Neurocomputing 599 (2024) 128061

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Insights on the different convergences in Extreme Learning Machine

Davide Elia De Falco [a], Francesco Calabrò [b,*], Monica Pragliola [b]

[a] Scuola Superiore Meridionale, Via Mezzocannone, 4, Naples, I-80138, Italy
[b] Department of Mathematics and Applications "Renato Caccioppoli", University of Naples Federico II, Via Cintia, Monte S. Angelo, Naples, I-80126, Italy

ARTICLE INFO

Communicated by G.-B. Huang

*MSC:*
65D05
65Y20

*Keywords:*
Extreme Learning Machine
Computational costs
Convergence rates
Neural Networks

ABSTRACT

Neural Networks (NN) are a powerful tool in approximation theory because of the existence of Universal Approximation (UA) results. In the last decades, a significant attention has been given to Extreme Learning Machines (ELMs), typically employed for the training of single layer NNs, and for which a UA result can also be proven. In a generic NN, the design of the optimal approximator can be recast as a non-convex optimization problem that turns out to be particularly demanding from the computational viewpoint. However, under the adoption of ELM, the optimization task reduces to a – possibly rectangular – linear problem. In this work, we detail how to design a sequence of ELM networks trained via a target dataset. Different convergence procedures are proposed and tested for some reference datasets constructed to be equivalent to approximation problems.

## 1. Introduction

In the last decades, Neural Networks (NNs) have significantly attracted the interest of researchers. One of the most relevant theoretical aspects related to the analysis of NNs, and motivating their extensive use, is the so-called Universal Approximation (UA) property, which states that NNs with increasing sizes can recover functions in very general settings. A simple construction of NN satisfying the UA property is the case of networks trained via Extreme Learning Machine (ELM) [1–4]. Neural Networks trained via ELM fall in the category of Random Neural Networks [5] and are strongly related to other definitions, including Random Feature Learning [6], Random Neural Networks in Reservoir Systems [7], Echo State Networks [8,9], Random Vector Functional Link Network [10]. ELM networks are very simple to handle, being single-layer random projection networks, and are demonstrated to work very well in different applications, see [11,12]. Among others, we point out that recently ELMs have been profitably applied for the resolution of Partial Differential Equations, both via Physics Informed Neural Networks [13–15] and via collocation [16,17]. In particular, the choices made for the first time in [16] regarding the determination of randomization have inspired the analysis of Section 3.3 in the present work.

Here, we aim to explore the behavior of ELMs when their sizes increase along the computations. In fact, there exist several results stating that in many cases increasing the number of neurons in NNs can improve the reproduction capabilities of the networks; some of the mentioned results are stated in terms of expected convergence rates. For instance, the case of activation functions given by sigmoidal-type functions is covered by the theory of superposed sigmoidal functions, also known as Ridge functions [18]. In this case, it can be proven that the squared approximation errors decrease as $1/N$, as $N$ tends to $+\infty$, $N$ being the number of superposed sigmoidal functions, see [19–24]. However, such results only hold true when all the parameters of the network are fixed in an optimal way, which is not the case of ELM whose design is based on random projections. Interestingly, also the case of random projection networks gives squared approximation errors of the type $1/N$, $N$ being the number of activation functions, see [5–7,25–28]. Moreover, the case of increasing available data is also considered in some references, leading to similar estimates, see ad example [6]. Nevertheless, in practical cases, the observed convergence rates of ELM networks when both the number of activation functions and the number of interpolation sites (thus available data) is increased, are of spectral type thus overperforming the theoretical result, see also [29].

Clearly, the mentioned result is related to the ability of determining optimal choices both for the internal parameters that determinate the activation functions and the weights of the linear combination. Since the training in ELM networks only involves a least square resolution for the external weights, this fails in designing optimal functions, and in general is challenging to provide theoretical results on the convergence rates in the case of specific target functions.

* Corresponding author.
*E-mail addresses:* de.defalco@ssmeridionale.it (D.E. De Falco), francesco.calabro@unina.it (F. Calabrò), monica.pragliola@unina.it (M. Pragliola).

https://doi.org/10.1016/j.neucom.2024.128061
Received 1 August 2023; Received in revised form 15 May 2024; Accepted 12 June 2024
Available online 18 June 2024

The limitations in providing a rigorous analysis of the ELMs convergence rates motivate our interest in exploring the empirical convergence of ELMs in different incremental scenarios, i.e. when the number of neurons and/or the number of available data is increased along the computations.

Although, it is well-established that in many applications – e.g., denoising – regularization is also included in the ELM network so as to improve its performance, here we restrict our attention to non-regularized networks, so as to analyze the convergence of ELMs without the additional problem of determining a suitable value for the regularization parameter, which is a particularly difficult task in general settings [3,30]. Moreover, we fix the case of sigmoidal functions as activation functions in neurons, so as to be as close as possible to the case of previous references. Nevertheless, we point out that UA applies to ELM networks for a much more general class of activation functions [2].

In this work, we provide a uniform framework for the analysis of different configurations for the increase of the size of ELM networks. The considered strategies are based on augmenting, separately and jointly, the number of neurons and the number of data. The computational costs of the different schemes are discussed, and their convergence rates are compared in an extensive computational analysis. Finally, as a way to mitigate the ill-conditioning of the problem solved in the ELM training – that can be considered a natural consequence of the size increase – we propose a novel initialization strategy for the input parameters aimed at defining more informative sigmoidal activation functions.

The paper is organized as follows. In Section 2 we recall some preliminary linear algebra results. The generic extreme learning machine will be introduced in Section 3 together with a novel initialization strategy for the input parameters of the network. In Section 4 we discuss in detail the different scenarios considered in the experimental part, presented in Section 5. We conclude the paper with some outlook for future research in Section 6

## 2. Preliminaries

In this section, we recall some classical linear algebra results – see, e.g., [31] – that will be exploited in Section 4. The first result has to do with the inversion of a block matrix.

**Lemma 1.** *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{C} \in \mathbb{R}^{m\times m}$ be real matrices and consider a block matrix of the form*

$$\mathbf{D} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^{\mathrm{T}} & \mathbf{C} \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$$

*Then,*

$$\mathbf{D}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{Q}\mathbf{P}\mathbf{Q}^{T} & -\mathbf{Q}\boldsymbol{P} \\ -\mathbf{P}\mathbf{Q}^{T} & \mathbf{P} \end{bmatrix}$$

*where*

$$\mathbf{P} = (\mathbf{C} - \mathbf{B}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{B})^{-1} \qquad \in \mathbb{R}^{m\times m}$$
$$\mathbf{Q} = \mathbf{A}^{-1}\mathbf{B} \qquad \in \mathbb{R}^{n\times m}$$

Notice that Lemma 1 turns out to be particularly advantageous when matrix $\mathbf{A}^{-1}$ is already available and $n \gg m$. In this case we only need to invert a $m\times m$ matrix in order to compute the inverse of a $n\times n$ matrix.

The next result is a specific instance of the Woodbury formula.

**Lemma 2.** *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{V} \in \mathbb{R}^{r\times n}$ be real matrices and consider a matrix of the form*

$$\mathbf{D} = \mathbf{A} + \mathbf{V}^{\mathrm{T}}\mathbf{V} \in \mathbb{R}^{n\times n}$$

*Then,*

$$\mathbf{D}^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{V}^{\mathrm{T}}\left(\mathbf{I} + \mathbf{V}\mathbf{A}^{-1}\mathbf{V}^{\mathrm{T}}\right)^{-1}\mathbf{V}\mathbf{A}^{-1} \tag{1}$$

Again, when $\mathbf{A}^{-1}$ is available and $n \gg r$, formula (1) can be computed very efficiently.

## 3. Main results on ELM

In this section, we are going to introduce the generic ELM network and recall some results that motivated interest in the topic. While reporting theoretical results, we emphasize that such results could be reported in a more general framework. As pointed out in the introduction, there are available results following different notations, and the unification of such is out of the scope of the present paper. Then, we are going to outline a simple procedure aimed at mitigating the ill-conditioning of the linear problems arising when training ELM networks.

