Deeper Insights into Neural Nets with Random Weights[⋆]

Ming Li$^{1[0000-0002-1218-2804]}$, Giorgio Gnecco$^{2[0000-0002-5427-4328]}$, and Marcello Sanguineti$^{3[0000-0003-0355-8483]}$

[1] Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, Jinhua 321004, China
mingli@zjnu.edu.cn

[2] AXES Research Unit, IMT School for Advanced Studies, Lucca 55100, Italy
giorgio.gnecco@imtlucca.it

[3] Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genoa, Genova 16145, Italy
marcello.sanguineti@unige.it

Abstract. In this work, the "effective dimension" of the output of the hidden layer of a one-hidden-layer neural network with random inner weights of its computational units is investigated. To do this, a polynomial approximation of the sigmoidal activation function of each computational unit is used, whose degree is chosen based both on a desired upper bound on the approximation error and on an estimate of the range of the input to that computational unit. This estimate of the range is parameterized by the number of inputs to the network and by an upper bound both on the size of the random inner weights of the network and on the size of its inputs. The results show that the Root Mean Square Error (RMSE) on the training set is influenced by the effective dimension and by the quality of the features associated with the output of the hidden layer.

Keywords: Neural Networks with Random Weights · Hyperbolic Tangent · Polynomial Approximation · Effective Dimension · Approximate Rank.

1 Introduction

This work analyzes the "effective dimension" of the output of the hidden layer of a random neural network based on a sigmoidal activation function $\sigma : \mathbb{R} \to \mathbb{R}$.

[⋆] G. Gnecco and M. Sanguineti are members of INdAM. G. Gnecco and M. Li acknowledge financial support from the research program ICTP-INdAM Research in Pairs in Mathematics 2020, for the project "On the Expressive Power of Neural Nets with Random Weights". The work of G. Gnecco was supported in part by the Italian Project ARTES 4.0 – Advanced Robotics and enabling digital TEchnology & Systems 4.0, funded by the Italian Ministry of Economic Development (MISE). The work of M. Li was supported in part by the National Natural Science Foundation of China under Grant 62172370.

Such a network computes a function $g : X \subset \mathbb{R}^D \to \mathbb{R}$ of the form

$$g(\mathbf{x}) := \sum_{m=1}^{M} c_m \sigma \left(\mathbf{a}_m \cdot \mathbf{x} - b_m \right) \,, \tag{1}$$

where $\cdot$ denotes the dot product, $\mathbf{x} \in X \subset \mathbb{R}^D$, the weights $c_1, \ldots, c_M$ are optimizable (e.g., using ordinary least squares), whereas the other weight vectors $\mathbf{a}_1, \ldots, \mathbf{a}_M \in \mathbb{R}^D$ and the biases $b_1, \ldots, b_M \in \mathbb{R}$ are randomly extracted before training (i.e., they are nonoptimizable during the training of the network).

The motivation for this study is as follows. Although [6] characterized the approximation power of neural networks with random weights, its main theoretical result (i.e., a sufficient condition for universal approximation[4] by random neural networks in a probabilistic sense) is based on the assumption that there exist some "ideal" support ranges/distributions for randomly assigning the hidden parameters of such networks. Hence, universal approximation is not guaranteed for every random assignment of such parameters, and some key issues are still unsolved [8]: e.g., what are the limitations of neural networks with random weights to approximate certain classes of target functions? For instance, the random parameters of the random neural network are typically extracted from the same interval $[-\lambda, \lambda]$, for some $\lambda > 0$. The choice of λ clearly has an influence on the resulting approximation capability, as partially discussed in [10].

In this paper, we shed some further light on this issue by considering the case of a target function defined on the hypercube $[0,1]^D$ and belonging to the function family $\mathit{\Gamma}_{[0,1]^D,1,C}$ (see [5] for a precise definition of this family), i.e., loosely speaking, a function whose actual dependence is only on one of its variables, and the index of this variable is not known a-priori by the network. Our analysis is based on a polynomial approximation of the sigmoidal activation function (here, the hyperbolic tangent), where the degree of that approximation is chosen implicitly as a function of λ and D, as the "effective domain" of the activation function (the domain on which its input actually ranges) depends on these choices, for each hidden unit. So, by varying λ and D, the "effective dimension" of the output of the hidden layer of the random neural network is expected to change typically (as confirmed by our numerical results). Moreover, we show that by increasing D, an increasingly larger percentage of the hidden features computed by the network is expected to be unrelated to the only variable on which a target function belonging to $\mathit{\Gamma}_{[0,1]^D,1,C}$ depends. This may help to explain, for this choice of the target function, the quite large RMSE error achieved on the training set for large D.

