Multithreshold neurons with smoothed activation functions

Vladyslav Kotsovsky

State University "Uzhhorod National University", Narodna Square 3, 88000, Uzhhorod, Ukraine

Abstract

The concept of non-linearity in a neural computation is provided by a proper activation function that plays a crucial role in the training process of the network and impacts its generalization ability. The model of smoothed multithreshold activation function (SMTAF) is proposed in order to improve the quality of modeling complicated nonlinear mappings by increasing the efficiency of the network learning compared to standard sigmoid-like activations or their modern modifications. The analysis of properties of three kinds of SMTAF is performed as well as its comparison with classical and modern activation functions. The performance of SMTAFs is estimated through two series of experiments: the approximation of a complex discontinuous function and the benchmark real-world regression problem. Obtained results suggest that a properly chosen SMTAF is capable to gain better accuracy in solving problems for which standard activation functions are inappropriate due to their limited efficiency.

Keywords

neural network, multithreshold activation function, multithreshold neuron, smoothing

1. Introduction

Neural-like models have numerous successful applications in intelligent systems [1] due to their great capacity to solve important real-world [2, 3], as well as scientific problems [4, 5]. This capacity is provided by the key property of the neural units—their ability to learn from examples that is crucial for the application of different machine learning (ML) techniques in the training of such units. A single neural unit has limited potential to solve tasks related to the function approximation or pattern classification [6]. But many properly connected units form a neural network (NN) whose capabilities increase significantly compared to a single neural unit. So called NN-based approach in ML leads to the development of high-performance AI systems, which use neural computation [7] in their learning or synthesis [6,8].

The main ability of NN is its capability to produce right responses via output values of last layer neurons when specified input stimulus are fed to the network input layer. This ensures the successful application of NNs by providing their ability to learn from the data [9] and to make predictions [10]. The last is supported by the capacity of a NN to extrapolate on the base of received and processed data [11, 12]. The proper choice of activation functions for NN nodes is one of the most important features ensuring the above-mentioned ability of NN to produce right reaction in response to given input patterns. Just the application of new continuous activations instead of step-wise ones in 1980s revived the interest in neural models and inspired the future tremendous achievements in the field. Problem-oriented activation functions warrant the non-linearity of a computation model [13] and can ensure the finiteness or convergence of the network learning, as well as the speed of the training process, its stability, and the network ability to infer from data [6,17].

ISW-2025: Intelligent Systems Workshop at 9th International Conference on Computational Linguistics and Intelligent Systems (CoLInS-2025), May 15–16, 2025, Kharkiv, Ukraine

* Corresponding author.

vladyslav.kotsovsky@uzhnu.edu.ua (V. Kotsovsky)

0000-0002-7045-7933 (V. Kotsovsky)

Usual continuous sigmoid-shaped activation functions such as logistic sigmoid, the hyperbolic tangent (tanh), as well as their rectified-based counterparts such as the rectified linear unit (ReLU), leaky ReLU [6] or exponential linear unit (ELU) and its modification scaled ELU [7] have their advantages [6,7]; however, they also have significant limitations [14]. E.g., despite the prototype in biological neurons sigmoid-shaped activations perform often badly in the learning of deep NN because they suffer from the vanishing gradient problem [15,16], as well as they have a saturation drawback [7]. The widespread ReLU activation function is vulnerable to the "dying ReLUs" problem [16]. More modern functions like ELU, SELU, Swish, GELU or Mish improve efficiency [7,14,17] but still are unable to approximate precisely dependencies characterized by rapid changes [18,19].

The drawbacks of classical sigmoid and ReLU activation functions were stated in [6,15, 22]. In order to overcome these limitations modified activations such as GELU, Swish or Mish were proposed [7,14,16,17]. It is well-known [6] that wide range of activation functions is suitable for backpropagation-based learning rules, where only conditions of continuity of activation function /as well as its derivative would be satisfied. But the almost 40 years long practice of the application of backpropagation proves that the choice of the activation function has the significant impact on the behavior of the learning process. Particular activations are good enough for one ML problems and are bad for other ones [6]. This observation has theoretical foundation in famous *No Free Lunch Theorem*, which, speaking informally, states that "we cannot achieve positive performance on some problems without getting an equal and opposite amount of negative performance on other problems" [6]. Therefore, we would choose activation functions depending on the particular ML problem [20] by using the prior knowledge concerning the problem scope. For example, consider the problem of the approximation of functions with many peaks and rapid multi-level changes [20, 22]. Standard activations are not well appropriate to model such mappings [23]. This makes difficult the learning of NN to solve very common tasks concerning the processing complicated signal transformations, which are not smooth enough or have a fluctuated gradient [17]. Thus, the design of activation functions appropriate in the training of NN for such difficult problems is an important task, as well as its tuning in order to improve the ability of NN to produce more precise predictions.