### 3.1. Definitions and notations

A Single Layer Feedforward Neural Network (SLFN) with $N$ hidden neurons is determined by the input weights $\{\boldsymbol{a}_j \in \mathbb{R}^d\}_{1\le j\le N}$, the hidden neurons' biases $\{b_j \in \mathbb{R}\}_{1\le j\le N}$, and an activation function $g : \mathbb{R} \to \mathbb{R}$; in what follows, we restrict our attention to the case in which $g$ is the sigmoid activation function, that is denoted by $\sigma(\cdot)$.

Given $n$ samples $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, the SLFN aims to approximate the data $y_1, \dots, y_n$ with weighted sums; in formula

$$\sum_{j=1}^{N} w_j\, \sigma(\boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{x}_i + b_j) \approx y_i\,, \quad i = 1\ \dots, n\,,$$

where $\boldsymbol{w} = [w_1, \dots, w_N] \in \mathbb{R}^N$ is referred to as output weights vector. We remark that in general, the scalar input data $y_i$ may be a vector, i.e. $\boldsymbol{y}_i \in \mathbb{R}^m$, $m > 1$. To avoid heavy notations, in our derivations we consider $m = 1$, which, as we will show shortly, does not represent a restrictive hypothesis.

ELM is a training algorithm for SLFNs that computes the output weights vector $\boldsymbol{w}$ as the solution of a linear optimization problem of the form

$$\min_{\boldsymbol{w}\in\mathbb{R}^N} \|\mathbf{H}\boldsymbol{w} - \boldsymbol{y}\|\,,$$

with $\|\cdot\|$ denoting the Euclidean norm, and where $\mathbf{H} \in \mathbb{R}^{n\times N}$ is defined as the output matrix of the hidden layer, whose entries are given by

$$\mathrm{H}_{ij} = \sigma(\boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{x}_i + b_j)\,. \tag{2}$$

The input weights $\{\boldsymbol{a}_j\}_{1\le j\le n}$ and the biases of hidden neurons $\{b_j\}_{1\le j\le n}$ are randomly selected from a continuous distribution, therefore the matrix $\mathbf{H}$ is fixed and the weights $\boldsymbol{w}$ can be computed by means of the Moore–Penrose pseudoinverse:

$$\boldsymbol{w} = \mathbf{H}^{\dagger}\boldsymbol{y}\,,$$

where

$$\begin{aligned} &\mathbf{H}^{\dagger} = \mathbf{H}^{-1} && \text{when } n = N \\ &\mathbf{H}^{\dagger} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}} && \text{when } n > N \ \text{ (over-determined system)} \\ &\mathbf{H}^{\dagger} = \mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{H}^{\mathrm{T}})^{-1} && \text{when } n < N \ \text{ (under-determined system)}\,. \end{aligned} \tag{3}$$

Note that in presence of vector data $\boldsymbol{y}_i \in \mathbb{R}^m$, ELM amounts to compute the solution of the minimization problem

$$\min_{\mathbf{W}\in\mathbb{R}^{m\times N}} \|\mathbf{H}\mathbf{W}^{\mathrm{T}} - \mathbf{Y}^{\mathrm{T}}\|\,,$$

where $\mathbf{H} \in \mathbb{R}^{n\times N}$ is defined as in (2), $\mathbf{Y} = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_n] \in \mathbb{R}^{m\times n}$ is the data matrix, and $\mathbf{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_N]$, with $\boldsymbol{w}_j \in \mathbb{R}^m$, is the output weights matrix. In other words, due to the linearity of the problem, the case of vector data amounts to consider $m$ SLFNs sharing input weights and hidden neurons' biases, but each one with its own output weights. Also notice that the Moore–Penrose pseudoinverse is not the sole choice to obtain the solution to the minimization problem. In fact, being the problem of Least-Square type, one can consider other valid resolution strategies such as the ones related to the resolution of the normal equations. In order to keep a general framework for the incremental procedures presented in Section 4 we adopt the pseudoinverse, interested readers in valid alternatives can refer to [32].

### 3.2. Theoretical results

To explore the approximation capabilities of ELMs, one can adopt the interpolation theory viewpoint, so that the following result can be proven.

**Theorem 1** (*Theorem 2.1 & 2.2 [2]*). *Given any small positive value $\epsilon > 0$ and any activation function $g : \mathbb{R} \to \mathbb{R}$, which is infinitely differentiable in any interval, and $n$ arbitrary distinct samples $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, there exists $N \leq n$ such that, for any $\{(\boldsymbol{a}_j, b_j)\}_{1 \leq j \leq N}$ randomly generated from any interval of $\mathbb{R}^d \times \mathbb{R}$, according to any continuous probability distribution, with probability one, $\|\mathbf{H}\boldsymbol{w} - \boldsymbol{y}\| < \epsilon$. Furthermore, if $N = n$, then with probability one, $\|\mathbf{H}\boldsymbol{w} - \boldsymbol{y}\| = 0$.*

In the following, by "continuous probability distribution" or "continuous sampling distribution" we mean the law of a random variable $X : (\Omega, \mathcal{F}, \mathbb{P}) \to (\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$ whose cumulative distribution function $F_X(\boldsymbol{x}) = \mathbb{P}(X_1 < x_1, \dots, X_n < x_n)$ is continuous.

Theorem 1 provides a finite upper bound to the size of the network for any given training dataset, since one can achieve the highest accuracy by considering $N = n$. Notice that, in practice, one tends to keep $N < n$ so as to limit the ill-conditioning of matrix $\mathbf{H}$. Nonetheless, the previous theorem is a reference result in investigations concerning the decreasing rate of $\|\mathbf{H}\boldsymbol{w} - \boldsymbol{y}\|$ when regarded as a function of $N$.

The next theorem extends the previous results to the case of function approximation:

**Theorem 2** (*Theorem 2.3 [33]*). *Given any nonconstant piecewise continuous function $G : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$, if $\operatorname{span}\{G(\boldsymbol{x}, \boldsymbol{a}, b) : (\boldsymbol{a}, b) \in \mathbb{R}^d \times \mathbb{R}\}$ is dense in $\mathcal{L}^2$, for any continuous target function $f$ and any function sequence $\{\sigma(\boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{x} + b_j) = G(\boldsymbol{x}, \boldsymbol{a}_j, b_j)\}_{1 \leq j \leq N}$ randomly generated according to any continuous sampling distribution, then $\lim_{N \to \infty} \|f - f_N\| = 0$ holds with probability 1, with $f_N = \sum_{j=1}^{N} w_j^* G(\boldsymbol{x}, \boldsymbol{a}_j, b_j)$ and the weights $w_j^*$ determined by minimizing $\left\| \sum_{j=1}^{N} w_j G(\boldsymbol{x}, \boldsymbol{a}_j, b_j) - f(\boldsymbol{x}) \right\|$.*

This time the conclusions are exclusively theoretical, but if we look at a function $f$ as an infinite dataset $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$ we can see some similarities with Theorem 1 when $n \to \infty$.

### 3.3. Some consequences of random initialization

In Theorems 1, 2, one assumes that the input parameters $\{\boldsymbol{a}_j, b_j\}_j$ are random realizations of a given continuous probability distribution that, in general, is chosen to be a uniform distribution. However, this choice can strongly affect the condition number of the overall resolution because the $j$th column of $\mathbf{H}$ is the sampling of the function $\sigma(\boldsymbol{a}_j^{\mathrm{T}}\boldsymbol{x} + b_j)$ on the point $\boldsymbol{x}_i$. We aim to explore such behavior with the final goal of designing a strategy that can mitigate this phenomenon.

Since we are interested in the approximation of functions on a bounded domain, a source of computational instability is the case in which several sigmoids have a flat behavior inside the domain — see also the discussion in [34]. For simplicity, we consider the case in which there is a single node in the input layer, therefore the $j$th sigmoid has an inflection point in $-b_j/a_j$, while far from this value its variations are modest.