[4] This refers to the approximation of continuous functions on compact sets with arbitrary accuracy, using elements of the specific family of neural networks.

2 Literature review on polynomial approximations of activation functions

The idea of approximating a sigmoidal activation function through a polynomial with a suitable degree is quite natural and has been investigated in previous works (with various research objectives). A comparison of the approximation capability of feedforward neural networks with sigmoidal and polynomial activation functions was made in [2] (see this reference for a precise statements of results). Of the other findings, it was proved that, under mild conditions, the two kinds of networks are equivalent in the sense that, if it is possible to approximate a Lipschitz-continuous function with a suitable Lipschitz constant by one kind of network with an error (in the supremum norm) smaller than a properly defined threshold and a suitable bound on the Lipschitz constant of its neural network approximation, then it is also possible to do the same using the other kind of network, possibly by increasing its size, its number of layers, and the bound on the Lipschitz constant of the corresponding neural network approximation. Feedforward neural networks with polynomial activation functions are also of practical interest because of their suitability to digital implementation, based, e.g., on lookup tables [9]. A drawback, however, is that they do not satisfy the universal approximation property [1]; indeed, according to [7], a one-hidden-layer feedforward neural network can approximate any continuous function defined on a compact set up to any degree of accuracy (with respect to the supremum norm) if and only if its activation functions are not given polynomials. A final remark has to be made on how the coefficients of the polynomial approximation of an activation function can be obtained in practice. While a Taylor approximation appears to be suitable in the case in which only a local approximation is needed (provided the original activation function is locally smooth enough), either a least-squares approximation or a Chebyshev approximation is more suitable to a global approximation (e.g., on a closed and bounded interval).

3 Analysis

In this work, focus is given to the rank (but also to a sort of “approximate rank”, based on the distribution of singular values, as discussed later in this paper) of the hidden output matrix of a one-hidden-layer random neural network, since this can be a measure of the “effective dimension” of the feature space associated with that network.

To undertake the analysis, the following proposition, inspired by [3, Lemma 1], is used. It differs from that result because the proof of the next proposition does not allow one to conclude that the equality holds in the following Equation (3). However, this is enough to complete the successive analysis. In the proposition, $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$ refers to the elements of a training set of size N. We recall that the hidden output matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$ collects in each of its N rows the M outputs of the hidden neurons of the neural network (each row being associated with a particular input vector belonging to the training set).

Proposition 1. *Let the rows of the hidden output matrix* $\mathbf{H} \in \mathbb{R}^{N \times M}$ *have the following form:*

$$\mathbf{r}_n = \left[\sigma_{pol}^{(P)}\left(\mathbf{a}_1 \cdot \mathbf{x}_n - b_1\right), \ldots, \sigma_{pol}^{(P)}\left(\mathbf{a}_M \cdot \mathbf{x}_n - b_M\right)\right], n = 1, \ldots, N, \tag{2}$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_M, \mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$ *and* $\sigma_{pol}^{(P)} : \mathbb{R} \to \mathbb{R}$ *is a given polynomial approximation (with degree* P*) of a sigmoidal function* $\sigma : \mathbb{R} \to \mathbb{R}$*. Then, one has*

$$\operatorname{rank}(\mathbf{H}) \leq \min\left\{M, N, \binom{D+P}{P}\right\}. \tag{3}$$

Proof. By introducing the multi-index $\boldsymbol{\mu} \in \mathbb{N}_0^D$ and the notation $|\boldsymbol{\mu}| := \sum_{d=1}^{D} \mu_i$, then, for $n = 1, \ldots, N$, the column vectors $\mathbf{s}_n = \{x_{n,1}^{\mu_1} \cdots x_{n,D}^{\mu_D}\}_{|\boldsymbol{\mu}| \leq P} \in \mathbb{R}^{\binom{D+P}{P}}$, and, for $m = 1, \cdots, M$, suitable vectors of coefficients $\mathbf{c}_m \in \mathbb{R}^{\binom{D+P}{P}}$ (which depend both on $\mathbf{a}_m$ and on the $P+1$ coefficients of the polynomial $\sigma_{pol}^{(P)}$), and finally, the matrix $\mathbf{C} = \left[\mathbf{c}_1 \middle| \ldots \middle| \mathbf{c}_M\right] \in \mathbb{R}^{\binom{D+P}{P} \times M}$, one gets