Multithreshold activation functions (MTAF) provide the one of possible solutions of this problem [13]. Their use can increase the quality of modeling significantly nonlinear functions. This approach is based on the ability of multithreshold activations to represent more precisely the change of data using numerous levels of discretization by the adaptive tuning of the thresholds. It allows to process properly discrete patterns as well as to enhance the efficiency of the network learning [8].

Models of MTAF were proposed in early works devoted to the application of threshold logic in pattern recognition as a more powerful alternative for Heaviside step function (bibliographical and historical remarks can be found in [13]). Capabilities of multithreshold neuron (MTN) were studied in [20] and [21]. The learning algorithm for multi-valued MTN was proposed in [8]. MTN-based NN architectures were introduced in [13, 19]. The synthesis algorithm for MTN-based classifier was proposed in [21]. The application of bithreshold neurons with the binary outputs in pattern recognition was studied in [21,23]. Deeper hybrid NN architectures with heterogenous hidden layers were proposed in [18]. The synthesis procedure for MTN-based NN considered in [21]. It should be noted that only discrete activation functions were used in all above-mentioned multithreshold units and NN. Thus, they are almost not applicable for problems dealing with the approximation of general real-valued functions. As it is stated in [24], the application of multithreshold models in the solution of regression problems was unknown. The reference [25] gives the first example of the use of multithreshold approach in the design of a bithreshold NN regressor. As mentioned in [10], this model of NN regressor can be extended to the case of an arbitrary number of thresholds.

All known examples of the multithreshold approach in neural computation concerns only neural units and networks employing discrete-valued activation functions. The disadvantage of such activation is that the modern optimization algorithms based on the use of gradient vector of the network error function are not applicable in the case of such MTAF. The *main goal of current research* is the extension of multithreshold approach to the case of continuous differentiable activation functions. This allows to employ the backpropagation-like training techniques to the learning of MTN-

based NN. The combination of MTAF and *sigmoid smoothing* shall be used in order to combine the advantages provided by the power of multithreshold approach as well as the effectiveness of the gradient descent optimization.

2. Models and methods

2.1. Model of neural unit with a smoothed multithreshold activation function

2.1.1. Definition of smoothed multithreshold activation function

Let $\mathbf{t}=(t_1,t_2,\ldots,t_k)$ be a k-dimensional real vector of strictly increasing thresholds: $t_1<t_2<\ldots<t_k$, $\mathbf{v}=(v_0,v_1,\ldots,v_k)\in\mathbf{R}^{k+1}$ be an arbitrary value vector. Consider a discrete-valued function

$$g_{\mathbf{t},\mathbf{v}}(x)=\begin{cases} v_0, & \text{if } x<t_1,\\ v_1, & \text{if } t_1\le x<t_2,\\ \vdots & \\ v_{k-1}, & \text{if } t_{k-1}\le x<t_k,\\ v_k, & \text{if } x\ge t_k. \end{cases} \tag{1}$$

Step function (1) is well suitable to approximate different nonlinear dependencies. It should be noted that in the case $\mathbf{v}=(0,1,\ldots,k)$ $g_{\mathbf{t},\mathbf{v}}(x)$ reduces to the activation function of multi-valued k-threshold neural unit [20], whereas in the case $v_0=v_2=\ldots=v_{2[k/2]}=0,\ v_1=v_3=\ldots=v_{2[(k+1)/2]-1}=1$ (where the floor function $[x]$ denotes the integer part of the number x) (1) becomes the activation function binary-valued k-threshold neuron. Nevertheless, it is possible to use (1) as activation function and, thus, to treat it as MTAF.

The most significant property of MTAF (1) is its discreteness. Therefore, MTAF (1) cannot be employed in mode network architectures for which continuous activations are required in order to maintain the gradient-based learning techniques. This drawback of (1) it is caused by zero value of the derivative of MTAF (1) (moreover, $g_{\mathbf{t},\mathbf{v}}(x)$ is even undefined, if $x\in\{t_1,t_2,\ldots,t_k\}$). Consequently, backpropagation approach to the training is not directly applicable to NN containing nodes with such MTAF, because zero value of gradient vector makes weight changes impossible.

Thus, we need apply some mapping in order to transform MTAF to the form that is acceptable for gradient-based optimization. I.e., the transformed MTAF should be smooth enough, namely, it may satisfy the main condition—the existence of a continuous nonzero derivative. Consider the following transformation:

$$s_{\mathbf{t},\mathbf{v},a}(x)=v_0+\sum_{i=1}^{k}(v_i-v_{i-1})\sigma(a(x-t_i)), \tag{2}$$

where a is a positive scalar parameter defining the steepness of the transformed function in the neighborhood of thresholds and $\sigma(x)=\dfrac{1}{1+e^{-x}}$ is the *standard logistic sigmoid* [6]. It is natural to call a transformation $s_{\mathbf{t},\mathbf{v},a}(x)$ a *logistic sigmoid smoothing.*

Consider an example. Let us employ the sigmoid smoothing to the MTAF $g_{\mathbf{t},\mathbf{v}}(x)$ having four thresholds −5, −1, 2, 6 and five distinct values 3, −2, 2, −1, 5. The results of sigmoid smoothing (2) performed in the cases $a\in\{0.5,1,2,3\}$ are presented in Figure 1, where vertical dashed lines indicated regions with constant values of MTAF (which are depicted using horizontal dotted lines).