Let $p_a$, $p_b$ denote the probability density functions of which $a_j$ and $b_j$, respectively, are realizations. Assume that both $a_j$ and $b_j$ are drawn from a uniform distribution in the interval $[-M, M]$, that is

$$p_a(x) = p_b(x) = \begin{cases} \frac{1}{2M} & \text{if } x \in [-M, M] \\ 0 & \text{otherwise} \end{cases}.$$

Then, one can regard $b_j/a_j$, i.e. the opposite of the inflection point, as a realization of a random variable given by the ratio of two uniform distributions in $[-M, M]$, whose probability density function, based on basic calculus rules, reads

$$p_{b/a}(z) = \begin{cases} \frac{1}{4} & \text{if } z \in [-1, 1] \\ \frac{1}{4z^2} & \text{otherwise} \end{cases}. \tag{4}$$

An alternative approach is to use a uniform distribution to initialize directly the position of the inflection points – originally computed as $c_j = -b_j/a_j$ – and the slopes – i.e. $a_j$ – of the sigmoids. Finally, we can compute the biases as $b_j = -a_j c_j$. In Section 5.1 we see how this choice is much more feasible for our computations.

## 4. Convergence of ELM

The theoretical results reported in the previous section motivate the interest in exploring the convergence of ELM network when the number of neurons $N$ or the number of input data $n$ increases. Here, we consider these two scenarios in a separate and combined way.

More specifically, given a SLFN characterized by $N$ neurons and taking as input $n$ sampling data, we are going to discuss the following cases:

(i) the number of neurons $N$ is increased and the output weights vector is entirely updated;
(ii) the number of neurons $N$ is increased, the weights corresponding to the new neurons are computed and the old weights are not modified;
(iii) the number of training samples $n$ is increased;
(iv) the number of neurons $N$ and the number of training samples $n$ are increased.

We are going to detail the computational cost of each procedure, while a comparison in terms of convergence rate will be set up in Section 5. Notice that increasing the number of neurons $N$, as in cases $(i)$, $(ii)$, will give us the chance to empirically explore the convergence result stated in Theorem 2 and holding for $N \to +\infty$. This is the usual framework of incremental ELMs, as presented in [1,35]. The case where the increase involves the training samples is referred to as online sequential, meaning that the training data are received sequentially [36]. Moreover, generally speaking, cases $(i)$, $(iii)$ involve the inversion either of a block matrix or of a matrix that is adjusted by adding a low-rank update. In this perspective, we will heavily exploit Lemma 1, 2. Finally, in case $(iv)$ we will investigate the implications of Theorem 1.

### 4.1. Case (i): increase the number of neurons and update the output weights

The first case that we consider is the one where, having more samples than neurons, we can increase the number of neurons. This is the case of the least-square approximation, where the system is overdetermined — see (3).

Assume a SLFN has already been trained with $N_k$ hidden neurons on $n$ samples, with $N_k < n$, and the resulting output weights vector is $\boldsymbol{w}_k \in \mathbb{R}^{N_k}$.

We want to re-train the same network with $\delta N_k$ additional hidden neurons on the same data. We introduce

$$N_{k+1} = N_k + \delta N_k$$
$$\mathbf{H}_{k+1} = \begin{bmatrix} \mathbf{H}_k & \boldsymbol{\delta}\mathbf{H}_k \end{bmatrix},$$

with $\boldsymbol{\delta}\mathbf{H}_k$ containing the columns of the new coefficient matrix accounting for the newly introduced neurons.

Note that we want the overall system to remain over-determined, that is $\delta N_k$ is such that $N_{k+1} < n$. According to (3), the new output weights vector $\boldsymbol{w}_{k+1}$ can be computed as follows:

$$\begin{aligned} \boldsymbol{w}_{k+1} &= (\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{H}_{k+1})^{-1}\mathbf{H}_{k+1}^{\mathrm{T}}\boldsymbol{y} \\ &= \begin{bmatrix} \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k & \mathbf{H}_k^{\mathrm{T}}\boldsymbol{\delta}\mathbf{H}_k \\ \boldsymbol{\delta}\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k & \boldsymbol{\delta}\mathbf{H}_k^{\mathrm{T}}\boldsymbol{\delta}\mathbf{H}_k \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{H}_k^{\mathrm{T}} \\ \boldsymbol{\delta}\mathbf{H}_k^{\mathrm{T}} \end{bmatrix} \boldsymbol{y}. \end{aligned}$$

Notice that $\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k$ takes the form of a block matrix; for computing its inverse, we can thus rely on Lemma 1, that can be easily applied by setting

$$\mathbf{A} = \mathbf{A}_k = \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k\,, \quad \mathbf{B} = \mathbf{B}_k = \mathbf{H}_k^{\mathrm{T}}\delta\mathbf{H}_k\,, \quad \mathbf{C} = \mathbf{C}_k = \delta\mathbf{H}_k^{\mathrm{T}}\delta\mathbf{H}_k\,.$$

This procedure can be iterated many times without retraining the network from scratch. It produces the ELM network at each iteration.

Notice that, when $N_k = n$ we recover the same optimal solution obtained by matrix inversion so that in this case the expected training error vanishes, as stated in Theorem 1. This implies that, neglecting numerical perturbations arising in the solution of the linear system, there is no advantage in terms of training error when selecting $N_k > n$.

### 4.2. Case (ii): increase the number of neurons and compute the new output weights

Let us consider a SLFN with $N-1$ hidden neurons and $n$ input samples. Assume that the output weights vector $[w_1,\dots,w_{N-1}]^{\mathrm{T}} \in \mathbb{R}^{N-1}$ is available. In what follows we will address the case in which a new neuron is added to the network, which implies that a novel weight $w_N$ has to be computed, while the remaining weights, up to $w_{N-1}$, are kept unchanged.

The minimization problem of interest now takes the form

$$\min_{w_N} \|\boldsymbol{r}[n,N]\|^2 \tag{5}$$

where $\boldsymbol{r}[n,N] \in \mathbb{R}^n$ is the residual error of a network with $N$ hidden neurons trained on $n$ samples, with entries defined as

$$r_i[n,N] = y_i - \sum_{j=1}^{N} \mathrm{H}_{i,j} w_j\,, \qquad i = 1,\dots,n\,. \tag{6}$$

The explicit form of problem (5) reads

$$\min_{w_N} \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{N} \mathrm{H}_{i,j} w_j \right)^2 .$$

We isolate the term depending on $w_N$

$$\min_{w_N} \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{N-1} \mathrm{H}_{i,j} w_j - \mathrm{H}_{i,N} w_N \right)^2 ,$$

and using (6) we get

$$\min_{w_N} \sum_{1=1}^{n} (r_i[n,N-1] - \mathrm{H}_{i,N} w_N)^2 .$$

By imposing a first order optimality condition for the above minimization problem, we have that $w_N$ has to solve

$$\sum_{i=1}^{n} 2(\mathrm{H}_{i,N}^2 w_N - \mathrm{H}_{i,N} r_i[n,N-1]) = 0\,.$$

Referring to the $N$th column of $\mathbf{H}$ as $\boldsymbol{h}_N$, we get in vector notation:

$$\boldsymbol{h}_N^{\mathrm{T}}\boldsymbol{h}_N w_N - \boldsymbol{h}_N^{\mathrm{T}}\boldsymbol{r}[n,N-1] = 0$$

and finally

$$w_N = \frac{\boldsymbol{h}_N^{\mathrm{T}}\boldsymbol{r}[n,N-1]}{\boldsymbol{h}_N^{\mathrm{T}}\boldsymbol{h}_N}\,.$$

Note the computation of $w_N$ does not involve solving linear systems, therefore there is no need to make a distinction between over-determined and under-determined problems.

### 4.3. Case (iii): increase the number of samples

Assume a SLFN has already been trained with $N$ hidden neurons on $n_k$ samples, with $N < n_k$ and resulting output weights vector $\boldsymbol{w}_k \in \mathbb{R}^N$.