$$\mathbf{r}_n = \mathbf{s}_n^\top \mathbf{C}. \tag{4}$$

Hence, by introducing the matrix $\mathbf{S} = \begin{bmatrix} \mathbf{s}_1^\top \\ \hline \cdots \\ \hline \mathbf{s}_N^\top \end{bmatrix} \in \mathbb{R}^{N \times \binom{D+P}{P}}$, one gets

$$\mathbf{H} = \mathbf{S}\mathbf{C}. \tag{5}$$

Since $\operatorname{rank}(\mathbf{C}) \leq \min\{\binom{D+P}{P}, M\}$ and $\operatorname{rank}(\mathbf{S}) \leq \min\{N, \binom{D+P}{P}\}$, one gets Equation (3). □

Based on Proposition 1, we analyze different polynomial approximations of the hyperbolic tangent sigmoidal function $\sigma(z) = \frac{1}{1+\exp(-z)}$, and the effect of training random neural networks based on such approximations. The reason why a polynomial approximation is considered instead of the hyperbolic tangent itself is that the hidden output matrix corresponding to the latter has a full rank under mild conditions [4] (even though some of its positive singular values could be very small). The rank (or the "approximate" rank, as discussed in the next section) of the hidden output matrix corresponding to the use of a good polynomial approximation (i.e., having a reasonably small approximation error over the domain of interest for the analysis) is expected to be a good approximation of the number of largest singular values of the hidden output matrix (with respect to a reasonable threshold) corresponding to the use of the hyperbolic tangent.

Our polynomial approximations are constructed as follows. First, for $L > 0$, we restrict the domain of σ to $[-L, L]$. The specific choice of L is discussed later and depends on the fact that, for the random neural network architecture studied in our analysis, it is possible to find a lower and an upper bound on the inputs to its computational units, depending on the constraints on the weights of the hidden neurons, on the constraints on the components of the feature vector (which are the inputs to such a network), and on the dimension D of that feature vector. Then, for each degree P, we get the best polynomial approximation $\sigma_{pol,[-L,L]}^{(P)}$ of degree P to σ in the weighted $\mathcal{L}_2([-L,L],m_u)$ norm[5] (where m_u is the uniform probability measure on $[-L,L]$). Finally, we fix a desired upper bound $\varepsilon > 0$ on the approximation error in that norm[6], and we choose the smallest degree P° for which $\|\sigma_{pol,[-L,L]}^{(P)} - \sigma\|_{\mathcal{L}_2([-L,L],m_u)} \leq \varepsilon$. To avoid boundary effects (e.g., undesired oscillations near the boundary, see Figure 1), the domain is further restricted – for some choice of $\Delta L \in (0,L)$ – to $[-\bar{L},\bar{L}] := [-L+\Delta L, L-\Delta L]$. As an example, Figure 1(a) shows the graphs of the hyperbolic tangent and of its polynomial approximation found by the procedure above on the domain $[-7,7]$ and with tolerance $\varepsilon = 10^{-4}$. The approximation has degree $P^\circ = 8$. To avoid boundary effects, its domain is further restricted to $[-5,5]$, having chosen $\Delta L = 2$ by trial and error. Similarly, Figure 1(b) refers to the case $\varepsilon = 10^{-5}$, for which the optimal degree of the polynomial approximation is $P^\circ = 12$.

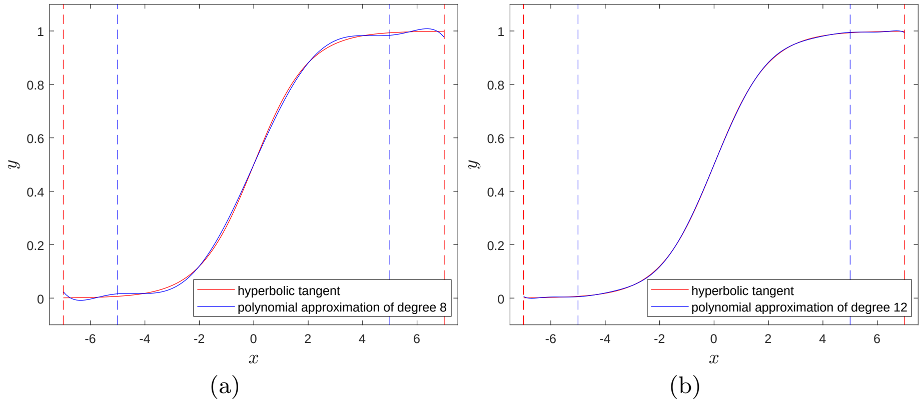


Fig. 1: (a) Polynomial approximation (with degree $P = 8$) of the hyperbolic tangent on $[-7,7]$, and a restriction of its domain to $[-5,5]$. (b) Polynomial approximation (with degree $P = 12$) of the hyperbolic tangent on $[-7,7]$, and a restriction of its domain to $[-5,5]$.