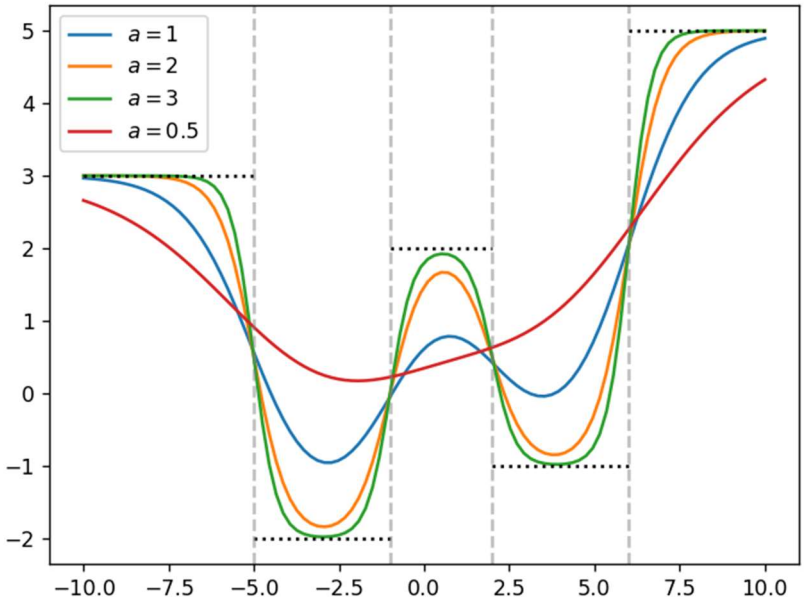

Figure 1: Logistic sigmoid smoothing of MTAF for different values of steepness parameter a

By analyzing curves shown in Figure 1, it is possible to conclude that in the cases $a = 0.5$ the smoothing is too harsh and obtained curve does not adequately represent the initial step function. If $a = 1$, then logistic sigmoid was capable to perform more adequate smoothing, but the middle peak values corresponding to v_1, v_2, v_3 are almost equalized and are too distant from the corresponding values of initial MTAF. In the case $a \in \{2, 3\}$ the behavior of the smoothed function is more appropriate and the shapes of obtained curves are quite similar to the curve of the step function.

The transformation (2) has been obtained using standard logistic smoothing. But there are many other sigmoid functions [16], which can be also useful for the smoothing of MTAF. E.g., consider the following smoothing obtained by the use hyperbolic tangent instead of logistic sigmoid:

$$h_{\mathbf{t},\mathbf{v}}(x) = \frac{1}{2}\left(v_0 + \sum_{i=1}^{k} (v_i - v_{i-1}) \tanh(x - t_i) + v_k \right). \tag{3}$$

This kind of smoothing will be further referred as *tanh smoothing*.

Consider one more kind of sigmoid smoothing obtained using a rational sigmoid $r(x) = \frac{x}{1 + |x|}$. Rational sigmoid $r(x)$ has the same range of a function as tanh x. Thus, the similar equation for the smoothing should be used. Let us define the *rational smoothing* in the following way:

$$r_{\mathbf{t},\mathbf{v}}(x) = \frac{1}{2}\left(v_0 + \sum_{i=1}^{k} (v_i - v_{i-1}) r(x - t_i) + v_k \right). \tag{4}$$

All functions (2)-(4) use the similar idea to the transformation of MTAF—the application of sigmoid functions to approximate "leaps" of step function by smooth functions with almost surely nonzero derivatives. Hence, it is natural to call a result of one of such transformations a *smoothed multithreshold activation functions* (SMTAF).

The behavior of different kinds of SMTAF is demonstrated in Figure 2, where the curves are shown for transformation results of the same function $g_{\mathbf{t},\mathbf{v}}(x)$ as was shown in Figure 1, which are obtained using logistic sigmoid, tanh and rational smoothing, respectively. Note that the effect of rational smoothing is stronger than the one provided by its tanh counterpart.

