

# Synthesis of multithreshold neural network classifier

Vladyslav Kotsovsky\*

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

## Abstract

Issues related to the application of multithreshold neural units in machine learning are considered in the paper. A labeled modification of the model of binary-valued multithreshold neuron is proposed. The 2-layer feedforward network architecture with the hidden layer consisting of multithreshold neurons is treated. The multicategory classifier is designed on the base of such architecture. The supervised algorithm is proposed for the synthesis of multithreshold neural networks. Simulation results are presented related to the performance of multithreshold classifier on real-world dataset compared to some popular classifiers. The impact of algorithm hyperparameters on the classifier performance, as well as on its representational capacity is discussed.

## Keywords

Multithreshold neural unit, artificial neural network, classification, machine learning

## 1. Introduction

Neural networks play extremely important role in machine learning [1] and intelligent data proceeding [2, 3]. They are used as efficient components of modern smart hardware [4] and software solutions [5, 6]. Powerful capabilities of artificial neural network (NN) are provided by the possibility of the use of problem-oriented network architecture [7], appropriate supervised or unsupervised learning techniques [8, 9] and the proper choice of hyperparameters of the learning algorithm [2].

During few past decades we were witnesses of numerous rises and falls of the interest in neural computations. Early neural-like models (e.g., a model of McCulloch and Pitts artificial neuron) used linear threshold units in which binary-valued Heaviside step function [9] was used. These models were inspired by the brain study of the structure of biological neurons [2]. But the limited capacity of a single threshold unit [10] along with difficulty in the design of learning algorithms for threshold networks even in case of the relatively small number of units were stumbling blocks for an efficient application of threshold NN in machine learning and caused the decline of interest in them [11].


In order to overcome these drawbacks of early models of neurons, many improvements were proposed [2, 12] intended to boost the recognition capability of a single threshold unit and make easier the process of the network learning [9]. They can be divided into two big categories. The

---

*MoMLeT-2024: 6th International Workshop on Modern Machine Learning Technologies, May, 31 - June, 1, 2024, Lviv-Shatsk, Ukraine*

\* Corresponding author.

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

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



© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

first contains models using the more complicated approach to the aggregation of the input signals of the neural unit instead of the weighted sum of inputs in linear threshold units, e.g., polynomial threshold units [11, 13]. The second category contains models in which the basic threshold function is replaced by its more sophisticated modifications [14]. This category is much more useful because it allows to use diverse learning techniques adapted to the particular problem or application [15, 16].

The models with multithreshold activation function were one of first in the second category [17]. A multithreshold neuron has weight vector  $\mathbf{w} = (w_1, \dots, w_n) \in \mathbf{R}^n$  associated with inputs  $x_1, \dots, x_n$  and threshold vector  $\mathbf{t} = (t_1, \dots, t_k) \in \mathbf{R}^k$  instead of a single threshold used in McCulloch and Pitts neuron. The use of multiple thresholds allows to operate with neuron in two modes: binary-valued and multi-valued, respectively [18]. The first mode uses the following activation:

$$f_{\mathbf{t}}^{(2)}(x) = \begin{cases} 1, & \text{if } t_{2j-1} \leq x < t_{2j}, j \in \{1, \dots, [k/2]\}, \\ 0, & \text{if } t_{2j} \leq x < t_{2j+1}, j \in \{0, 1, \dots, [k/2]\}. \end{cases} \quad (1)$$

Note that bipolar output values are also acceptable for such function, if 0 is replaced by  $-1$  in the last row in braces in equation (1).

The second mode can be preferable for application in multiclass classification [19, 20]. In this mode the following  $(k+1)$ -valued counterpart of (1) is used

$$f_{\mathbf{t}}^{(k+1)}(x) = \begin{cases} 0, & \text{if } x < t_1, \\ 1, & \text{if } t_1 \leq x < t_2, \\ \dots\dots\dots & \\ k-1, & \text{if } t_{k-1} \leq x < t_k, \\ k, & \text{if } t_k \leq x. \end{cases} \quad (2)$$

Notice that both activations (1) and (2) can be used in multiclass classification [18]. But in the case of the binary-valued activation (1) the output layer of special shape is required as well as the employing of the “one versus all” approach for the learning of classifier [21].