This is the case where new labeled data are available and the dimension of the network cannot change, that in supervised learning is known as online sequential training. Also in this case, we can benefit from the structure of the matrix problem and avoid calculating the new weights from scratch, but we rather update them with lower computational cost. In fact, if we want to retrain the same network with the same hidden neurons on $\Delta n_k$ additional samples, we can introduce

$$n_{k+1} = n_k + \Delta n_k\,, \quad \mathbf{H}_{k+1} = \begin{bmatrix} \mathbf{H}_k \\ \boldsymbol{\Delta}\mathbf{H}_k \end{bmatrix}, \quad \boldsymbol{y}_{k+1} = \begin{bmatrix} \boldsymbol{y}_k \\ \boldsymbol{\Delta}\boldsymbol{y}_k \end{bmatrix}.$$

The new output weights vector $\boldsymbol{w}_{k+1}$ can be computed as follows:

$$\begin{aligned}\boldsymbol{w}_{k+1} &= (\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{H}_{k+1})^{-1}\mathbf{H}_{k+1}^{\mathrm{T}}\boldsymbol{y} \\ &= (\mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k + \boldsymbol{\Delta}\mathbf{H}_k^{\mathrm{T}}\boldsymbol{\Delta}\mathbf{H}_k)^{-1} \begin{bmatrix}\mathbf{H}_k^{\mathrm{T}} & \boldsymbol{\Delta}\mathbf{H}_k^{\mathrm{T}}\end{bmatrix} \boldsymbol{y}\,.\end{aligned}$$

The updated matrix can be inverted as shown Lemma 2, with the following substitutions

$$\mathbf{A} = \mathbf{A}_k = \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k\,, \quad \mathbf{V} = \mathbf{V}_k = \boldsymbol{\Delta}\mathbf{H}_k\,.$$

Hence, the network can be updated when new data is available without re-training, at the cost of storing the inverted matrix of the previous iteration.

We point out that in this case, we are not in the framework of Theorem 1, and do not expect convergence of the training error, while we do expect a better generalization error because new issues are taken into account.

### 4.4. Case (iv): increase the number of samples and the number of neurons

We can also combine the two approaches to efficiently increment both hidden neurons and training samples in one step. We introduce:

$$\begin{aligned} n_{k+1} &= n_k + \Delta n_k \\ N_{k+1} &= N_k + \delta N_k \\ \mathbf{H}_{k+1/2} &= \begin{bmatrix} \mathbf{H}_k \\ \boldsymbol{\Delta}\mathbf{H}_k \end{bmatrix} \\ \mathbf{H}_{k+1} &= \begin{bmatrix} \mathbf{H}_{k+1/2} & \delta\mathbf{H}_{k+1/2} \end{bmatrix} \\ \boldsymbol{y}_{k+1} &= \begin{bmatrix} \boldsymbol{y}_k \\ \boldsymbol{\Delta}\boldsymbol{y}_k \end{bmatrix}. \end{aligned}$$

The fractional index is used to make explicit the fact that we are first adding new samples, so that we apply Lemma 2 by setting

$$\mathbf{A} = \mathbf{A}_k = \mathbf{H}_k^{\mathrm{T}}\mathbf{H}_k\,, \quad \mathbf{V} = \mathbf{V}_k = \boldsymbol{\Delta}\mathbf{H}_k\,.$$

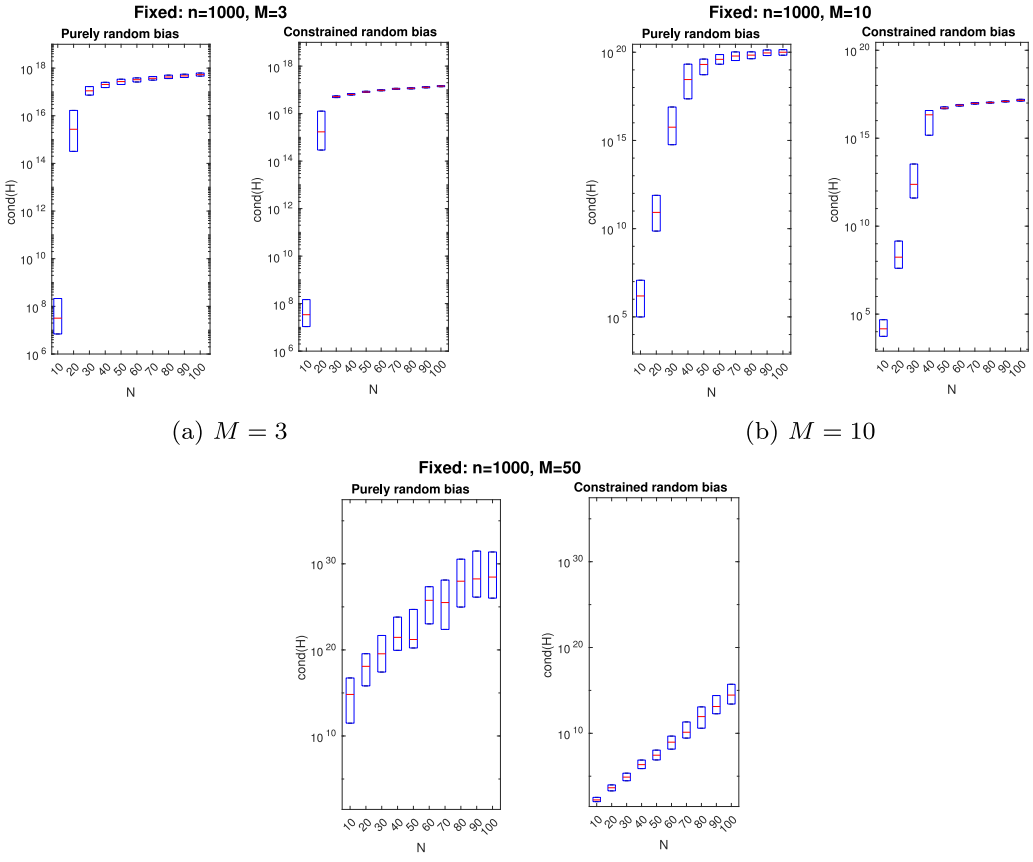
Then, in a second stage, new neurons are added. The output weights vector is computed on the updated sample set relying again on Lemma 1 with

$$\begin{aligned} \mathbf{A} &= \mathbf{A}_{k+1/2} = \mathbf{A}_k + \boldsymbol{\Delta}\mathbf{H}_k^{\mathrm{T}}\boldsymbol{\Delta}\mathbf{H}_k \\ \mathbf{B} &= \mathbf{B}_{k+1/2} = \mathbf{H}_{k+1/2}^{\mathrm{T}}\boldsymbol{\delta}\mathbf{H}_{k+1/2} \\ \mathbf{C} &= \mathbf{C}_{k+1/2} = \boldsymbol{\delta}\mathbf{H}_{k+1/2}^{\mathrm{T}}\boldsymbol{\delta}\mathbf{H}_{k+1/2}\,. \end{aligned}$$

In this case, we benefit from both approaches, so we expect the best results. Moreover, we point out that the underdetermined (over parametrized) problem, solved via pseudo-inversion, has an intrinsic regularization due to the least square interpretation of the solution so that the test (or generalization) error can further decrease in the case where the number of neurons is taken greater than the number of samples.

## 5. Experimental studies

The goal of this numerical section is twofold. First, we want to assess the performance of the initialization strategy, the *constrained biases*, outlined in Section 3.3, see Section 5.1. To this purpose, we monitor the behavior of the condition number of the matrix $\mathbf{H}$ of the ELM training, with and without the proposed strategy, for different number of neurons. Second, our initialization is used to test the four different scenarios discussed in Section 4 on the function approximation problem, see Section 5.2.




(a) $M = 3$ (b) $M = 10$ (c) $M = 50$

**Fig. 1.** Behavior of 2-norm condition number of matrix **H** for different values of M. In all three cases $n = 1000$ and each panel gives a boxplot description of quartiles obtained with 100 initializations of the basis functions.

### 5.1. Testing constrained biases: Condition number

The tails in Eq. (4) make explicit that even with small values of $N$, the classical initialization of input weights and hidden biases could produce a very ill-conditioned linear problem. To measure the condition number in this work, we evaluate numerically the 2-norm condition number of the matrix, that is to say, the ratio of the biggest and smallest singular values. We evaluate the collocation matrix defined via Eq. (2), the basis functions evaluated on random points, and calculate the condition number of this matrix **H**. In Fig. 1 we report results obtained after 100 initializations of the basis functions: in the boxplot we report first and third quartile (blue box) and the median value (red line).

One can notice that when all the inflection points are inside the domain, the condition number of the matrix is smaller by several orders of magnitude, and on top of that increasing $M$ actually helps keep the condition number under control, while increasing $M$ is unfeasible in the classic ELM training. Recall that $a_j$ is taken in the interval $[-M, M]$ so that having more choices gives more various behaviors and that greater $M$ (in norm) gives steeper functions. In the rest of the paper we initialize the biases as specified at the end of Section 3.3, with $M = 10^3$.