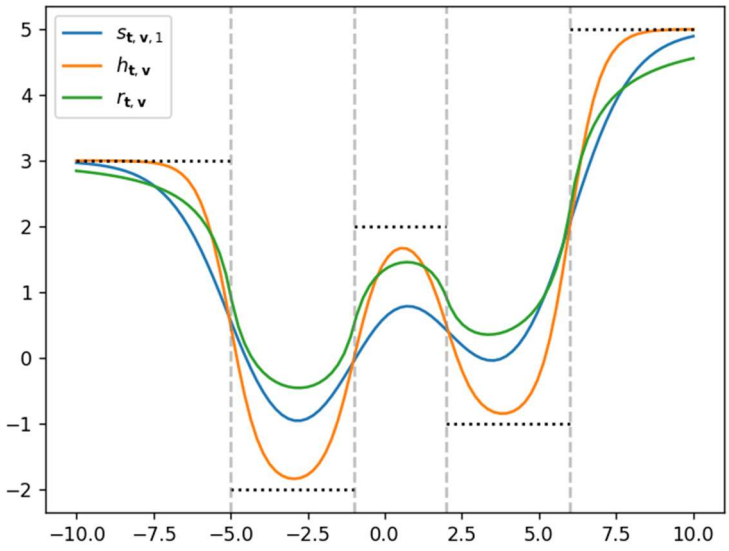

Figure 2: Comparison of results of logistic sigmoid, tanh and rational smoothing

2.1.2. Properties of SMTAF

Let us establish basic properties of SMTAFs. It is easy to prove that if $\varphi \in \{s_{\mathbf{t},\mathbf{v},a}, h_{\mathbf{t},\mathbf{v}}, r_{\mathbf{t},\mathbf{v}}\}$, then $\lim\limits_{x\to-\infty}\varphi(x)=v_0$, $\lim\limits_{x\to+\infty}\varphi(x)=v_k$. Consider first the case $\varphi = s_{\mathbf{t},\mathbf{v},a}$. From $\lim\limits_{x\to-\infty}\sigma(x)=0$ we can infer that

$$\lim_{x\to-\infty} s_{\mathbf{t},\mathbf{v},a} = v_0 + \sum_{i=1}^{k}(v_i - v_{i-1}) \lim_{x\to-\infty}\sigma(ax - t_i) = v_0.$$

$\lim\limits_{x\to+\infty}\sigma(x)=1$ implies that

$$\lim_{x\to+\infty} s_{\mathbf{t},\mathbf{v},a} = v_0 + \sum_{i=1}^{k}(v_i - v_{i-1}) \lim_{x\to-\infty}\sigma\big(a(ax - t_i)\big) = v_0 + \sum_{i=1}^{k}(v_i - v_{i-1}) = v_k.$$

Consider $\varphi = r_{\mathbf{t},\mathbf{v}}$. Then $\lim\limits_{x\to-\infty} r(x) = -1$ allows us to conclude that

$$\lim_{x\to-\infty} r_{\mathbf{t},\mathbf{v}}(x) = \frac{1}{2}\left(v_0 + \sum_{i=1}^{k}(v_i - v_{i-1}) \lim_{x\to-\infty} r(x) + v_k\right) = \frac{1}{2}\left(v_0 - \sum_{i=1}^{k}(v_i - v_{i-1}) + v_k\right) = v_0.$$

Similarly, we can infer from $\lim\limits_{x\to+\infty} r(x) = 1$ that

$$\lim_{x\to-\infty} r_{\mathbf{t},\mathbf{v}}(x) = \frac{1}{2}\left(v_0 + \sum_{i=1}^{k}(v_i - v_{i-1}) + v_k\right) = \frac{1}{2}(v_0 + v_k - v_0 + v_k) = v_k.$$

Note that the proof of the similar properties in the case $\varphi = h_{\mathbf{t},\mathbf{v}}$ is almost identical.

Let us show that tanh smoothing (3) can be performed using the logistic sigmoid smoothing (2). This fact is the direct consequence of the well-known [6] relation between the standard sigmoid and the tangent hyperbolic: $\tanh x = 2\sigma(2x) - 1$. Therefore,

$$\begin{aligned} h_{\mathbf{t},\mathbf{v}}(x) &= \frac{1}{2}\left(v_0 + \sum_{i=1}^{k}(v_i - v_{i-1})\big(2\sigma(2(x - t_i)) - 1\big) + v_k\right) \\ &= v_0 + \frac{1}{2}\sum_{i=1}^{k}(v_i - v_{i-1})2\sigma(2(x - t_i)) - \frac{1}{2}\left(v_0 + \sum_{i=1}^{k}(v_i - v_{i-1}) - v_k\right). \end{aligned}$$

It is easy to show that the sum in the last parentheses is equal to zero. Therefore,

$$h_{\mathbf{t},\mathbf{v}}(x) = v_0 + \sum_{i=1}^{k} (v_i - v_{i-1})\sigma\big(2(x - t_i)\big) = s_{\mathbf{t},\mathbf{v},2}(x).$$

Despite the fact that an arbitrary MTAF can be smoothed by using transformation (2)-(4), the one of most important applications of smoothing is the case when the such transformation is applied to the monotonic step function (1), i.e., when the vector $\mathbf{v} = (v_0, v_1, \ldots, v_k)$ from (1) satisfies inequalities $v_0 < v_1 < \ldots < v_k$. In this case All kinds of sigmoid used in (2)–(4) are increasing,. This implies that transformations (2)-(4) result in increasing SMTAFs, because all differences $v_i - v_{i-1}$ are positive as well as all considered sigmoids are increasing.