In the following section, two simulation scenarios are considered. For each such scenario and for each choice of the tolerance ε on the approximation error,

[5] In our numerical implementation, the integral in the definition of the weighted $\mathcal{L}_2([-L,L],m_u)$ norm is approximated by a finite summation on a uniform and fine grid.

[6] The error could be also measured according to the supremum norm, and Chebyshev polynomials [12] could be used to achieve this aim. However, the results obtained with the weighted $\mathcal{L}_2([-L,L],m_u)$ are already good enough, as shown in Figure 1.

a suitable value for $\bar{L}$ (and, as a consequence, one also for P°) is obtained as follows. First, an admissible range for the input $\mathbf{a}_m \cdot \mathbf{x}_n - b_m$ of each computational unit is found. Assuming that $x_{n,d} \in [0,1]$ for each component of $\mathbf{x}_n$ and, given $\lambda > 0$, one has $a_{m,d} \in [-\lambda, \lambda]$ for each component of $\mathbf{a}_m$, and $b_m \in [-\lambda, \lambda]$, one also gets $\mathbf{a}_m \cdot \mathbf{x}_n - b_m \in [-(D+1)\lambda, (D+1)\lambda]$, i.e., $\bar{L} := (D+1)\lambda$. Then, by taking, e.g., $\varDelta L = 2$ as before, a possible choice for L is $L = \bar{L} + \varDelta L = (D+1)\lambda + 2$.

In the next section, to further justify the use of a polynomial approximation of the hyperbolic tangent sigmoidal function, the random neural network is trained both by using the activation function, then its polynomial approximation. In both cases, the same training inputs are used. Then, to check the quality of the approximation, the results obtained in the two cases are compared on the training set, and later on the test set. For a fair comparison, the same training set is used in both cases, and later, the same test set is used to compare the generalization capabilities of the two trained models.

4 Simulations

The target function used in Simulation 1 (for which $D = 1$) is defined as

$$f(x;\theta) = 0.6 \exp\left(-\frac{(x-0.2)^2}{\theta^2}\right) + 0.4 \exp\left(-\frac{(x-0.5)^2}{\theta^2}\right) + \exp\left(-\frac{(x-0.8)^2}{\theta^2}\right), \tag{6}$$

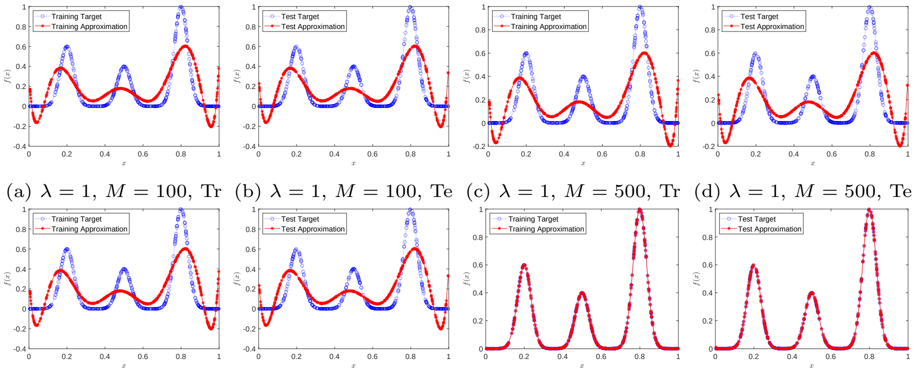
where $x \in [0,1]$, and $\theta > 0$ is a scalar which directly determines the complexity of f. In the case of Simulation 2 (for which $D > 1$), it is defined as

$$f(\mathbf{x};\theta) = 0.6 \exp\left(-\frac{(x_1-0.2)^2}{\theta^2}\right) + 0.4 \exp\left(-\frac{(x_1-0.5)^2}{\theta^2}\right) + \exp\left(-\frac{(x_1-0.8)^2}{\theta^2}\right), \tag{7}$$