### 5.2. Testing approximation accuracy: Mean square error

In the test reported in this section, we address the problem of approximating a function $f : \mathbb{R} \to \mathbb{R}$ starting from a given number of training samples $S_{\text{train}} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ with $y_i = f(x_i)$, $i = 1, \dots, n$. The examples that we start from are the following:

$$f_1(x) = \cos(20\,x)\,, \quad f_2(x) = \frac{1}{1 + 25\,x^2}\,, \quad f_3(x) = step\left(x - \frac{1}{\sqrt{2}}\right),$$

with function $step : \mathbb{R} \to \mathbb{R}$ being defined as follows

$$step(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}.$$

Notice that $f_1$ is a highly-oscillating and smooth function, $f_2$ is the Runge function, while $f_3$ is non-differentiable — see Fig. 2. These are considered the standard target functions. Then, with the aim of considering more general behaviors, we parametrize the vertical scaling $k_1$, the horizontal scaling $k_2$, and the horizontal translation $k_3$, so as to obtain the following families of test functions:

$$\begin{aligned} f_1(x; k_1, k_2, k_3) &= k_1 \cos\big(k_2(x - k_3)\big) \\ f_2(x; k_1, k_2, k_3) &= \frac{k_1}{1 + k_2(x - k_3)^2} \\ f_3(x; k_1, k_3) &= k_1\, step\,(x - k_3) \end{aligned} \tag{7}$$

with $k_1 \in [-3, 3]$, $k_2 \in [0, 30]$, and $k_3 \in [-1, 1]$.

In the tests, the training points $x_1, \dots, x_n$ are randomly selected within the interval $[-1, 1]$. To test the generalization capability of the ELM in the different cases, for each test we are also going to consider a testing set $S_{\text{test}} = \{(\bar{x}_1, \bar{y}_1), \dots, (\bar{x}_{\bar{n}}, \bar{y}_{\bar{n}})\}$, with $\bar{x}_1, \dots, \bar{x}_{\bar{n}}$ being equidistant points in $[-1, 1]$.

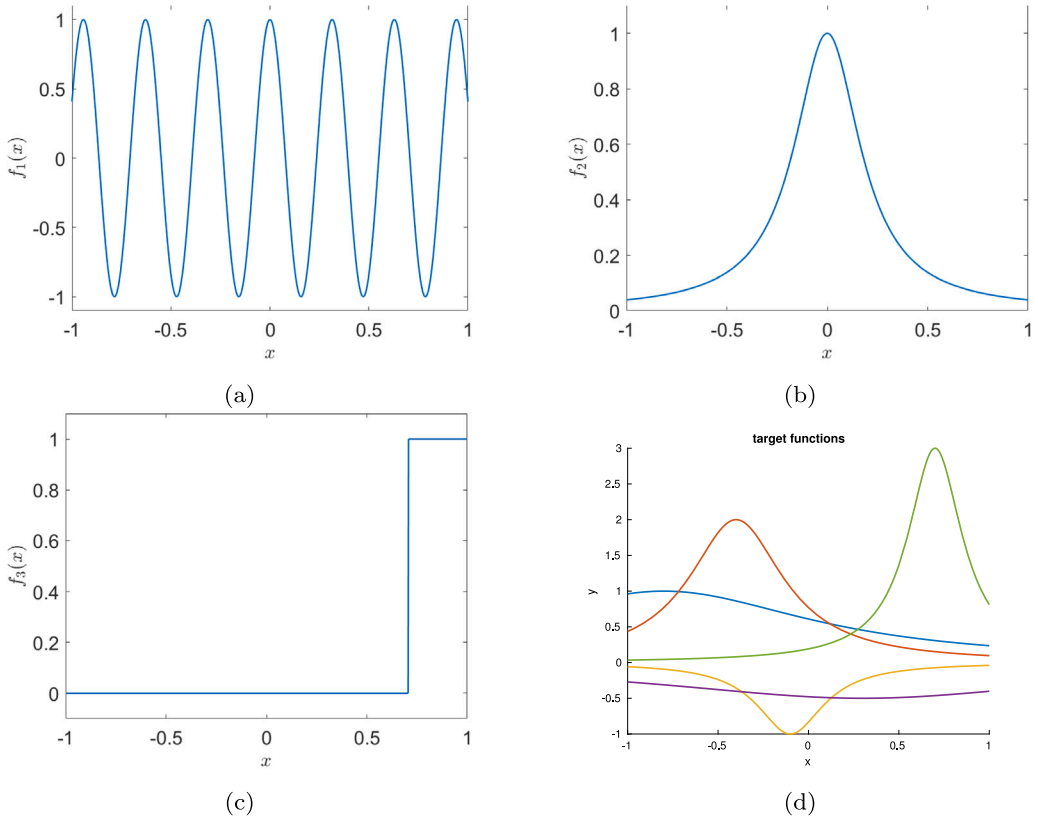



**Fig. 2.** Target functions that are used to populate training and test sets: in (a) function $f_1(x)$; in (b) function $f_2(x)$; in (c) function $f_3(x)$; in (d) some realizations of function $f_2(x; k_1, k_2, k_3)$.

The accuracy of ELM on the training and testing set is measured in terms of the Mean Squared Error (MSE)

$$\text{MSE}_{\text{train}} = \frac{1}{n}\sum_{i=1}^{n}(o_i - f(x_i))^2, \quad \text{MSE}_{\text{test}} = \frac{1}{\bar{n}}\sum_{i=1}^{\bar{n}}(\bar{o}_i - f(\bar{x}_i))^2$$

with $o_i$, $\bar{o}_i$ denoting the value of the approximation provided by ELM and corresponding to $x_i$, $\bar{x}_i$, respectively.

### 5.2.1. Case (i)

We now consider the first scenario discussed in Section 4 and corresponding to the case in which the number of neurons $N$ in the SLFN underlying the ELM increases and the output weights vector is entirely update every time that new neurons are added.

For each function, ELM is run $K$ times with $N_k = r^k \cdot N_0$ (geometrically scaled) neurons, $k = 0, \dots, K-1$, and with $N_0$ denoting the initial number of neurons; we select $N_0 = 30$, $r = 1.2$ and $K = 13$.[1] The number of training samples is $n = 10^3$, while the number of test samples is $\bar{n} = 6 \times 10^3$.

In Fig. 3, we show the behavior of $\text{MSE}_{\text{train}}$, $\text{MSE}_{\text{test}}$ for the three set of functions when $N_k$ increases. In particular, with the continuous line we report the results obtained for the original function, while the boxpolt reports the first and third quartile (blue box) and the median value (red line) obtained with 100 runs of the ELM approximation with different choices of the parameters $k_i$ in the definition of the family of functions, see (7). The behavior of convergence in the regular cases is very similar to what one would expect for general superposed sigmoidal functions, even if these are constructed in an optimal way, with the training also involves the optimization of the internal parameters. Then, we can conclude that ELMs, where internal parameters are not trained and fixed randomly do not suffer of a reduced convergence.

In light of Theorem 1, one would expect $\text{MSE}_{\text{train}}$ to decrease to 0 when $N$ gets larger. However, the recorded metrics achieve a minimum around $N_k = 220$. Then, the MSEs increase, thus suggesting that the dimension of the linear systems to be solved at each instance of ELM, which clearly influences the condition number of the coefficient matrices, is such that round-off errors take over, and prevents from exactly recovering the theoretical result stated in Section 3.2. To make this more explicit, in the same figure we have included the condition number of the matrix.

### 5.2.2. Case (ii)

Here, we consider the scenario of case $(ii)$, where the number of neurons is increased and only the output weights corresponding to the newly introduced neurons are computed, while the others remain unchanged.

Here, we consider $N_k = 2^k \times N_0$, with $k = 0, \dots, 8$ and $N_0 = 50$. We also set $n = 2 \times 10^3$ and $\bar{n} = 10^4$.