Consider the example of smoothing of increasing MTAF $g_{\mathbf{t},\mathbf{v}}(x)$ with same four thresholds −5, −1, 2, 6 and five ordered values −2, −1, 2, 3, 5. The plots of curves obtained after logistic sigmoid smoothing in cases $a = 1$ and $a = 3$ as well as curves of tanh and rational sigmoid SMTAF are presented in Figure 3, where horizontal lines indicates values of initial MTAF $g_{\mathbf{t},\mathbf{v}}(x)$.

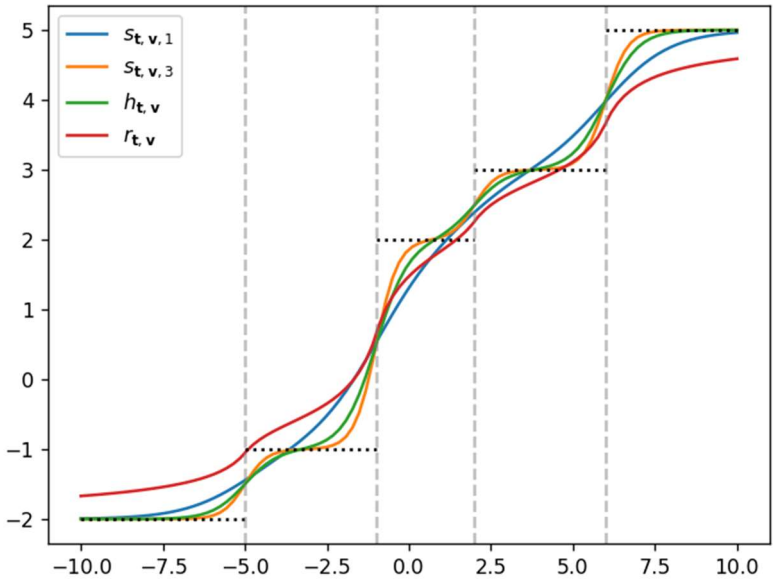

Figure 3: Logistic sigmoid, tanh and rational smoothing of increasing function

It is possible to conclude from Figure 3 that the logistic sigmoid smoothing was unsuccessful in the case $a = 1$, because curve of corresponding SMTAF has the shape of basic logistic sigmoid and steps of MTAF are almost neglected, whereas the result of tanh smoothing preserves the main tendencies of $g_{\mathbf{t},\mathbf{v}}(x)$. Rational smoothing (4) performs better than $s_{\mathbf{t},\mathbf{v},1}$ but its curve does not seem so similar to $g_{\mathbf{t},\mathbf{v}}(x)$ as $s_{\mathbf{t},\mathbf{v},3}$ and $h_{\mathbf{t},\mathbf{v}}$, respectively.

2.2. Learning of neural units with smoothed multithreshold activation

Consider issues related to the application of SMTAFs in the learning algorithms based on the gradient descent or its generalization. It is well-known that backpropagation algorithm uses the chain rule of the calculation of gradient of the network loss function. The one of calculation steps is the computing of the partial derivatives of loss function with respect to the activation function of the given neuron. Thus, we need formulas for these derivatives when dealing with SMTAFs. Let us compute the derivatives of SMTAFs obtained as results of transformation (2)-(4). By using the well-known equality $\sigma'(x) = \sigma(x) \cdot \big(1 - \sigma(x)\big)$, we can infer that

$$\frac{d}{dx}s_{\mathbf{t},\mathbf{v},a}(x)=a\sum_{i=1}^{k}(v_i-v_{i-1})\sigma\big(a(x-t_i)\big)\big(1-\sigma\big(a(x-t_i)\big)\big). \tag{5}$$

Consider the derivative of the result of tanh smoothing. It follows from $(\tanh x)'=1-\tanh^2 x$ that

$$\frac{d}{dx}h_{\mathbf{t},\mathbf{v}}(x)=\frac{1}{2}\frac{d}{dx}\left(v_0+\sum_{i=1}^{k}(v_i-v_{i-1})\tanh\big(x-t_i\big)+v_k\right)=\frac{1}{2}\sum_{i=1}^{k}(v_i-v_{i-1})\big(1-\tanh^2\big(x-t_i\big)\big). \tag{6}$$

It is easy to verify that $\frac{d}{dx}r_{\mathbf{t},\mathbf{v}}(x)=\frac{1}{\big(1+|x|\big)^2}=\big(1-r_{\mathbf{t},\mathbf{v}}\big(|x|\big)\big)^2$.