where $\mathbf{x} \in [0,1]^D$, i.e., the target function depends only on the first component x_1 of the feature vector $\mathbf{x}$ (but this is not known a-priori by the random neural network). This target function was chosen because it belongs, for $C > 0$ large enough and $D' = 1$, to the family $\varGamma_{[0,1]^D, D', C}$ of D-variable functions on the hypercube $[0,1]^D$ whose actual dependence is only on D' of such variables (whose identity is not known a-priori by the network), and which satisfies a suitable smoothness condition, related to the choice of C (see [5] for the precise definition). In the previous work [5], it was proved that (one-hidden-layer) non-random neural networks (i.e., having optimizable inner parameters of their computational units) have a better approximation capability of functions belonging to $\varGamma_{[0,1]^D, D', C}$ than linear combinations of fixed basis functions. This holds, loosely speaking, because the computational units of such neural networks are flexible enough to be matched each time to the specific subset of D' variables associated with every element of $\varGamma_{[0,1]^D, D', C}$, via suitable choices of their inner parameters. Nevertheless, this flexibility disappears when a random neural network is considered, since the inner parameters of its computational units are fixed once they have been randomly extracted, hence they are not optimizable.

Simulation 1: One-dimensional function approximation. We set $\theta = 0.05$ and sample 1000 instances $\{x_i, f(x_i)\}_{i=1}^{1000}$ based on a regularly spaced grid on [0,1], then randomly (and uniformly) select $N = 500$ samples as the training set, whereas the remaining samples are used for test.

Case 1: Using the hyperbolic tangent sigmoidal function. We test the performance of two random models with $\lambda = 1$ and $\lambda = 10$, respectively, with different numbers of M of hidden neurons. For all the simulations, the sigmoidal activation function $\sigma(z) = \frac{1}{1+\exp(-z)}$ is used. Figure 2 shows the training and test approximation results for four different learner models, as reported respectively in (a) and (b) for the model built with $\lambda = 1, M = 100$, in (c) and (d) for the model built with $\lambda = 1, M = 500$, in (e) and (f) for the model built with $\lambda = 1, M = 10000$, and in (g) and (h) for the model built with $\lambda = 10, M = 200$. It is clear that $\lambda = 1$ does *not* work at all for this simple function approximation problem, even when the number of hidden nodes is sufficiently large. In contrast, the model built with $\lambda = 100$ and trained with $M = 200$ shows a very good learning and generalization capability. On the other hand, based on our empirical experience, some values larger than 1, such as $\lambda = 50, 100, 150, 200$, also work favorably on this regression task, implying that the setting of λ has a strong impact on the expressive power of the resulting randomized learner model.



(a) $\lambda = 1,\ M = 100$, Tr (b) $\lambda = 1,\ M = 100$, Te (c) $\lambda = 1,\ M = 500$, Tr (d) $\lambda = 1,\ M = 500$, Te

(e) $\lambda = 1,\ M = 10000$, Tr (f) $\lambda = 1,\ M = 10000$, Te (g) $\lambda = 10,\ M = 200$, Tr (h) $\lambda = 10,\ M = 200$, Te

Fig. 2: Performance visualization for training (Tr) and testing (Te) results for random neural network models for various combinations of λ and M: (a-b) $\lambda = 1, M = 100$; (c-d) $\lambda = 1, M = 500$; (e-f) $\lambda = 1, M = 10000$; (g-h) $\lambda = 10, M = 200$.

Case 2: Using the polynomial approximation. We investigate the performance of two random neural network models with $\lambda = 1$ and $\lambda = 10$, respectively, where the polynomial approximation of the activation function is used for all the simulations. For simplicity, in this case, we only consider $M = 200$. As shown in Figure 3, the training and testing performances of the random model based on the polynomial approximation of the activation function are quite similar to those of the model based on the sigmoidal activation function, confirming the good quality of the approximation constructed in line with the procedure reported in Section 3. Since the computation of the rank of a matrix may suffer from numerical issues, to verify Proposition 1, we calculate an "approximate" rank of $\mathbf{H}$ in MATLAB in the following way. First, we run the command $svd(\mathbf{H})$, which returns the singular values $\{v_1, v_2, \ldots, v_R\}$ of $\mathbf{H} \in \mathbb{R}^{N\times M}$, where $R \leq \min\{N, M\}$. Then, the number of "largest" singular values (hence, the approximate rank) is selected in the following way:

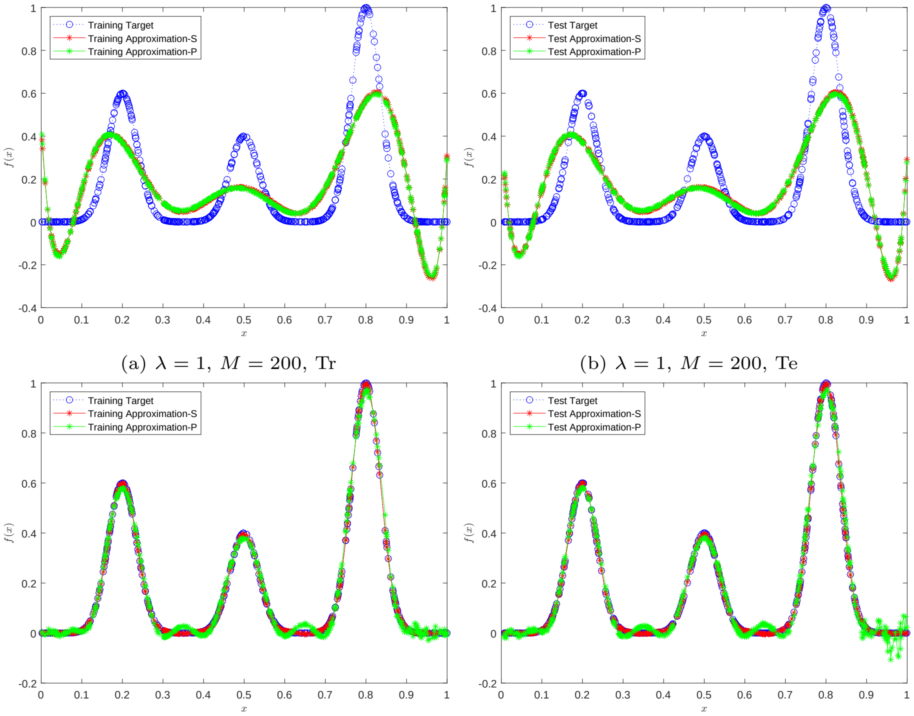
$$\widehat{\text{rank}}(\mathbf{H}) := \text{Cardinality}\{r | v_r > \max\{v_1, v_2, \ldots, v_R\}/S, r = 1, 2, \ldots, R\}, \quad (8)$$

where we set $S = 100$ in the simulations. Figure 4 shows an example of a singular value distribution for two random realizations of matrix $\mathbf{H}$, which clarifies the idea behind the approximation above. Table 1 shows the records of approximate rank values for the hidden output matrix $\mathbf{H}$ (for the case in which the polynomial approximation of the activation function is used), in which we can find that the simulation results are consistent with Proposition 1. In this case, it can be seen from the table that, by increasing λ, also the "effective dimension" of the feature space associated with the hidden output layer tends to increase, providing a better approximation capability of the network with respect to the target function. This may explain the smaller RMSE achieved on the training set for larger λ.

Table 1: Summary of the approximate rank values of the hidden output matrix $\mathbf{H} \in \mathbb{R}^{N\times M}$, of P°, of the upper bound $\min\left\{M, N, \binom{D+P^\circ}{P^\circ}\right\}$ on the rank of $\mathbf{H}$, and of the training RMSE: $N = 500$, $M = 200$, $L = 2\lambda + 2$, $\varepsilon = 10^{-4}$.

λ	$\widehat{\text{rank}}(\mathbf{H})$	P°	$\min\left\{M, N, \binom{D+P^\circ}{P^\circ}\right\}$	Training RMSE
$\lambda = 1$	2	6	7	0.2103
$\lambda = 3$	3	10	11	0.1609
$\lambda = 5$	3	14	15	0.0908
$\lambda = 10$	4	22	23	0.0641
$\lambda = 15$	5	30	31	0.0233
$\lambda = 20$	6	38	39	0.0170

Simulation 2: Multi-dimensional function approximation. To further verify our theoretical result reported in Proposition 1, we extend the 1-D regression task studied in Simulation 1 to a multi-dimensional function approximation