The behavior of the MSEs for the three functions is shown in Fig. 4, and also in this case we report results for randomized versions of the original test functions, as discussed in the previous example and in the relative caption. We can note that in these settings, the computation of the new weights does not involve the solution of a possibly ill-conditioned system. Recall that the iterative procedure is very fast,

[1] Such choice, and the following in the next tests, are only dictated by explanation purposes, and in all our testing the overall behavior is similar to the one presented.

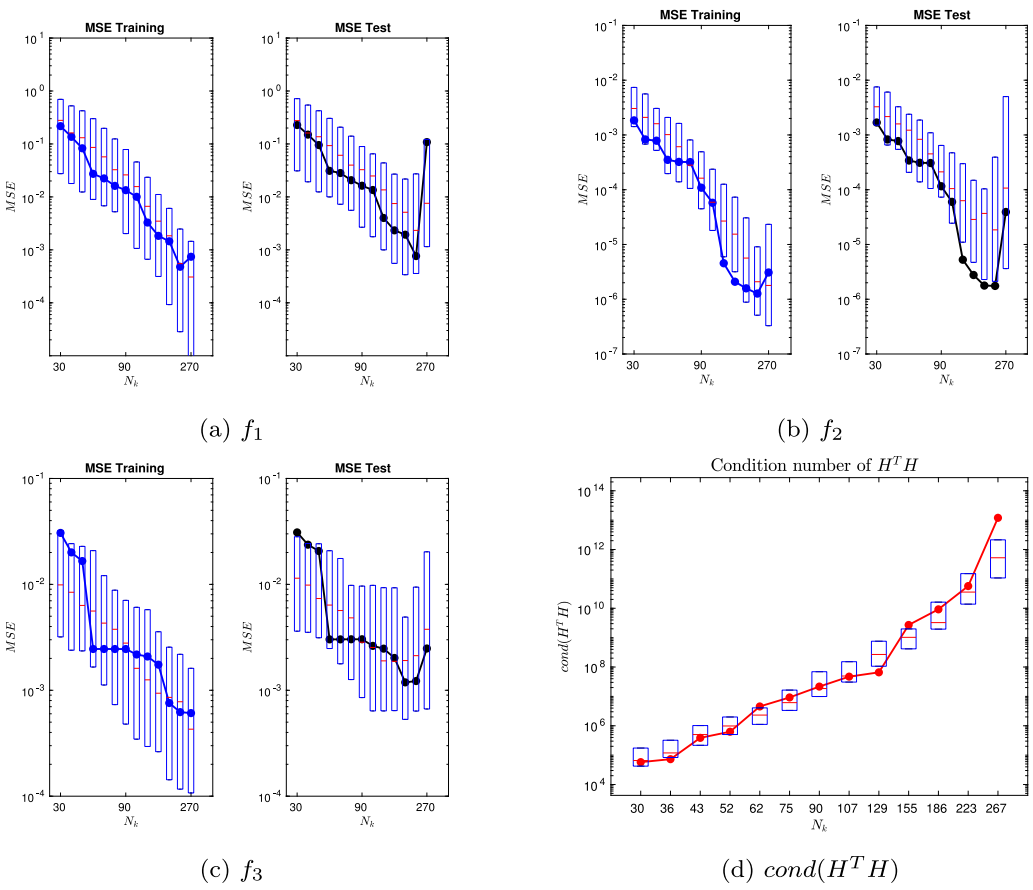



**Fig. 3.** Behavior of $\text{MSE}_{\text{train}}$ (on the left) and $\text{MSE}_{\text{test}}$ (on the right) for the approximation of (a) function $f_1$, (b) function $f_2$, (c) function $f_3$ in case $(i)$, i.e. when the number of neurons increases and the output weights vector is entirely updated. The continuous line we report the results obtained for the original function, in blue the case on training points and in black test points. The boxpolt reports the first and third quartile and the median value obtained with 100 runs of the ELM approximation with different choices of the parameters $k_i$ in the definition of the family of functions, see (7). In panel (d) we report the condition number of the linear system that is solved, in red line the initial case and in the boxplot the quartile values.

due to the structure of the update — see discussion in Section 4.2. As a general result, we notice that the quality metrics decrease without reaching 0, as in the current configuration Theorem 1 does not hold.

#### 5.2.3. Case (iii)

Now, we consider the case in which $N$ is fixed and the number of training samples increases. More specifically, we set $N = 100$, $\bar{n} = 10^4$, and consider $n_k = r^k \times n_0$, with $n_0 = 500$, $r = 1.2$ and $k = 1, \dots, 12$.

From Fig. 5, one can observe that in the three examples, $\text{MSE}_{\text{train}}$ increases while $\text{MSE}_{\text{test}}$ decreases. This is exactly the generalization performance improvement we expected from theory. As already noticed in Section 4.3, in this case we do not expect overall convergence, but the tests confirm that the incremental procedure is stable and able to decrease test error. In this case we do not report results for the randomized functions being the behavior not significant.

#### 5.2.4. Case (iv)

We conclude by addressing the last case, here compared to the other for the first time, where the number of input samples is increased together with the number of neurons.

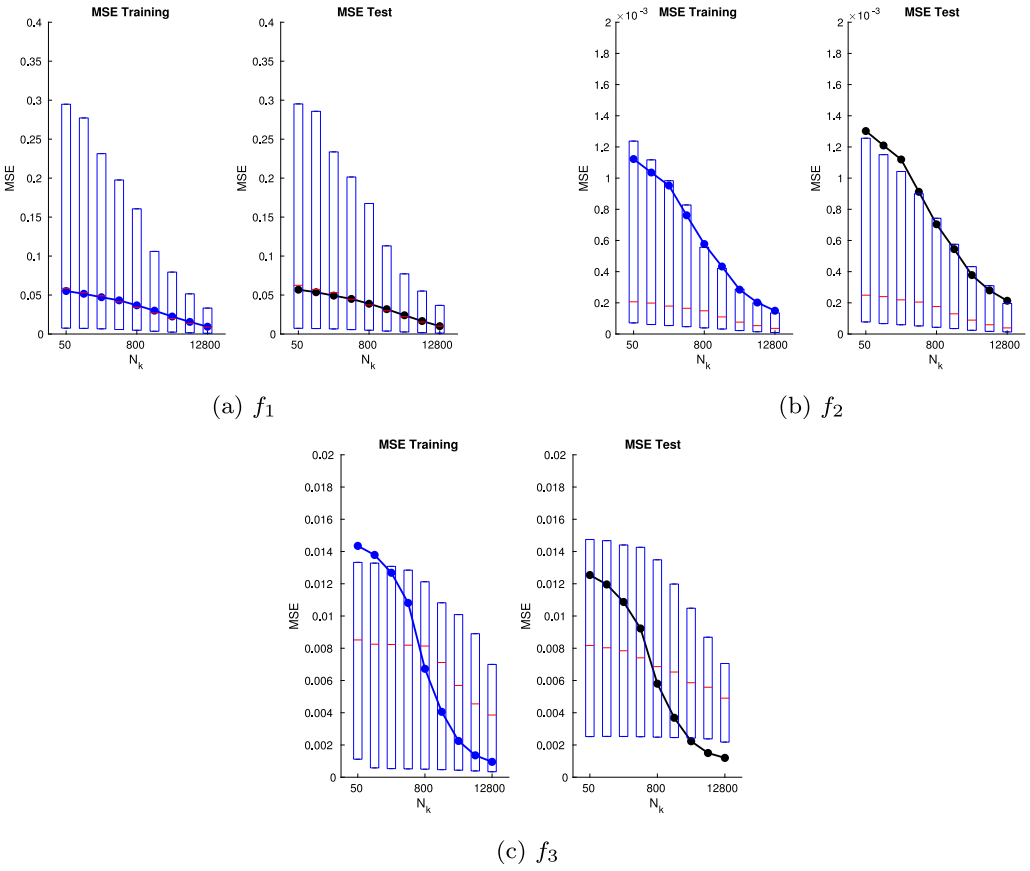
We set $N_k = r^k \times N_0$ and $n_k = r^k \times n_0$, with $N_0 = 30$, $n_0 = 10^3$, $r = 1.2$, $k = 1, \dots, 12$ and $\bar{n} = 10^4$.

