Mansoura Engineering Journal

Volume 23 | Issue 3 Article 2

9-1-2020

A Radial Base Localized Distribution for Deterministic Function Approximation Neural Networks.

K. El-Serafi
Electrical Engineering Department ., Suez Canal University., Port-Said., Egypt.

K. Moustafa
Electrical Engineering Department ., Suez Canal University., Port-Said., Egypt.

Recommended Citation
El-Serafi, K. and Moustafa, K. (2020) "A Radial Base Localized Distribution for Deterministic Function Approximation Neural Networks.," *Mansoura Engineering Journal*: Vol. 23 : Iss. 3 , Article 2.
Available at: https://doi.org/10.21608/bfemu.2021.149971

A Radial Base Localized Distribution for Deterministic Function Approximation Neural Networks

التوزيع القطرى المحلى للشبكات العصبية المستخدمة فى التقريب المحدد للدوال

K. A. El-Serafi and K. Z. Moustafa.

Dept. of Elect. Eng., Suez Canal University, Port-Said, Egypt.

ملخص البحث . فى هذا البحث تمت دراسة إمكانية تخفيض كمية الحسابات اللازمة للخلايا العصبية من النوع DEFAnet ، وتم هذا التحسين فى الأداء اعتماداً على تنشيط عدد أقل من الخلايا العصبية تبعاً لتوزيع محلى بدلاً من تنشيط جميع الخلايا ، وقد أثبتت الدراسة إمكانية تعلم الشبكة العصبية المعدلة لأى دالة اختيارية بعمليات حسابية أقل وزمن تعلم أقل

Abstract— In this work, we investigate the possibility of reducing the computational requirements of the deterministic neural network function approximator, DEFAnet. The performance enhancement is based on activating smaller number of neurons according to certain localized distributions. The study shows the ability of this scheme to learn an arbitrary function. The interaction of localization with other network's parameters is discussed.

I. Introduction

Neural Networks may be used as approximators of continuous function, as they allow parallel computation of function values. They are usually trained by examples on the basis of local delta rules. Approximation capability of a neural network for an arbitrary function can be guaranteed with a three layer neural network with sigmoidal function on the hidden layer[3],[4],[5]. Networks with Radial Basis Function (RBF 's) have been used, recently, with success to achieve the same goal[5]. It is found that the boundedness condition on the sigmoidal activation function plays an essential role in approximation, as contrast to continuity or monotonity condition[4]. One of the main problems associated with the use of Neural Networks as a function approximator is the undeterminism on the topology required to reach network with capability of realizing required function. Another problem, is the probability of being trapped by a local minimum, even with correct topology.

The above two problems were, almost, solved by the introduction of the DEFAnet proposed by W. J. Dalmicht [1]. The only problem associated with this network is the computationally demanding cost of computation and storage that increase exponentially with the number of inputs and division count across each of the inputs. The paper is concerned with reducing the

Accepted July 29, 1998

computational requirements of the DEFAnet, while, preserving its generalization capability. The storage problem may be overwhelmed with hashing techniques.

This paper is organized as follows: We first review the definition of the DEFAnet. Second, we show the changes in formulation to adopt the concept of localization. Then, we test the system capability of learning a test function.

II. Definition of DEFAnet [2]

The DEFAnet is a multilayer feedforward network. Each neuron of which forms the sum of its inputs weighted by the respective synaptic strengths. The neuron outputs are monotonous functions of this sum.

The *first* layer of a DEFAnet may consist of n neurons with an identity activation function; serving as fan-out units. The first layer neurons may be indexed by $\nu = 1,\cdots,n$.