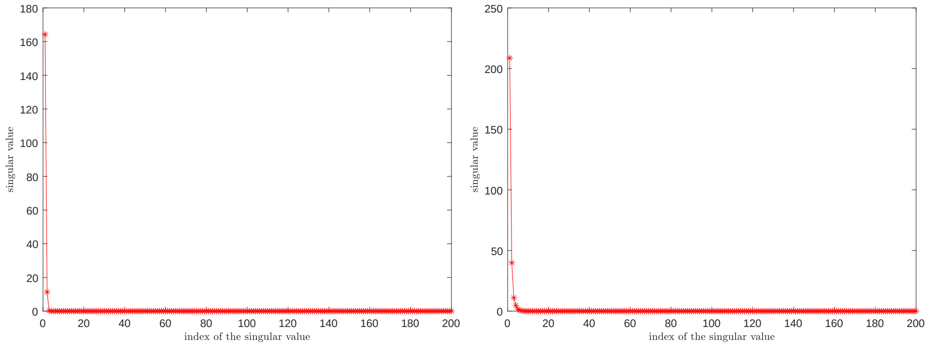
(a) $\lambda = 1,\ M = 200,$ Tr

(b) $\lambda = 1,\ M = 200,$ Te

(c) $\lambda = 10,\ M = 200,$ Tr

(d) $\lambda = 10,\ M = 200,$ Te

Fig. 3: Performance visualization for training (Tr) and testing (Te) results for random neural network models with $\lambda = 1$ and $\lambda = 10$, respectively, for $D = 1$ (i.e., for Simulation 1, Case 2). In the figure, “S” and “P” represent, respectively, the cases in which the hyperbolic tangent sigmoidal activation function and its polynomial approximation are used: (a-b) $\lambda = 1, M = 200,\ L = 4,\ \Delta L = 2,\ \varepsilon = 10^{-7},\ P = 10$; (c-d) $\lambda = 10, M = 200,\ L = 22,\ \Delta L = 2,\ \varepsilon = 10^{-7},\ P = 54$.



(a) $\lambda = 1$

(b) $\lambda = 10$

Fig. 4: Distribution of the singular values for two random realizations of $\mathbf{H}$: (a) for $\lambda = 1$; (b) for $\lambda = 10$, respectively.

problem (with the target function reported in Equation (7)), via artificially including additional input features. Concretely and without loss of generality, we investigate the cases in which $D = 2, 3, 4, 5$, that is, additional features are included as inputs together with the original feature x_1, while still keeping the output $f(\mathbf{x}, \theta)$ as a scalar. In other words, the additional features are only "noisy features", completely uncorrelated with the output. For simplicity, we randomly generate the additional features from the uniform distribution on $[0, 1]^{D-1}$, where $D - 1 = 1, 2, 3, 4$, corresponding to the cases $D = 2, 3, 4, 5$, respectively.

Table 2 summarizes the results obtained. In this case, the increase in the "effective dimension" of the feature spaces does not usually help in reducing the training RMSE. In order to explain the difference with respect to Simulation 1 (see the next paragraph), the following definitions are introduced. One can call "good features" those that are associated with monomials only in x_1 in the polynomial approximation (in total, there are at most $P+1$ such features, where the number $P + 1$ is obtained when all the coefficients of these monomials are nonzero), whereas all the other features can be called "bad features", since they refer to monomials containing at least one of the noisy inputs (in total, there are at most $\binom{D+P}{P} - (P + 1)$ such features, where the number $\binom{D+P}{P} - (P + 1)$ is obtained when all the coefficients of these monomials are nonzero). For simplicity, we use the abbreviations "GF" and "BF" to represent the upper bound on the number of good features ($:= P + 1$) and the upper bound on the number of bad features ($:= \binom{D+P}{P} - (P + 1)$), respectively. It is worth mentioning that this distinction between good features and bad features is highly dependent on the specific simulation scenario, in which the target function depends on one variable but not on the other variables. However, this analysis can be extended to other target functions belonging to the same class $\mathit{\Gamma}_{[0,1]^D,1,C}$. Finally, in the setting considered in Case 2 of Simulation 1, one has $GF = P + 1$ and $BF = 0$, i.e., there are only good features (see Table 1 for the values assumed by the optimal P in the various cases).

Table 2 shows that, by increasing D, the ratio $\frac{GF}{GF+BF}$ is decreasing to 0, i.e., the (upper bound on the) number of bad features tends to dominate the (upper bound on the) number of good features. Moreover, in spite of the quite large values of GF and BF reported in the table, the approximate rank obtained ($\widehat{\mathrm{rank}}(\mathbf{H})$) is typically quite small. Since one can extract from the output of the hidden layer of the network a maximal set of about $\widehat{\mathrm{rank}}(\mathbf{H})$ linearly independent features, and these are linear combinations of both good and bad features (see Equations (4) and (5) in the proof of Proposition 1), one expects that by increasing D, of the coefficients defining the features extracted, the ones associated with the set of bad features dominate the other ones associated with the set of good features.