The MSE curves shown in Fig. 6 are similar to the ones in Fig. 3, after all, the only difference is that we are also increasing the number of samples, and as a consequence, we can see that the condition number of the matrix is actually lower by roughly an order of magnitude. This suggests that maybe increasing neurons and samples with different values of $r$ could be an alternative way to perform regularization. This gives that the convergence can be compared to the spectral one typical of polynomial interpolation, see [29].

## 6. Conclusions

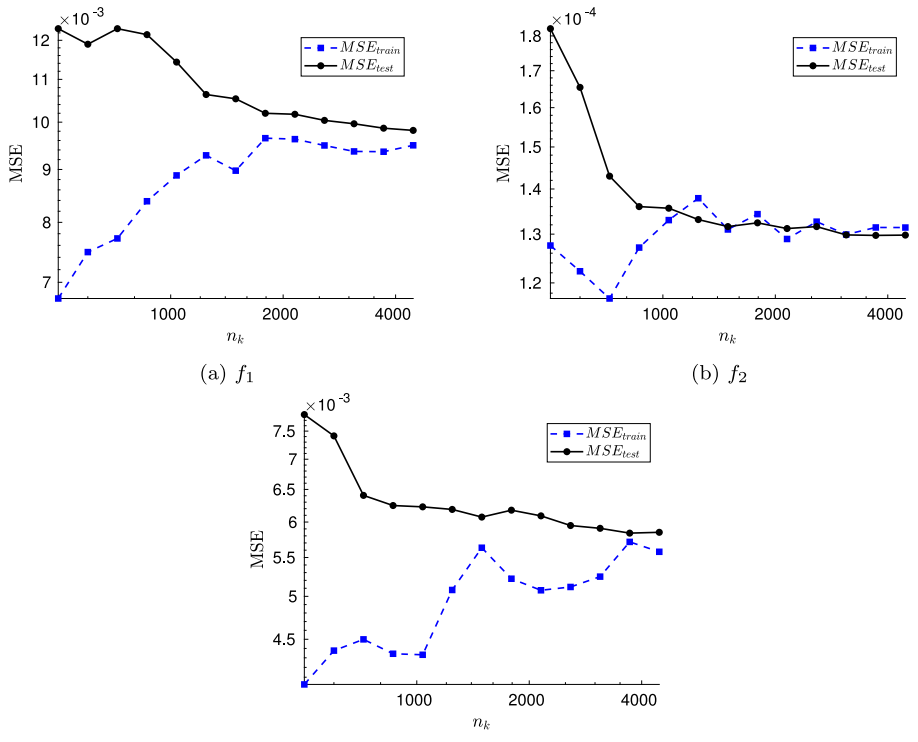
In the present paper, we have studied what happens when one considers constructing a convergent sequence of NNs trained via ELM. Different increasing strategies have been introduced and for each of them, we explore the advantages and disadvantages. A general matrix framework has been introduced both for the evaluation and for the incremental calculations in all cases; computational costs are compared. Being interested in the interpolation and approximation ability of the NN, we have evaluated the mean square error both in the training and in the test set. Our numerical tests confirm that ELM with sigmoidal activation functions is a very powerful machinery that has to be used carefully in order not to be in the presence of ill-conditioned linear problems. Our first result is that with a modification in the first phase of the training, namely the selection of the random parameters in a wise way, a remarkable reduction of the condition number is obtained. Then, we empirically observed that the convergence of incremental ELM with sigmoidal activation functions is not worst than the one known for general superposed sigmoidal functions. Moreover, we obtain spectral-type of convergence when the number of input samples and the number of neurons increase at the same time.

We point out that in the ELM community, other strategies are considered in order to increase the dimension of the network by selecting new neurons iteratively, such as enhanced method [37] or pruning [38,39]. These strategies can help in efficiency but, as seen




(a) $f_1$ (b) $f_2$ (c) $f_3$

**Fig. 4.** Behavior of $MSE_{train}$ (on the left) and $MSE_{test}$ (on the right) for the approximation of (a) function $f_1$, (b) function $f_2$, (c) function $f_3$ in case (*ii*), i.e. when the number of neurons increases and only the weights corresponding to the new neurons are computed. The continuous line we report the results obtained for the original function, in blue the case on training points and in black test points. The boxpolt reports the first and third quartile and the median value obtained with 100 runs of the ELM approximation with different choices of the parameters $k_i$ in the definition of the family of functions, see (7).




(a) $f_1$ (b) $f_2$ (c) $f_3$

**Fig. 5.** Behavior of $MSE_{train}$ and $MSE_{test}$ for the approximation of function $f_1$ (a), $f_2$ (b), $f_3$ (c) in case (*iii*), i.e. when the number of input samples increases.

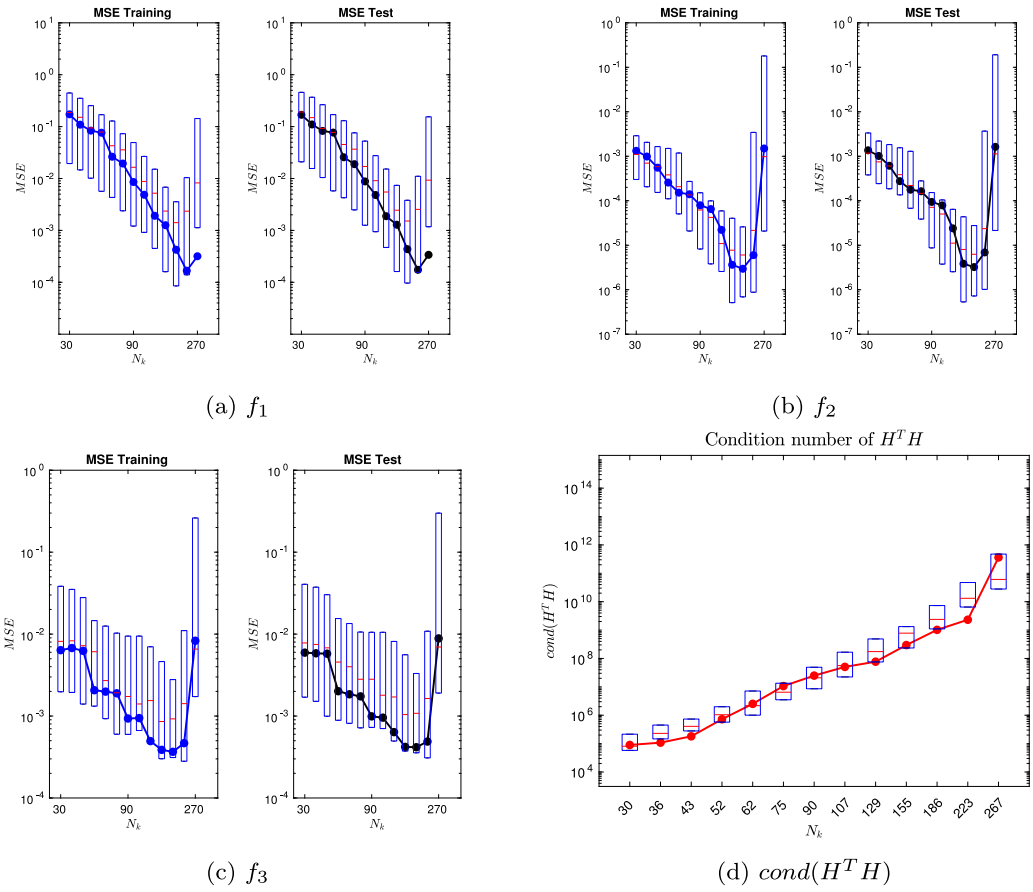



**Fig. 6.** Behavior of $MSE_{train}$ (on the left) and $MSE_{test}$ (on the right) for the approximation of (a) function $f_1$, (b) function $f_2$, (c) function $f_3$ in case $(iv)$, i.e. when the number of input samples and the number of neurons increase. The continuous line we report the results obtained for the original function, in blue the case on training points and in black test points. The boxpolt reports the first and third quartile and the median value obtained with 100 runs of the ELM approximation with different choices of the parameters $k_i$ in the definition of the family of functions, see (7). In panel (d) we report the condition number of the linear system that is solved, in red line the initial case and in the boxplot the quartile values.

in our results, for the considered tests we are able to obtain good results without other modifications. Also in applications where data are affected by errors, these strategies can help but, as discussed in the introduction, special care has to be paid in these cases starting with regularization strategies.

