied algorithms (RRQR and triangular matrix inversion) are available in various software packages. The computational complexity of the proposed modification is $O(M^3)$. All operations are done "without seeing the data" and the final training of the network (calculation of β_{opt}) is done in one step. The only additional parameter is the numerical rank threshold r.

The proposed procedure may lead to stagnation of the number of neurons, limited by the numerical rank condition, in spite of the user's plan to use more neurons. Therefore RRQR factorization may be used to recognize "non-contributing" neurons. Such neurons are indicated by the permutation matrix P_2 and their parameters must be modified. The modification is discussed in the next section, and the pseudo-code for the complete design procedure is presented in Fig. 4.

1. Select weights and biases for M hidden neurons.
2. Select the numerical rank threshold r.
3. Calculate the matrix H using neurons parameters and input samples.
4. While LoopCounter < Max do
   4.1 Perform the RRQR factorization of the matrix H with the numerical rank threshold r.
   4.2 Use the permutation matrix P_2 to recognize the non-contributing neurons.
   4.3 Modify the weights and biases of the non-contributing neurons.
   4.4 Calculate the new matrix H using new neurons' parameters and input samples.
   4.5 Increment the LoopCounter and go to 4.1.
5. Perform the RRQR factorization of the matrix H with the numerical rank threshold r.
6. Eliminate the non-contributing neurons and calculate the optimal output weights from (11).

Fig. 4. The pseudo-code for the modified ELM

4 Modification of Non-contributing Neurons

According to step 4.3 of the design procedure presented in Fig. 4, weights and biases of the selected neurons need to be modified. The aim of this modification is to change columns of the matrix H which do not contribute to the numerical rank for the given threshold, i.e. the columns in HP_2. The modification has to preserve the nature of weights selection – at random, using a continuous probability distribution, or from an LDS, in a compact hypercube. Several approaches are possible, but it is well-known that multicollinearity of columns in the matrix H may be caused by an insufficient variance of the AFs. The easy way to enhance the variance of sigmoid AFs was proposed in [4–6] and may be applied to modify the weights and biases of the non-contributing neurons.

The first step to enlarge variation of sigmoid activation functions is to increase the range of weights. The weights must be large enough to expose the nonlinearity of the sigmoid AF, and small enough to prevent saturation. Therefore, the already selected weights of non-contributing neurons will be multiplied by a random factor taken from the interval $[q, p]$. The values $[q, p] = [3, 10]$ are suitable.

Next, the biases are selected to guarantee that the range of a sigmoid function is sufficiently large. The minimal value of the sigmoid function

$$h_k(x) = \frac{1}{1 + exp\left(-\left(w_k^T x + b_k\right)\right)}, x \in [0, 1]^n, \tag{12}$$

is achieved at the vertex selected according to the following rules:

$$w_{k,i} > 0 \Rightarrow x_i = 0, w_{k,i} < 0 \Rightarrow x_i = 1 \; i = 1, \ldots, n, \tag{13}$$

and equals

$$h_{k,min} = \frac{1}{1 + exp\left(-\left(\sum_{i:w_{k,i}<0} w_{k,i} + b_k\right)\right)}. \tag{14}$$

The maximal value is attained at the vertex defined by

$$w_{k,i} > 0 \Rightarrow x_i = 1, w_{k,i} < 0 \Rightarrow x_i = 0 \; i = 1, \ldots, n, \tag{15}$$

and is

$$h_{k,max} = \frac{1}{1 + exp\left(-\left(\sum_{i:w_{k,i}>0} w_{k,i} + b_k\right)\right)}. \tag{16}$$

Therefore, to get $h_{k,min} < r_1, h_{k,max} > r_2$ for given $0 < r_1 < r_2 < 1$ requires to have

$$\bar{b} := -\sum_{i:w_{k,i}>0} w_{k,i} - ln\left(\frac{1}{r_2} - 1\right) < b_k < -ln\left(\frac{1}{r_1} - 1\right) - \sum_{i:w_{k,i}<0} w_{k,i} := \tilde{b}. \tag{17}$$

As the initial bias $b_{k\,old}$ was selected from the interval $[-1,1]$, it is modified according to the linear transformation

$$b_{k\,new} = \frac{1}{2}(\tilde{b} - \bar{b}) b_{k\,old} + \frac{1}{2}(\tilde{b} + \bar{b}), \tag{18}$$

providing the chance for $\bar{b} < b_{k\,new} < \tilde{b}$.

5 Numerical Examples

The two-dimensional function

$$z = \sin(2\pi(x_1 + x_2), x_1, x_2 \in [0, 1] \tag{19}$$

is considered. 200 samples selected at random constitute the training set, and 100 samples are use as the test set. The surface (18) is plotted in Fig. 5.