Table 2: Summary of the approximate rank values of the hidden output matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$, of P°, of GF, of BF, and of the upper bound $\min\left\{M, N, \binom{D+P^{\circ}}{P^{\circ}}\right\}$ on the rank of $\mathbf{H}$. $N = 500$, $M = 200$, $\lambda = 1$, $L = (D+1)\lambda + 2$, $\Delta L = 2$, $\varepsilon = 10^{-4}$. e_1 stands for the training RMSE of the random neural network model based on the polynomial activation function, while e_2 stands for the training RMSE of the random neural network model based on the sigmoidal activation function.

D	$\widehat{\mathrm{rank}}(\mathbf{H})$	P°	$\min\left\{M, N, \binom{D+P^{\circ}}{P^{\circ}}\right\}$	GF	BF	$\frac{GF}{GF+BF}$	e_1	e_2
$D=2$	3	6	28	7	21	0.3333	0.1724	0.1430
$D=3$	4	8	165	9	156	0.0577	0.1838	0.1507
$D=4$	5	8	200	9	486	0.0185	0.1620	0.1589
$D=5$	6	10	200	11	2992	0.0048	0.1637	0.1617

5 Conclusions

In this work, we analyzed the "effective dimension" of the output of the hidden layer of a one-hidden-layer neural network with random inner weights. Our analysis has been based on a polynomial approximation of the sigmoidal activation function of the network, whose degree has been chosen by taking into account both a desired upper bound on the approximation error, an upper bound λ both on the size of the inner weights of the network and on the size of its inputs, and the input dimension D. Our analysis is limited to explaining the behavior of the trained network on the training set, when the D-variable target function depends only on one of its inputs. As a future investigation, we plan to also explain the generalization capability of the trained neural network as a function of D and λ, and to investigate the case in which the target function depends on two or more among its inputs (whose identities are not known in advance), and the case in which its approximation is achieved by using a neural network with random inner weights having two or more hidden layers. Finally, although only the hyperbolic tangent has been used as a case study, our approach based on an optimized polynomial approximation could be applied also to other activation functions which are commonly used in the context of feedforward neural networks [11].

References

1. Cybenko, G.: Approximation by Superposition of a Sigmoidal Function, Mathematics of Control, Signals, and Systems, vol. 2, pp. 303–314, 1989.
2. DasGupta, B., and Schnitger, G.: The Power of Approximation: A Comparison of Activation Functions, Advances in Neural Information Processing Systems (NIPS), pp. 615–622, 1992.
3. Fan, J., and Udell, M.: Online High Rank Matrix Completion, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8690–8698, 2019.
4. Fu, A. M, Wang, X. Z., He, Y. L., and Wang, L. S.: A Study on Residence Error of Training an Extreme Learning Machine and its Application to Evolutionary Algorithms, Neurocomputing, vol. 146, pp. 75–82, 2014.
5. Gnecco, G.: A Comparison between Fixed-Basis and Variable-Basis Schemes for Function Approximation and Functional Optimization, Journal of Applied Mathematics, vol. 2012, article ID 806945, 17 pages, 2012.
6. Igelnik, B., and Pao, Y.-H.: Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-link Net, IEEE Transactions on Neural Networks, vol. 6, pp. 1320–1329, 1995.
7. Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S.: Multilayer Feedforward Networks with a Nonpolynomial Activation Function can Approximate any Function, Neural Networks, vol. 6, pp. 861–867, 1993.
8. Li, M., and Wang, D.: Insights into Randomized Algorithms for Neural Networks: Practical Issues and Common Pitfalls, Information Sciences, vol. 382, pp. 170–178, 2017.
9. Piazza, F., Uncini, A., and Zenobi, M.: Artificial Neural Networks with Adaptive Polynomial Activation Function, Proceedings of the International Joint Conference on Neural Networks (IJCNN), pp. 343–348, 1992.
10. Sonoda, S., Li, M., Cao, F., Huang, C., and Wang, Y. G.: On the Approximation Lower Bound for Neural Nets with Random Weights, arXiv preprint arXiv:2008.08427, https://arxiv.org/abs/2008.08427, 2020.
11. Szandala, T.: Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks, in: Bhoi, A., Mallick, P., Liu, C. M., Balas, V. (Eds.), Bio-inspired Neurocomputing. Studies in Computational Intelligence, Springer, vol. 903, pp. 203–224, 2021.
12. Vlcěk, M.: Chebyshev Polynomial Approximation for Activation Sigmoid Function, Neural Network World, vol. 4, pp. 287–393, 2012.