## CRediT authorship contribution statement

**Davide Elia De Falco:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Francesco Calabrò:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Monica Pragliola:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

No data was used for the research described in the article.

## Acknowledgments

All authors are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM); FC and MP were partially supported by projects ‘Progetti di Ricerca GNCS, Italy’.

FC was partially supported by PRIN, Italy 2022 PNRR P2022WC2ZZ ‘A multidisciplinary approach to evaluate ecosystems resilience under climate change’ and by PRIN, Italy 2022 2022N3ZNAX ‘Numerical Optimization with Adaptive Accuracy and Applications to Machine Learning’.

MP acknowledges the 2022 funding research program (FRA) of the University of Naples Federico II, and the European Union-FSE- 394 REACT-EU, PON Research and Innovation 2014–2020 DM1062/2021.

## References

[1] G.-B. Huang, L. Chen, C.K. Siew, et al., Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892.
[2] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (1) (2006) 489–501.
[3] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. B 42 (2) (2011) 513–529.
[4] G. Huang, G.-B. Huang, S. Song, K. You, Trends in extreme learning machines: A review, Neural Netw. 61 (2015) 32–48.
[5] A. Neufeld, P. Schmocker, Universal approximation property of random neural networks, 2023, arXiv preprint arXiv:2312.08410.
[6] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Advances in Neural Information Processing Systems, vol. 20, 2007.

[7] L. Gonon, L. Grigoryeva, J.-P. Ortega, Approximation bounds for random neural networks and reservoir systems, Ann. Appl. Probab. 33 (1) (2023) 28–69.
[8] L. Grigoryeva, J.-P. Ortega, Echo state networks are universal, Neural Netw. 108 (2018) 495–508.
[9] A. Hart, J. Hook, J. Dawes, Embedding and approximation theorems for echo state networks, Neural Netw. 128 (2020) 234–247.
[10] B. Igelnik, Y.-H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Trans. Neural Netw. 6 (6) (1995) 1320–1329.
[11] S. Ding, X. Xu, R. Nie, Extreme learning machine and its applications, Neural Comput. Appl. 25 (2014) 549–556.
[12] J. Wang, S. Lu, S.-H. Wang, Y.-D. Zhang, A review on extreme learning machine, Multimedia Tools Appl. 81 (29) (2022) 41611–41660.
[13] V. Dwivedi, B. Srinivasan, Physics Informed Extreme Learning Machine (PIELM)–a rapid method for the numerical solution of partial differential equations, Neurocomputing 391 (2020) 96–118.
[14] E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, D. Mortari, Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations, Neurocomputing 457 (2021) 334–356.
[15] M. De Florio, E. Schiassi, F. Calabrò, R. Furfaro, Physics-informed neural networks for 2nd order odes with sharp gradients, J. Comput. Appl. Math. 436 (2024) 115396.
[16] F. Calabrò, G. Fabiani, C. Siettos, Extreme learning machine collocation for the numerical solution of elliptic PDEs with sharp gradients, Comput. Methods Appl. Mech. Engrg. 387 (2021) 114188.
[17] F. Calabrò, S. Cuomo, D. di Serafino, G. Izzo, E. Messina, Time discretization in the solution of parabolic PDEs with anns, Appl. Math. Comput. 458 (2023) 128230.
[18] A. Pinkus, Approximation theory of the MLP model in neural networks, Acta Numer. 8 (1999) 143–195.
[19] T. Chen, H. Chen, R.W. Liu, A constructive proof and an extension of Cybenko's approximation theorem, in: Computing Science and Statistics: Statistics of Many Parameters: Curves, Images, Spatial Models, Springer, 1992, pp. 163–168.
[20] H.N. Mhaskar, C.A. Micchelli, Degree of approximation by neural and translation networks with a single hidden layer, Adv. in Appl. Math. 16 (2) (1995) 151–183.
[21] B.I. Hong, N. Hahm, Approximation order to a function in C(R) by superposition of a sigmoidal function, Appl. Math. Lett. 15 (5) (2002) 591–597.
[22] G. Lewicki, G. Marino, Approximation of functions of finite variation by superpositions of a sigmoidal function, Appl. Math. Lett. 17 (10) (2004) 1147–1152.
[23] D. Costarelli, R. Spigler, Constructive approximation by superposition of sigmoidal functions, Anal. Theory Appl. 29 (2) (2013) 169–196.
[24] S. Goebbels, On sharpness of error bounds for univariate approximation by single hidden layer feedforward neural networks, Results Math. 75 (3) (2020) 109.
[25] L. Gonon, Random feature neural networks learn Black-Scholes type PDEs without curse of dimensionality, J. Mach. Learn. Res. 24 (189) (2023) 1–51.
[26] S. Mei, A. Montanari, The generalization error of random features regression: Precise asymptotics and the double descent curve, Comm. Pure Appl. Math. 75 (4) (2022) 667–766.
[27] A. Rahimi, B. Recht, Uniform approximation of functions with random bases, in: 2008 46th Annual Allerton Conference on Communication, Control, and Computing, IEEE, 2008, pp. 555–561.
[28] A. Rahimi, B. Recht, Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning, in: Advances in Neural Information Processing Systems, vol. 21, 2008.
[29] F. Auricchio, M.R. Belardo, F. Calabrò, G. Fabiani, A.F. Pascaner, On the accuracy of interpolation based on single-layer artificial neural networks with a focus on defeating the runge phenomenon, Soft Comput. Accepted Publicat. (2024) 00.
[30] J.M. Martínez-Martínez, P. Escandell-Montero, E. Soria-Olivas, J.D. Martín-Guerrero, R. Magdalena-Benedito, J. Gómez-Sanchis, Regularized extreme learning machine for regression problems, Neurocomputing 74 (17) (2011) 3716–3721.
[31] D.S. Bernstein, Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear Systems Theory, Princeton University Press, 2005.
[32] Å. Björck, Numerical Methods for Least Squares Problems, SIAM, 1996.
[33] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16) (2007) 3056–3062.
[34] P. Horata, S. Chiewchanwattana, K. Sunat, Robust extreme learning machine, Neurocomputing 102 (2013) 31–44.
[35] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Trans. Neural Netw. 20 (8) (2009) 1352–1357.
[36] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Netw. 17 (6) (2006) 1411–1423.
[37] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (16–18) (2008) 3460–3468.
[38] H.-J. Rong, Y.-S. Ong, A.-H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, Neurocomputing 72 (1–3) (2008) 359–366.
[39] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: Optimally pruned extreme learning machine, IEEE Trans. Neural Netw. 21 (1) (2009) 158–162.

**Davide Elia De Falco** is Ph.D. student at "Mathematical And Physical Sciences For Advanced Materials And Technologies" of the "Scuola Superiore Meridionale" (SSM) school of excellence. He is MS in Mathematical Engineering at the Università "Federico II" di Napoli summa cum laude in 2023 & BS in Computer Engineering at the Università di Pisa, scholarship holder of the "Scuola Superiore Sant'Anna" school of excellence.

**Francesco Calabrò** is Associate Professor in Numerical Analysis - University of Naples "Federico II", Italy, since 2020 and Member of the board of the Ph.D. program "Mathematical And Physical Sciences For Advanced Materials And Technologies" of the "Scuola Superiore Meridionale" (SSM). He is BS in Applied Mathematics at the Università "Federico II" di Napoli summa cum laude in 2001 and MS in Applications of Mathematics in Industry and Services at the Università di Milano "Bicocca", Italy. He received his Ph.D. in Computational Science and Informatics at the Università di Napoli "Federico II" in 2004. His research interests include IsoGeometric Analysis and Modeling in Nanochannels and Membranes. Recently he has worked on Scientific Machine Learning, in particular on the use of Extreme Learning Machines for the resolution of differential problems and applications in medical science.

**Monica Pragliola** received her Ph.D. in Pure and Applied Mathematics from the University of Bologna (Italy) in 2020. After a postdoctoral fellowship at the University of Bologna, she is now assistant professor at the Department of Mathematics and Applications of University of Naples Federico II (Italy). Her research interest includes numerical methods for variational approaches in image processing and Bayesian inverse problems.