Therefore,

$$\frac{d}{dx}r_{\mathbf{t},\mathbf{v}}(x)=\frac{1}{2}\sum_{i=1}^{k}(v_i-v_{i-1})\big(1-r^2\big(|x-t_i|\big)\big). \tag{7}$$

Thus, it is possible to conclude that smoothing transformations (2)-(4) have very simple derivatives (5)-(7) whose calculation does not require an additional efforts compared to the calculation of SMTAF.

It should be also mentioned that if value vector $\mathbf{v}=(v_0,v_1,\ldots,v_k)$ of MTAF (1) is increasing, then derivatives (5)-(7) are positive in every point of their domain of a function.

Notice that there are two possible ways of the treating of thresholds $t_1,t_2,\ldots,t_k$ used in SMTAF. The simplest one assumes that these thresholds are unlearnable, i.e., they are not changed during the training epochs. The second approach treats $t_1,t_2,\ldots,t_k$ as changeable parameters, which are updated on each correction step (like biases of classical feed-forward NN models). Under this assumption we need also the formulas for partial derivatives of SMTAFs (2)-(4) with respect to variables $t_1,t_2,\ldots,t_k$:

$$\frac{\partial}{\partial t_j}s_{\mathbf{t},\mathbf{v},a}=-a\frac{d}{dx}s_{\mathbf{t},\mathbf{v},a}=-a^2\sum_{i=1}^{k}(v_i-v_{i-1})\sigma\big(a(x-t_i)\big)\big(1-\sigma\big(a(x-t_i)\big)\big),\ \ j=1,\ldots,k,$$

$$\frac{\partial}{\partial t_j}h_{\mathbf{t},\mathbf{v}}(x)=-\frac{d}{dx}h_{\mathbf{t},\mathbf{v}}(x)=-\frac{1}{2}\sum_{i=1}^{k}(v_i-v_{i-1})\big(1-\tanh^2\big(x-t_i\big)\big),\ \ j=1,\ldots,k,$$

$$\frac{\partial}{\partial x}r_{\mathbf{t},\mathbf{v}}(x)=-\frac{d}{dx}r_{\mathbf{t},\mathbf{v}}(x)=-\frac{1}{2}\sum_{i=1}^{k}(v_i-v_{i-1})\big(1-r^2\big(|x-t_i|\big)\big),\ \ j=1,\ldots,k.$$

Above equations allow to integrate threshold corrections in the backpropagation process.

3. Experiment and results

3.1. Description of the learning framework

In order to estimate the capability of SMTAF in comparison with other popular activations, two series of experiments were performed. In the first series the ability of NN with different activation functions to solve the approximation task was considered. The second series consisted of the study of NNs provided with same activations to solve the regression problem.

The following hard discontinuous piecewise function containing the combination of constant and harmonic parts was used in the approximation task:

$$f(x)=\begin{cases}\sin(4\pi x), & \text{if } -0.5\le x<0.5,\\ 0.5\operatorname{sgn}x, & \text{if } 0.5\le|x|<1,\\ \operatorname{sgn}x, & \text{if } x\ge 1.\end{cases} \tag{8}$$

The values of function (8) computed in 1000 uniformly distributed in [-1.2, 1.2] points form the training set.

The approximation task was solved on the multilayer perceptron (MLP) with two hidden layers containing 128 and 64 neurons, respectively. Many classical and popular modern activation were tried: logistic sigmoid, tanh, ReLU, Swish, GELU, Mish [6, 22] as well as smoothed multithreshold functions (2)-(4). A single output linear neuron was used. The thresholds in SMTAF (2)-(4) were treated as hyperparameters of the neural model. Thus, they were not learnt during backpropagation, and random search [26] was performed instead. It was made over the distribution of thresholds on the segment [-2,2] and the distribution of output values of basic MTAF on the same segment Adam optimizer [6] was employed for the minimization of mean squared error (MSE), which served as NN loss function [6]. 5-fold cross-validation was used in order to obtain consistent results.

The second series of experiments treated the small "liver-disorders" real-world dataset available in OpenML machine learning repository [27]. This dataset has five input features and a single target feature. It contains 345 training instances and belongs to the so-called regression problems. 20% of instances were used as a test set. In order to employ mean absolute percentage error (MAPE) metric the constant 10 was added to every target value during the preprocessing. The 5-16-16-1 MLP architecture was used.

3.2. Results

The simulation results proved that the considered approximation task is hard enough for chosen activations and data obtained for different activation differ considerably. It is natural that behavior of SMTAFs was studied most carefully by tuning threshold and value vectors. Plots of the NN outputs with above-mentioned activations are shown in Figure 4, where the dashed line depicts the plot of the function (8).