In all experiments, the initial values of the hidden layer neurons weights and biases are selected from the Halton sequence. First, only orthogonalization-based elimination of the non-contributing neurons is applied, and this approach (Orthogonalized Extreme Learning Machine - OELM) is compared with the standard ELM. The numerical rank

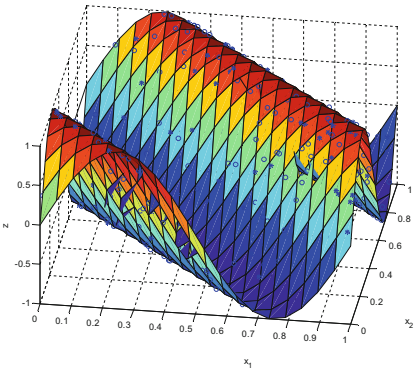

Fig. 5. The surface (18) with with the training (*circles*) and the testing (*stars*) data.

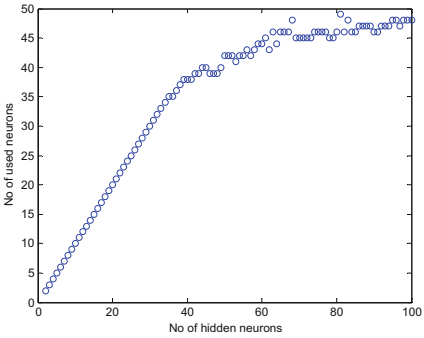

Fig. 6. Reduction of non-contributing neurons.

threshold is 10^{-9}. As it is presented in Fig. 6, the number of the finally used neurons is stabilized below 50, although up to 100 neurons were planned to be used initially. The achieved modeling accuracy is almost the same as obtained with the standard approach with 100 neurons (Fig. 7) and the conditioning of the output weights calculation is far better (Fig. 8).

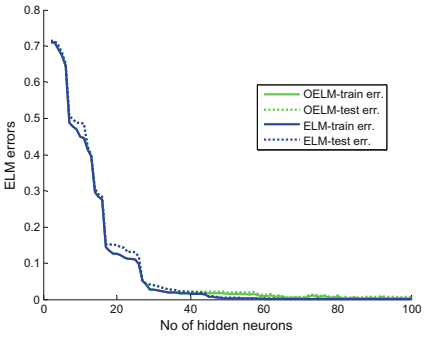

Fig. 7. Training and test errors of the ELM with reduced number of neurons (OLM) and the standard ELM.

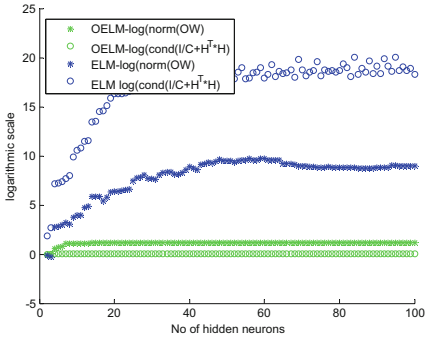

Fig. 8. The condition coefficient and the norm of output weights in OELM and standard ELM.

Of course, the increase of the numerical rank threshold reduces the number of the finally used neurons and the approximation errors increase. If the threshold equals 10^{-3} the number of neurons is reduced below 15 and the errors stabilize at ~ 0.5, which is far too large. Applying the procedure enhancing the variation of the AFs ($r_1 = 0.1$, $r_2 = 0.9$), it is possible to increase the number of finally used neurons and to reduce the approximation errors, preserving numerical rank threshold of 10^{-3}. In Fig. 9 the number of finally used neurons is presented after the first, second and third application of the variation enhancing procedure. In this case, the errors of the modified ELM are smaller

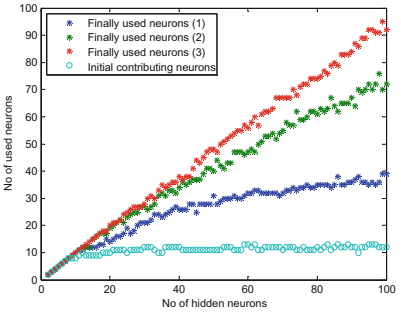

Fig. 9. The number of finally used neurons after the first, second and third application of enhancing variation procedure.

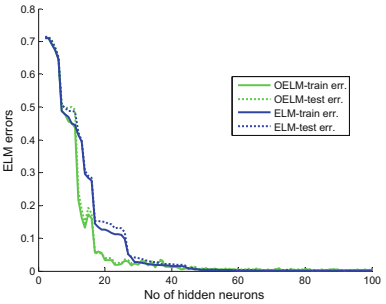

Fig. 10. Training and test errors of the ELM with reduced number of neurons and enhancing variation procedure (OLM) and the standard ELM.

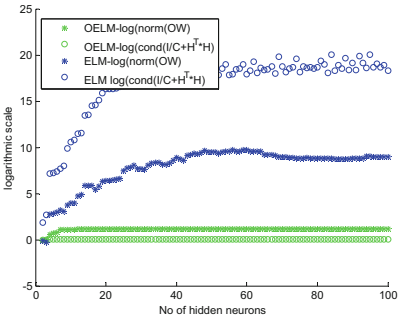

Fig. 11. The condition coefficient and the norm of output weights in ELM with neurons reduction and enhancing variation modification (OELM) compared with the standard ELM.