To define the connectivity and the activation function of the *second* layer neurons a set of l_ν points r_{λ_ν} with $\lambda_\nu = 1,\cdots,l_\nu$ is defined for each input. These points are distributed over the respective input range. The intervals between two adjacent points are thus double indexed by ν and κ_ν with $\kappa_\nu = 1,\cdots,l_\nu - 1$, where

$$\Delta r_{k_\nu \nu} = r_{(\kappa_\nu+1)\nu} - r_{k_\nu \nu}. \tag{1}$$

In the second layer of neurons, one neuron exists for each interval between adjacent points and bears the same index as the interval. Synapses to the second layer exist only between the ν^{th} neuron of the first layer and neurons indexed by ν in the second layer. All weights are fixed to 1. The activation functions of the neurons are the concatenation of two subtracted hyperbolic functions and a logarithmic function

$$\varphi_{\kappa_\nu \nu}(\gamma_{\kappa_\nu \nu}) = \frac{\ln(1 + \Delta r_{k_\nu \nu}\gamma_{\kappa_\nu \nu})}{\ln(1 + \Delta r_{k_\nu \nu})}, \tag{2}$$

where

$$\gamma_{\kappa_\nu \nu}(t) = \frac{1}{2} + \sqrt{\left(\frac{t - r_{k_\nu \nu}}{2\Delta r_{\kappa_\nu \nu}}\right)^2 + \alpha_\nu} - \sqrt{\left(\frac{t - r_{(k_\nu+1)\nu}}{2\Delta r_{k_\nu \nu}}\right)^2 + \alpha_\nu} \tag{3}$$

The hyperbolic functions depend on the smoothness parameters α_ν. The result is hard saturating sigmoidal function if $\alpha_\nu = 0$ and a soft saturating one if $\alpha_\nu \neq 0$.

In the *third* layer there are $\prod_{\nu=1}^{n} l_\nu$ neurons. They may be denoted by the vector index $\underline{\lambda}$. The synapses between the second and the third neuron layer are accordingly triple indexed and fixed to the following values

$$q_{\underline{\kappa},\nu} = \begin{cases} \ln(1 + \Delta r_{\kappa_\nu,\nu}) & \text{if } \kappa_\nu = \lambda_\nu - 1 \\ 0 & \text{otherwise} \end{cases} . \tag{3}$$

Hence, the connectivity in this synaptic layer is far from being complete. The activation function contains portions of the exponential function.

$$\chi_{\underline{\lambda}}(t) = \begin{cases} \exp(t) / a_{\underline{\lambda}} & \text{if } \underline{\lambda} = \underline{l} \text{ and } t \le \ln(a_{\underline{\lambda}}) \\ 1 & \text{otherwise} \end{cases} , \tag{4}$$

where

$$a_{\underline{\lambda}} = \prod_{\nu=1}^{n} \left(1 + \Delta r_{(\lambda_\nu - 1),\nu}\right)^{\eta(\lambda_\nu)} \qquad \text{with} \qquad \eta(t) = \begin{cases} 0 & \text{if } t \le 1 \\ 1 & \text{otherwise} \end{cases} . \tag{5}$$

These activation functions - similar to those of the second layer - are monotonous and their outputs are positive and not greater than 1.

The *fourth* layer of neurons contains the output neurons. They are fully connected to the neurons in the third layer. These synapses and only these are plastic. The activation function of each output neuron is identity.

It is possible to approximate arbitrary continuous functions by reducing the size of the intervals and thereby increasing the number of neurons in the second and the third layer. Since all plastic synapses are located in the third layer, their weights are also trained, e.g., by delta rule, with guarantee of convergence to a global optimum.

Due to the deterministic structure of the network and the linearity of the last neuron layer, it is possible to calculate the synapses weights in the last layer from desired function values by means of linear algebra [2]. In principle, it is required to form a matrix from the output function values of all in the third layer for all teaching input vectors. Then the pseudoinverse of this matrix is multiplied to the vector of teaching output values of each output neuron to yield the synaptic weights of that neuron. In case of quadratic matrix the number of plastic synapses corresponds to the number of desired function values. By calculation of the synaptic values and "down-loading" of the result, training may be shortened or replaced altogether.

III. Radial Base Localization of Distribution of the Second Layer Neurons.

The above formulation suffers the need of taking into account the activation of all neurons of second layer because of the possibility of all being activated. The problems associated with this configuration is the referencing of all synaptic weights of the fourth layer. The number of weights usually grows exponentially with the number of inputs of the network. Referencing all the weights not only makes the learning process very difficult but also limits the speed of the recall phase of the network for a well-trained network.