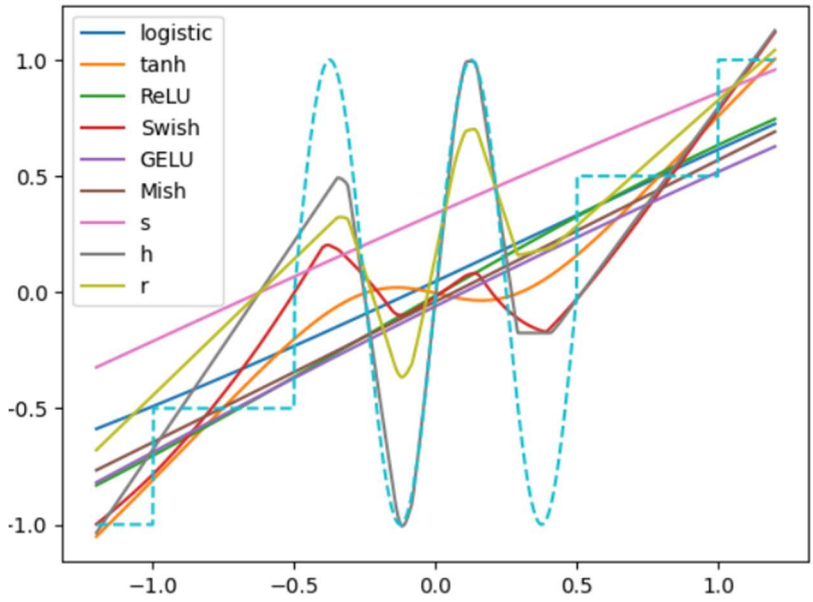

Figure 4: Result of the approximation of function (8) using standard and SMT activation functions.

By analyzing presented curves, we can conclude that logistic sigmoid, tanh, ReLU, GELU, Mish and SMTAF (2) failed completely on the both "harmonic" and step-wise parts of the function (8). Swish and tanh smoothed MTAF were more successful and could partially approximate the target curve on the part of its domain. In general, the tangent-based SMTAF $h_{t\,v}$ with only two near-symmetric thresholds achieved the best value 0.096 of the MSE metric and outperformed all other activations for more than 37%.

Consider the second series of experiments. Measured performance metrics are given in Table 1.

Table 1
Performance results of MLP regressors on liver-disorders dataset

	Logistic	tanh	ReLU	Swish	GELU	Mish	$s_{\mathbf{t},\mathbf{v},\mathbf{1}}$	$h_{\mathbf{t},\mathbf{v}}$	$r_{\mathbf{t},\mathbf{v}}$
MSE	10.66	10.61	11.89	17.13	11.85	17.42	9.05	10.98	11.03
MAPE	19.66	19.54	19.49	21.49	19.76	22.13	17.49	19.28	19.51

By analyzing above data, it is possible to conclude that the activation provided by the logistic sigmoid smoothing (2) overperformed all other activations by both metrics, whereas tanh smoothing (3) had the second-best result by MAPE metric. Notice also that the best performance of SMTAF was obtained in the case of the smoothing of non-monotonic bithreshold step function (1).

4. Discussions

Simulation results suggest that the smoothing of discrete step functions could result in the very powerful activation functions. Data presented in the previous chapter prove that SMTAFs are capable to overperform standard neuron activations in problems involved both synthetic and real-world datasets. But the application of SMTAF faces two main challenges concerning the choice of the threshold vector t and the value vector v, respectively. Future efforts would be made to Find solution of the problem of the choice of an appropriate combination of thresholds and values of SMTAF or the effective embedding their search in the backpropagation learning process.

5. Conclusions

The transformation of multithreshold functions using sigmoid smoothing has been considered in the paper. The impact of SMTAF to the approximation capability of NN has been studied. The empirical comparison with both traditional (logistic sigmoid, tanh and ReLU) and modern activations like Swish, Mish and GELU suggests that the application of SMTAF ensures often faster convergence and more adequate representation of complicated dependencies. The adaptive problem-based approach in choice of parameters of SMTAFs is very important for their successful application, because it allows the fine tuning of NN coefficients in the case of the presence of high degree of nonlinearity in dependencies between input and output features.

Experimental results obtained for two datasets confirm that the use of SMTAFs may improve the capability of NN and reduce the cost of the network training. Therefore, the above research confirmed the conjecture that the application of smoothed multithreshold activation function is promising enough for the learning of MLPs. Further studies are necessary to find the concrete combinations of thresholds and values of MTAF, which are capable to succeed in the NN training for a wide range of regression problems, as well as to investigate preferable approaches of the smoothing of MTAF.

Declaration on Generative AI

The author has not employed any Generative AI tools.

References

[1] E.H. Houssein et al., Soft computing techniques for biomedical data analysis: open issues and challenges, Artificial Intelligence Review 56 (2023): 2599–2649.

[2] V.K. Venkatesan, M.T. Ramakrishna, A. Batyuk, A. Barna, B. Havrysh, High-Performance artificial intelligence recommendation of quality research papers using effective collaborative approach, Systems 11.2 (2023): 81. doi:10.3390/systems11020081.