The *main goal* of the paper is the study of the model of 2-layer neural network whose hidden layer consists of binary-valued multithreshold neurons and examination whether this model is suitable for the multiclass classification, as well as the development of the synthesis algorithm for such networks.

## 2. Related works

As it was mentioned in introduction, multithreshold neural units were proposed in the early studies in artificial intelligence [17, 21, 22]. The first models employed the binary-valued activation in the form (1). The reason for the use of additional thresholds was the assumption that it can improve the capability of basic (single-threshold) linear threshold units [23, 24]. The theoretical reasons were studied in [17, 21, 23, 25] confirming this assumption and giving quantitative expressions for it. These works dealt with the estimation of capability of multithreshold elements to produce dichotomies of finite sets in  $n$ -dimensional real space [9]. These studies confirmed the

power of multithreshold approach and its advantage over single-threshold one. But the ways of the practical realization of this advantage remained almost unclear, because few applications of multithreshold-based models and networks were proposed [26] despite their better potential capacities compared to the basic linear threshold units [18, 24]. The hardness results stated in [5, 18] for 2-valued multithreshold neurons and networks shed some light on these difficulties related to the design of multithreshold systems in the case of two or more thresholds. Nevertheless, the lack of learning techniques for multithreshold systems implies the reduction of the interest in their development [6, 7, 27].

Papers [5, 28] stated some recent advances in the study of bithreshold neural units and NN that changed the situation somewhat. It was caused by new approaches in the synthesis of NN whose hidden layer consists of bithreshold neurons, which were proposed in [5, 24] using the generalization of the Baum's idea [29] for threshold networks combined (see [24]) with the power of the bithreshold activation function, which is the partial case of (1) obtained for  $k = 2$ .

The above observation was confirmed in [1, 3], where bithreshold neural networks were considered as effective tools for intellectual data processing and machine learning [4]. The drawbacks and limitations of the basic bithreshold NN were stated in [5]. They were explained as the consequence of the non-local nature of bithreshold activation function in [24]. Hybrid models of the multiclass classifier with additional hidden layers were proposed in [29] in order to reduce the mentioned downside of the bithreshold paradigm. It should be noted that bithreshold NN can be useful not only for solution of classification problems. The scope of their application can be considerably extended [8, 14] to new scopes. E.g., they were mentioned in design of powerful deep NN providing the exponential improvement of the memorization capacity [15]. Moreover, the bithreshold approach is not limited only by real-valued domain [16]. The generalization of multithreshold paradigm is known for systems proceeding with complex-valued data [16]. The complex bithreshold-like activations were introduced in [12] in order to increase the capability of complex-valued neural units.

Notice that the mentioned advances in the use of multithreshold activation are actually related to only networks that use bithreshold nodes in their hidden layers [6]. There are applications of networks including nodes neither with multithreshold activation (1) nor (2) in the case where the number of thresholds  $k > 2$  [5, 18]. Papers [18, 28] clarify the reasons causing difficulties related to the application of multithreshold approach. The learning of a single multithreshold unit with activation (1), as well as the learning of NN consisting of neurons with activation (2) proved to be NP-hard even in the case  $k = 2$ . Therefore, the intractability is the internal property of multithreshold models [30].