In this work, we devise a modification to the distribution of the activated neurons function to guarantee that the neurons activated in the second layer are small and being radially distributed around the input points.

The distributions suggested depends on the smoothing behavior required along certain input variable. If, behavior is required to be linear along with this variable, then, a count of two cells around the input variable activation center neuron may be considered. If the behavior is required to be smooth, a Gaussian distribution with a center of input variable and arbitrary adjustable variance is considered.

Let the active second layer neurons for output v is n_v. These activated neurons may be chosen under the assumptions that neurons with less than 1% of maximum activation are considered not active. Also, let first and last active neurons are denoted by s_v and f_v, respectively. Then, we define

$$\begin{aligned} s_v &= \max\left(0, p_v - \left\lfloor \frac{n_v}{2} \right\rfloor\right) \\ f_v &= \min\left(l_v, s_v + n_v\right) \end{aligned} \tag{6}$$

where $p_v = i$ for $r_{iv} < t \le r_{(i+1)v}$.

The activation of this scheme is defined as follows:

Let the smoothing value associated with a variable v is σ_v. Then for input v with normally distributed second layer output

$$\varphi_{\kappa_v v} = \begin{cases} \exp\left(-0.5 \times ((m_{k_v v} - t)/\sigma_v)^2\right) & \text{for } s_v < \kappa_v \le f_v \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

where $m_{k_v v} = \dfrac{\Delta r_{k_v v}}{2}$.

Accordingly, The synapses between the second and the third neuron layer are

$$q_{\lambda,v} = \begin{cases} 1 & \text{if } \kappa_v = \lambda_v - 1 \text{ and } s_v < \kappa_v < f_v \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

and the factors associated with this input v needed to compute $a_{\underline{\lambda}}$ in the fourth layer is replaced with $e^{\eta(t-1)}$

where

$$\eta(t) = \begin{cases} 0 & \text{if } t \le 1 \\ t & \text{otherwise} \end{cases}. \tag{9}$$

The summing points of the third layer neurons are replaced by multiplication points to ensure smooth transition on the output.

The performance of the network under these modifications is expected to be better. The reason is that as follows: Let S_1 and S_2 be two input vectors. If S_1 is near S_2, then many of the weights will be common. This will make the responses to both inputs are nearly the same. This property is called generalization, because it is similar to a biological organism to generalize from one learning experience to another [7].

However, If S_1 is not near S_2, then little or none of the weights will be common. This will make the response of S_1 and S_2 is independent. Consequently, it is very possible to adjust the response to each of the inputs easily. The original formulation suffers in this case because the adjustment for one of the inputs may conflict the requirements of the other. This problem is solved through higher number of learning iterations to find a globally best weight set. This solution in most cases proves to be demanding

III. Simulation and Results

Simulation has been conducted to test the learning capability for various neighborhoods, smoothness, and distributions, discussed in the previous sections.

The experiments is conducted to learn the following arbitrary smooth two-dimensional mathematical function

$$f(x,y) = \sin(x) \cdot \sin(y) \quad \begin{cases} 0^\circ \le x \le 360^\circ \\ 0^\circ \le y \le 180^\circ \end{cases}. \tag{10}$$

Weight update of the fourth layer, during all training experiment, is according to delta rule. The learning rate is taken proportional to the maximum number of updated weights. The

activation function of the fourth layer is a linear function. The points along each input is equally spaced by 10°.

For the original DEFAnet the training points are made on the corner points of the hypercubes of the inputs space. While, for the localized DEFAnet the training points are made on the center points of the hypercubes of the inputs space

The training is conducted for different learning rates and different smoothing parameters. The smoothing parameter determines the behavior of the function between the training points. This depends on the assumptions of the shape of the function if it is unknown. It is observed that the ability of the network to be trained is increased with reducing the smoothness parameter. Table I.

The learning rate changes up to 5 times without affecting the learning performance of the network significantly. The learning rate that gives best final r.m.s. errors at training points is 0.002 for all smoothness parameters.

