New Methods of Generating Random Parameters in Feedforward Neural Networks with Random Hidden Nodes*

Grzegorz Dudek[1] [0000−0002−2285−0327]

Electrical Engineering Department, Czestochowa University of Technology,
Czestochowa, Poland
dudek@el.pcz.czest.pl

Abstract. The standard method of generating random parameters in randomized neural network learning selects randomly hidden node weights and biases both from the same fixed interval regardless of the data scope and activation function type. This leads to the poor approximation property of the network. Recently more sophisticated methods have been developed which treat weights and biases separately. They distribute the sigmoids inside the input hypercube and adjust their slopes to the target function complexity using different approaches. These lead to improvement in approximation performance of the network and allow us to control the generalization degree of the model.

Keywords: Feedforward neural networks · Neural networks with random hidden nodes · Randomized learning algorithms.

1 Introduction

In feedforward neural networks (FNNs) the weights are learned iteratively from data using a gradient descent method. Due to a layered structure of the network and nonconvex character of the optimization problem, the gradient-based learning is time consuming, sensitive to the initial settings and leads to the local optima of the error function. In randomized learning the parameters of the hidden neurons are selected by random and stay fixed. The only parameters which are learned are the weights of output neurons. The optimization problem becomes convex, and to solve it the standard linear least-squares can be used [1]. This significantly speeds up the learning process and simplifies its implementation.

The single-hidden-layer FNN with random hidden nodes has universal approximation capability when the random parameters are selected from a symmetric interval according to any continuous sampling distribution [2]. The open questions are: what should be the interval bounds and the parameter distribution? These are the most important research gaps in FNN randomized learning [3]. Widely-used interval in practice, $[-1, 1]$, is misleading, and many authors

* Supported by Grant 2017/27/B/ST6/01804 from the National Science Centre, Poland.

criticize it as devoid of scientific justification, independent on data and activation function (AF) type. It is common practice recently to optimize this interval by looking for its bounds [4].

In this work we present three new approaches for generating FNN random parameters which have been developed recently. In these approaches the weights and biases of the hidden nodes are treated separately, due to their different functions. The biases, which are related with the AF location in space, are determined in such a way that the steepest AF fragments, which are the most useful for modeling the target function fluctuations, are introduced into the input hypercube (IH). The weights, related to the AF slopes, are selected randomly using different approaches, taking into account the TF complexity.

2 Improved Methods of Generating Random Parameters

In this study we consider sigmoids as the AFs of hidden nodes:

$$h(\mathbf{x}) = 1 \Big/ \left(1 + \exp\left(-\left(\mathbf{a}^T\mathbf{x} + b\right)\right)\right) \tag{1}$$

where weights $\mathbf{a} = [a_1, \dots, a_n]^T$ decide about sigmoid slopes in different directions and bias b decides about a shift of the sigmoid.

A linear combination of the sigmoids with coefficients β (output weights selected using a linear least-squares method) gives a fitted function which approximates TF. According to the standard approach, the weights and biases are selected both from the uniform distribution over the same fixed symmetrical interval, $[-u, u]$. As shown in [5], a drawback of this method is that many of the sigmoids have their steepest fragments, which are around of the inflection points, outside of the IH. So, they cannot be used for modeling the TF nonlinearities. Because the interval is common for biases and weights, it is difficult to select it optimally for both parameters in the same time.

Fig. 1 shows an example of single-variable function approximation when using the standard approach with the fixed intervals: $[-1, 1]$ (a) and $[-100, 100]$ (b). The bottom charts show the sigmoids whose linear combination forms the function fitting data (shown as a solid line in the upper charts). For $[-1, 1]$ case, the sigmoids are flat and their distribution in the input interval $[0, 1]$ (shown as a grey field) does not correspond to the TF fluctuations. This results in a very weak fit. For $[-100, 100]$ case, the sigmoids are steeper but many of them have their steepest fragments outside of the input interval. So, many of them are wasted. Moreover, the insufficient number of steep sigmoids in the input interval causes fluctuations of the fitted curve on the flat parts of the TF.

To improve the performance of the model in [5] the rsM method have been proposed which distributes the sigmoids across the IH and adjusts their slopes to the TF fluctuations. It takes into account the IH location and activation function type. According to this method the weights of the i-th hidden node are calculated as follows:

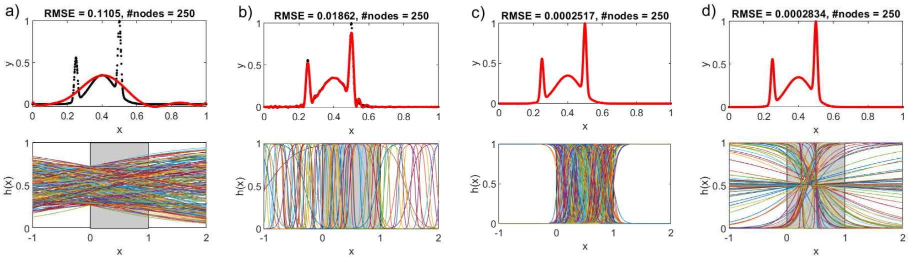

Fig. 1. Fitted curves and distribution of the hidden node sigmoids for: the standard method with interval $[-1, 1]$ (a) and $[-100, 100]$ (b), RARSM (c) and D-DM (d).