### 3. Models and methods

#### 3.1. Model of multithreshold neural unit

Let us consider again a model of  $k$ -threshold *binary-valued* neuron with the *weight vector*  $\mathbf{w}$  and (ordered) *threshold vector*  $\mathbf{t}$ , which output is given by (1). It is possible to enhance this model a bit by using a binary starting label. The performance of such unit can be described as follows:

$$y = \begin{cases} \lambda, & \text{if } t_{2j-1} \leq \mathbf{w} \cdot \mathbf{x} < t_{2j}, j \in \{1, \dots, \lfloor k/2 \rfloor\}, \\ \bar{\lambda}, & \text{if } t_{2j} \leq \mathbf{w} \cdot \mathbf{x} < t_{2j+1}, j \in \{0, 1, \dots, \lfloor k/2 \rfloor\}, \end{cases} \quad (3)$$

where  $\mathbf{x} = (x_1, \dots, x_n)$  is an input vector,  $\mathbf{w} \cdot \mathbf{x} = w_1 x_1 + \dots + w_n x_n$ ,  $\lambda \in \{0, 1\}$  is a binary label of the neuron,  $\bar{\lambda}$  is its negation, and  $y$  is the output value of the neuron. Further, we shall consider only binary-valued neurons that obey (3) and their attribute “binary-valued” will be omitted. The corresponding labeled activation function will be denoted as  $f_{\mathbf{t}, \lambda}^{(2)}$ .

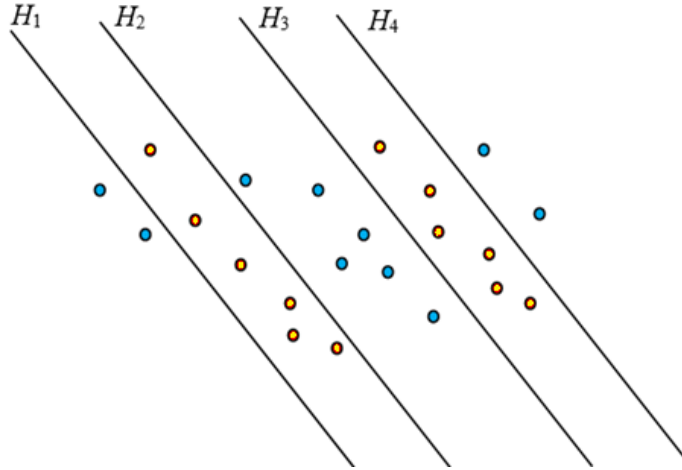
The labeled  $k$ -threshold neuron is completely defined by the triplet  $(\mathbf{w}, \mathbf{t}, \lambda)$ . As well as in [5] we call this triplet the *structure vector* of labeled multithreshold neuron.

Two sets  $A^- \subset \mathbf{R}^n$  and  $A^+ \subset \mathbf{R}^n$  are called  $k$ -separable [18] if there exists a  $k$ -threshold neuron with structure  $(\mathbf{w}, \mathbf{t}, \lambda)$  such that

$$\text{for all } \mathbf{x} \in A^+ \quad f_{\mathbf{t}, \lambda}^{(2)}(\mathbf{w} \cdot \mathbf{x}) = 1 \quad \text{and for every } \mathbf{x} \in A^- \quad f_{\mathbf{t}, \lambda}^{(2)}(\mathbf{w} \cdot \mathbf{x}) = 0.$$

In this case labeled  $k$ -threshold neuron with the structure  $(\mathbf{w}, \mathbf{t}, \lambda)$  produces the  $k$ -dichotomy  $(A^-, A^+)$  of the set  $A = A^- \cup A^+$  in two disjoint sets  $A^-$  and  $A^+$ . Therefore,  $k$ -threshold neurons can be used as binary classifiers. But, as stated in [18], the computational power provided by a single multithreshold unit is insufficient for the design of efficient multiclass classifiers [2].

Model (3) has a simple geometrical interpretation [18]. The family of parallel hyperplanes  $H_j : \mathbf{w} \cdot \mathbf{x} = t_j$ ,  $j \in \{1, \dots, k\}$  divides the space  $\mathbf{R}^n$  by  $k+1$  parts, which can be successively labeled by numbers  $0, 1, \dots, k$ . In the case  $\lambda = 1$  all points belonging to parts with even indices are attributed as “negative”, whereas “odd” parts are considered as “positive” [18]. If  $\lambda = 0$ , then the order of parts should be changed to opposite. The illustration is shown in Figure 1, where the case  $n = 2, k = 4, \lambda = 1$  is considered, and the sets of blue (negative) and yellow (positive) points are 4-separable.