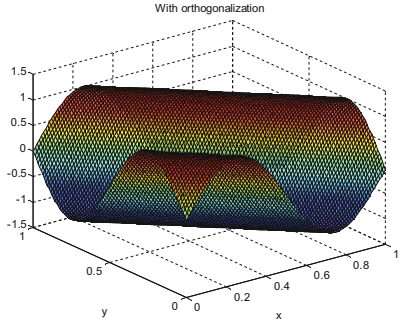

Fig. 12. The surface generated by the trained ELM.

than the standard ELM (Fig. 10), while it is still guaranteed that the condition number equals 1 and the norm of the output weights is minimized (Figs. 11 and 12).

6 Conclusions

The rank-revealing QR decomposition is an effective tool to indicate the neurons in ELM which do not contribute to the effective approximation due to multicollinearity of the columns in matrix H. The indicated neurons may be eliminated and the remaining neurons may be linearly transformed to get the orthogonal basis for the final linear mean square problem, which provides the output weights. If this procedure generates a too small number of neurons to get the desired approximation accuracy, the indicated

non-contributing neurons may be modified to enhance variation of AFs. The approach presented in Sect. 4 re-scales previously chosen weights and biases and increases the number of contributing neurons. The numerical rank threshold is the only additional parameter of the ELM design and it allows to control numerical properties of the network training effectively.

References

1. Huang, G., Huang, G.-B., Song, S., You, K.: Trends in extreme learning machines: a review. Neural Netw. **61**(1), 32–48 (2015)
2. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006)
3. Akusok, A., Bjork, K.M., Miche, Y., Lendasse, A.: High-performance extreme learning machines: a complete toolbox for big data applications. IEEE Access **3**, 1011–1025 (2015)
4. Kabziński, J.: Extreme learning machine with enhanced variation of activation functions. In: IJCCI 2016 - Proceedings of the 8th International Joint Conference on Computational Intelligence, vol. 3, pp. 77–82 (2016)
5. Kabzinski, J.: Extreme learning machine with diversified neurons. In: CINTI 2016 - 17th IEEE International Symposium on Computational Intelligence and Informatics: Proceedings, pp. 181-186 (2016)
6. Kabziński, J.: Is extreme learning machine effective for multisource friction modeling? In: Chbeir, R., Manolopoulos, Y., Maglogiannis, I., Alhajj, R. (eds.) AIAI 2015. IAICT, vol. 458, pp. 318–333. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23868-5_23
7. Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C., Lendasse, A.: OP-ELM: optimally pruned extreme learning machine. IEEE Trans. Neural Netw. **21**(1), 158–162 (2010)
8. Rong, H.J., Ong, Y.S., Tan, A.H., Zhu, Z.X.: A fast pruned-extreme learning machine for classification problem. Neurocomputing **72**(1–3), 359–366 (2008)
9. Huang, G.-B., Chen, L.: Enhanced random search based incremental extreme learning machine. Neurocomputing **71**(16–17), 3460–3468 (2008)
10. Feng, G., Bin Huang, G., Lin, Q., Gay, R.: Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans. Neural Netw., **20**(8), 1352–1357 (2009)
11. Zhang, R., Xu, M., Han, M., Li, H.: Multivariate chaotic time series prediction using based on improved Extreme Learning Machine. In: Proceedings of the 36th Chinese Control Conference, 26–28 July 2017, Dalian, China, pp. 4006–4011 (2017)
12. Han, H., Gan, L., He, L.: Improved variations for Extreme Learning Machine: space embedded ELM and optimal distribution ELM. In: 20th International Conference on Information Fusion, Fusion 2017 - Proceedings, no. 2 (2017)
13. Dick, J., Pillichshammer, F.: Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration. Cambridge University Press (2010)
14. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. SIAM, Philadelphia (1992)
15. Bin Huang, G., Chen, L., Siew, C.K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans. Neural Netw. **17**(4), 879–892 (2006)
16. Cervellera, C., Macciò, D.: Low-discrepancy points for deterministic assignment of hidden weights in extreme learning machines. IEEE Trans. Neural Netw. Learn. Syst. **27**(4), 891–896 (2016)

17. Tikhonov, A.N., Goncharsky, A., Stepanov, V.V., Yagola, A.G.: Numerical Methods for the Solution of Ill-posed Problems. Kluwer Academic Publishers, Dordrecht (1995)
18. Fierro, R.D., Hansen, P.Ch.: Low-rank revealing UTV decompositions. Numer. Algorithms, **15**, 37–55 (1997)
19. Chan, T.F.: Rank revealing QR factorizations. Linear Algebra Appl. **88**(89), 67–82 (1987)