Fig. 1 , 2 and 3 shows the learning behavior of the original DEFAnet with smoothing parameters 0.5, 0.1 and 0.0 respectively. Fig. 4, 5 and 6 shows the learning behavior to comparable modified networks with smoothing parameters 15, 10 and 8 respectively.

Comparing the results, it is observed that: The change of the learning rates changes the performance in more significant way; The r.m.s error is reduced; The r.m.s error drops in a higher speed. The less the number of effective neighbors the better the learning performance. but this affects the shape of function between the training points. This is beside the advantage fast learning and recalling phase due to the reduced number of referenced synaptic weights.

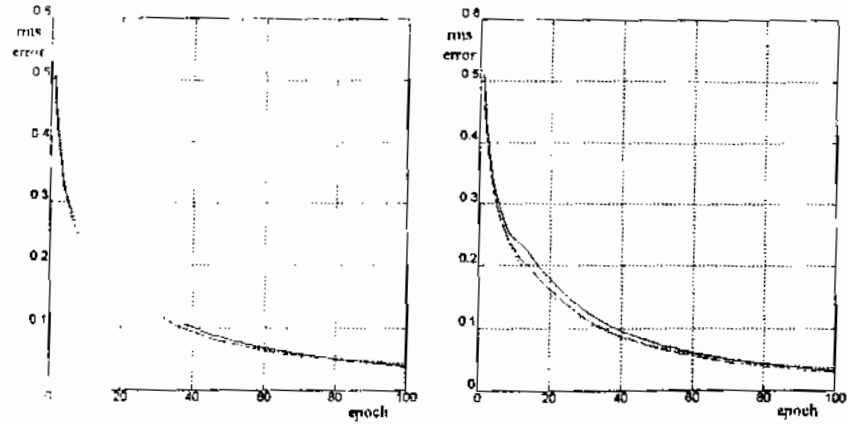

Fig. 1 Original DEFAnet with smoothing parameter $\alpha_s = 0.5$ Fig. 2 Original DEFAnet with smoothing parameter $\alpha_s = 0.1$

Table 1 Learning performance of localized DEFAnet vs. DEFAnet for different parameters and the required memory cells to reference.

Second Layer Distribution	Smoothness	Effective Neighbors	Referenced weights	Percentage of Referenced weights	Learning Rate	rms. error after 100 epoch
Saturation sigmoidal	$\alpha_i = 0.5$	all	19*37=703	100.0%	0.001	0.042268
					0.002	0.038183
					0.005	0.040009
	$\alpha_i = 0.1$	all	19*37=703	100.0%	0.001	0.034811
					0.002	0.029032
					0.005	0.031813
	$\alpha_i = 0.0$	all	19*37=703	100.0%	0.001	0.032811
					0.002	0.026496
					0.005	0.030073
Gaussian Distribution	$\sigma_i = 15.0$	7	7*7=49	6.97%	0.01437	0.026534
					0.02873	0.021808
					0.07184	0.026859
	$\sigma_i = 10.0$	5	5*5=25	3.56%	0.02812	0.015870
					0.05624	0.015231
					0.1406	0.018902
	$\sigma_i = 8.0$	5	5*5=25	3.56%	0.02812	0.012148
					0.05624	0.008293
					0.1406	0.006830

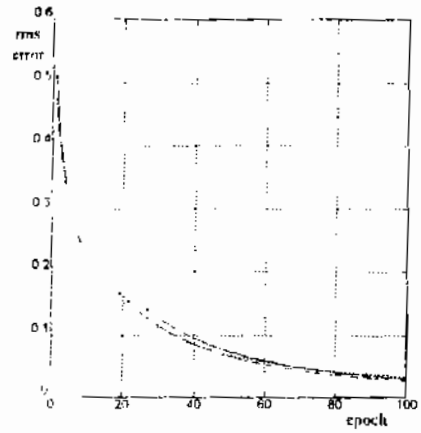

Fig. 3 Original DEFAnet with smoothing parameter $\alpha_s = 0.0$.