[3] I. Izonin et al., A hybrid two-ML-based classifier to predict the survival of kidney transplants one month after transplantation, in: CEUR Workshop Proceedings, volume 3609, 2023, pp. 322–331.

[4] S. Vladov, R. Yakovliev, M. Bulakh, V. Vysotska, Neural network approximation of helicopter turboshaft engine parameters for improved efficiency, Energies 17.9 (2024): 2233.

[5] J. Kaur, J. Verma, Past, present and future of computational intelligence: a bibliometric analysis, in: AIP Conference Proceedings 2916(1), 2023, 020001.
[6] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 3rd ed., O'Reilly Media, Sebastopol, CA, 2022.
[7] D. Misra, Mish: a self regularized non-monotonic neural activation function, arXiv: Machine Learning, 2019. URL: https://arxiv.org/vc/arxiv/papers/1908/1908.08681v1.pdf.
[8] V. Kotsovsky, Learning of multi-valued multithreshold neural units, in: CEUR Workshop Proceedings, volume 3688, 2024, pp. 39–49.
[9] M. Lupei, O. Mitsa, V. Sharkan, S. Vargha, N. Lupei, Analyzing Ukrainian media texts by means of support vector machines: aspects of language and copyright, in: Z. Hu., I. Dychka, M. He (Eds.), Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023, Lecture Notes on Data Engineering and Communications Technologies, volume 181, Springer, Cham, 2023, pp. 173–182.
[10] S. Moon, Y. Kim, Mounting angle prediction for automotive radar using complex-valued convolutional neural network, Sensors 25.2 (2025): 353.
[11] I. Izonin et al., Experimental evaluation of the effectiveness of ANN-based numerical data augmentation methods for diagnostics tasks, in: CEUR Workshop Proceedings, volume 3038, 2021, pp. 223–232.
[12] F. Geche et al., Synthesis of time series forecasting scheme based on forecasting models system, in: CEUR Workshop Proceedings, volume 1356, 2015, pp. 121–136.
[13] V. Kotsovsky, A. Batyuk, Multithreshold neural units and networks, in: Proceedings of IEEE 18th International Conference on Computer Sciences and Information Technologies, CSIT 2023, Lviv, Ukraine, 2023, pp. 1-5.
[14] P. Ramachandran, B. Zoph, Q. V. Le, Swish: a self-gated activation function, arXiv: Neural and Evolutionary Computing, 2017.
[15] S. R. Dubey, S. K. Singh, B. B. Chaudhuri, Activation functions in deep learning: a comprehensive survey and benchmark, Neurocomputing 503 (2022): 92–108.
[16] A. Apicella, F. Donnarumma, F. Isgrò, R. Prevete, A survey on modern trainable activation functions, Neural Networks 138 (2021): 14–32.
[17] I. Jahan, M.F. Ahmed, M.O. Ali, Y.M. Jang Self-gated rectified linear unit for performance improvement of deep neural networks, ICT Express 9.3 (2023): 320–325.
[18] V. Kotsovsky, Hybrid 4-layer bithreshold neural network for multiclass classification, in: CEUR Workshop Proceedings, volume 3387, 2023, pp. 212–223.
[19] N. Jiang, Y. X. Yang, X. M. Ma, and Z. Z. Zhang, Using three layer neural network to compute multi-valued functions, in 2007 Fourth International Symposium on Neural Networks, June 3-7, 2007, Nanjing, P.R. China, Part III, LNCS 4493, 2007, pp. 1-8.
[20] Z. Obradovic, I. Parberry, Learning with discrete multivalued neurons, Journal of Computer and System Sciences 49 (1994): 375–390.
[21] V. Kotsovsky, Synthesis of multithreshold neural network classifier, in: CEUR Workshop Proceedings, volume 3711, 2024, pp. 75–88.
[22] T. Szandała, Review and comparison of commonly used activation functions for deep neural networks, Studies in Computational Intelligence, volume 903, pp. 203-224, 2021.
[23] V. Kotsovsky, F. Geche, A. Batyuk, Artificial complex neurons with half-plane-like and angle-like activation function, in: Proceedings of 10th International Conference on Computer Sciences and Information Technologies, CSIT 2015, Lviv, Ukraine, 2015, pp. 57–59.
[24] I. Izonin et al., Regression-based model for predicting simulated vs actual building performance discrepancies, in: Procedia Computer Science, volume 251, 2024, pp. 633–638.
[25] V. Kotsovsky, A. Batyuk, Towards the Design of Bithreshold ANN Regressor, in: 19th IEEE International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2024. Lviv, October 16–19, 2024. To appear.
[26] Scikit-learn: Machine learning in Python, 2025. URL: https://scikit-learn.org/stable/.
[27] OpenML: A worldwide machine learning lab, 2025. URL: https://www.openml.org.