$$a_{i,j} = \Sigma_i \zeta_j \Big/ \sum_{l=1}^{n} \zeta_l, \quad j = 1, 2, ..., n \tag{2}$$

where $\zeta_1, \zeta_2, \ldots, \zeta_n \sim U(-1, 1)$ are i.i.d. numbers and Σ_i is the sum of weights of the i-th node, which is randomly chosen from the interval:

$$|\Sigma_i| \in [\ln((1-r)/r), s \cdot \ln((1-r)/r)] \tag{3}$$

where $r \in (0, 0.5)$ and $s > 1$ are the parameters controlling the sigmoid slopes.

The biases of the hidden nodes are determined setting the inflection points of the sigmoids (around these points the sigmoids are steepest) at some points $\mathbf{x}^*$ randomly selected from the IH or randomly chosen from the training set:

$$b_i = -\mathbf{a}_i^T \mathbf{x}^* \tag{4}$$

Another method of improving the randomized learning performance was proposed recently in [6] – random angle, rotation and shift method, RARSM. Firstly, it randomly chooses the slope angles of the sigmoids from an interval adjusted to the TF complexity. Then, the activation functions are randomly rotated around the y-axis and finally, they are distributed across the IH according to data distribution. Their biases are calculated from (4) and weights from:

$$a_{i,j} = -4a'_{i,j}/a'_{i,0}, \quad j = 1, 2, ..., n \tag{5}$$

where $a'_{i,j}$ are components of the normal vector $\mathbf{n}$ to the hyperplane, which is tangent to the sigmoid at their inflection points.

To determine the normal vector $\mathbf{n}$, we randomly select its slope angle $\alpha \in (\alpha_{min}, \alpha_{max})$, where α_{min} and α_{max} are adjusted to the TF. The sigmoid rotation is determined by selecting randomly normal vector components $a'_1, \ldots, a'_n \sim U(-1, 1)$ and calculating the component a'_0 from:

$$a'_0 = (-1)^c \sqrt{(a'_1)^2 + ... + (a'_n)^2} \Big/ \tan\alpha \tag{6}$$

where $c \sim U\{0, 1\}$.

To fit the sigmoids to the data more closely, in [7] a data-driven randomized learning have been proposed (D-DM). According to it, for each node an input space region is selected by choosing randomly one of the training points $\mathbf{x}^*$. Then the hyperplane T is fitted to $\mathbf{x}^*$ and its k nearest neighbors. Assuming that T is tangent to the sigmoid at its inflection points, we determine the weights as:

$$a_{i,j} = 4a''_{i,j}, \quad j = 1, 2, ..., n \tag{7}$$

where $a''_{i,j}$ are the hyperplane coefficients. The bias of the sigmoid is determined from (4), so that its inflection point is in $\mathbf{x}^*$.

Figs. 1 (c) shows the sigmoid distribution in the input interval for RARSM (similar distribution we obtain for rsM). Note that all sigmoids have their inflection points in the input interval and their slopes are adjusted to the TF. This results in decreasing of the fitting error to 0.00025. Similar error level we get when using D-DM. In this case the sigmoid slopes are adjusted to the TF fragments locally, so their slopes change across the input interval.

3 Conclusions

In this work we present three new approaches for generating FNN random parameters, which have been developed recently. According to them, the weights and biases of the hidden nodes are treated separately, due to their different functions. The biases, which are related with the activation function location, are determined in such a way that the steepest activation function fragments, which are the most useful for modeling the TF fluctuations, are introduced into the IH. The weights, related to the activation function slopes, are selected randomly taking into account the TF fluctuations. These new approaches improve the performance of randomized learning and lead to the more compact FNN architecture.

References

1. Principe, J., Chen, B.: Universal approximation with convex optimization: Gimmick or reality? IEEE Comput Intell Mag **10**, 68–77 (2015)
2. Husmeier, D.: Random vector functional link (RVFL) networks. In: Neural Networks for Conditional Probability Estimation: Forecasting Beyond Point Predictions, chapter 6, 87–97, Springer-Verlag London (1999)
3. Weipeng, C., Wang, X., Ming, Z., Gao, J.: A review on neural networks with random weights. Neurocomputing **275**, 278–287 (2018)
4. Li, M., Wang, D.: Insights into randomized algorithms for neural networks: Practical issues and common pitfalls. Information Sciences **382–383**, 170–178 (2017)
5. Dudek, G.: Generating random weights and biases in feedforward neural networks with random hidden nodes. Information Sciences **481**, 33–56 (2019)
6. Dudek, G.: Improving randomized learning of feedforward neural networks by appropriate generation of random parameters. IWANN 2019, LNCS, vol. 11506, pp. 517–530. Springer, Heidelberg (2019).
7. Dudek, G.: Data-Driven Randomized Learning of Feedforward Neural Networks. arXiv:1908.03891v1 (2019).