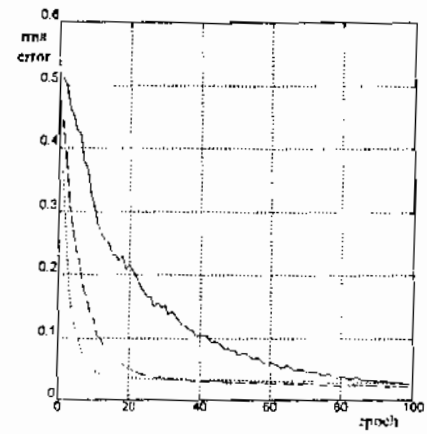

Fig. 4 Localized DEFAnet with $\sigma_l = 15$.

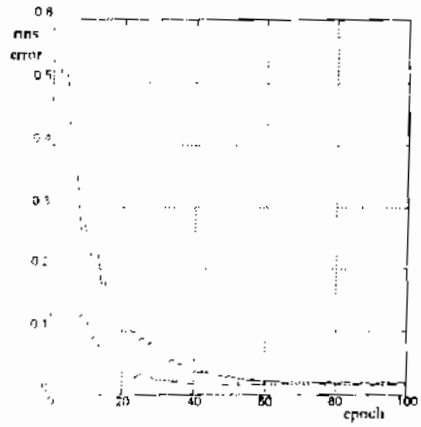

Fig. 5 Localized DEFAnet with $\sigma_l = 10$

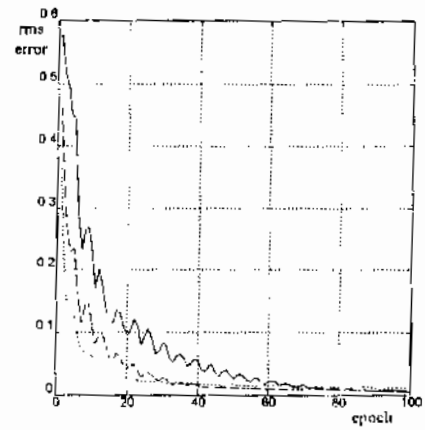

Fig. 6 Localized DEFAnet with $\sigma_l = 8$

V. Conclusions

The deterministic function approximation capability seems to be preserved with localization of the activated second layer neurons.

The number of referenced weights decreases logarithmic with decreasing the neighborhood. This makes the learning process much faster due to the small no of weights adjusted, also, the recall phase of the network is improved for the same reason.

The effective neighbors depend on the chosen smoothness associated with certain input. The smoothness depends on the desired behavior of the approximator, between the training points. The proposed modification still need rigorous proof of deterministic approximation capability

VI. REFERENCES:

[1] Wolfgang J. Daunicht, "DEFAnet a deterministic approach to function approximation by neural networks," *In: Proc. Intern Joint Conf. Neural Nets*, Washington, DC, 1990, Vol. I, 161-164.

[2] Wolfgang J. Daunicht, "DEFAnet - a Deterministic Function Approximator with asmoothness." *Artificial Neural Network*, T. Kohonon, K. M?kisara, O. Simula and J. Kangas (Editors), Elsevier Science Publishers B.V. (North Holland),1991.

[3] Tianping Chen and Hong Chen, "Approximations of Continuous Functionals by Neural Networks with Application to Dynamic systems," *IEEE Trans. on Neural Networks*, Vol. 4, No. 6, November 1993.

[4] Tianping Chen, Hong Chen, and Ruey-wen Liu, "Approximation capability in $C(\bar{R}^n)$ by Multilayer Feedforward Networks and Related problems." *IEEE Trans on Neural Networks*, Vol. 6, No. 1, January 1995.

[5] Andrew R. Webb, "Functional Approximation by Feedforward Networks: A Least-Squares Approaches to Generalization," *IEEE Trans. on Neural Networks*, Vol. 5, No. 3, May 1994

[6] Marios M. Polycarpou and Petros A. Ioannou, "Stable System Identification Using Neural Network Models," Neural Networks in Robotics, Editors George A. Bekey and kenneth Y. Goldberg, Kluwer Academic Publishers, 1993, pp. 147-164.

[7] Albus J. S., "A new approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC)," *Trans of the ASME, Journal of Dynamic Systems, Measurements and Control*, Vol. 97, No. 3, 1975, pp. 220-227.