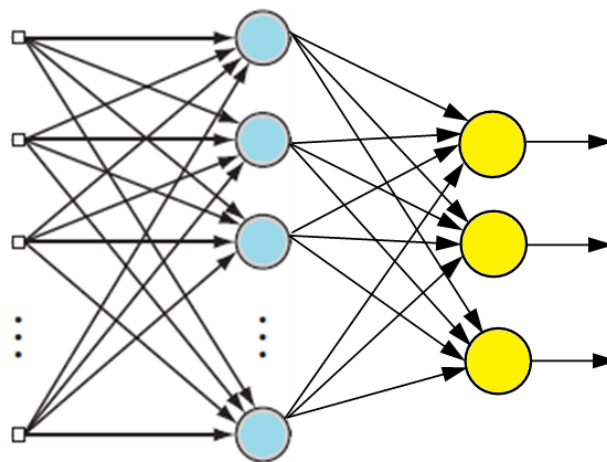


**Figure 1:** Illustration of the performance of binary-valued 4-threshold neuron.

Figure 1 can also illustrate the nature of difficulties related to the application of a binary-valued  $k$ -threshold neuron. Its value can alternate  $k$  times. It ensures the greater capability of the neuron on the one hand, but results in the hardness of its training on the other hand.

### 3.2. Model of 2-layer multithreshold neural network

We can overcome the limitation of a single multithreshold neuron by using a NN in which multithreshold neurons are connected layer-wise. The simplest architecture has only two computation layers and its hidden layer consists of labeled multithreshold neurons, whereas outputs layer contains single-threshold nodes each of them corresponds to a separate class of patterns. The architecture of a 2-layer feedforward fully-connected *multithreshold neural network* is shown in Figure 2.



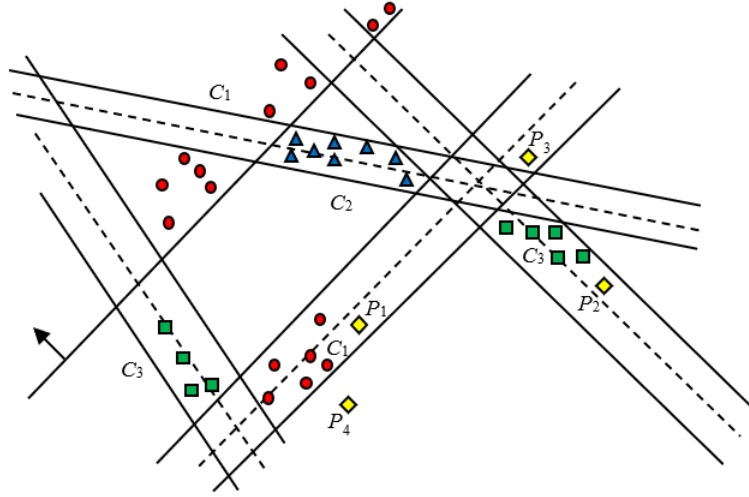
**Figure 2:** Architecture of a 2-layer multithreshold NN with  $n$ - $h$ - $l$  topology

The NN has  $n$  nodes in the input layer,  $h$  multithreshold neurons in the hidden layer and  $l$  threshold neurons in the output layer, where  $l$  is a predefined number of classes. Note that the hidden layer can be a bit heterogeneous in the sense of the number of thresholds and labels of neurons. Namely, the same number of thresholds is not required for neurons in this layer. This assumption can make network synthesis easier and may reduce the size of memory that is necessary to store all network parameters.

Consider how multithreshold NN can be used to perform the desired partition of patterns in  $n$ -dimensional space. The example of the network performance in two dimensions is presented in Figure 3.

Circles are members of the first class  $C_1$ . The second class  $C_2$  is formed by triangles, and squares belong to the third class  $C_3$ . Note that both  $C_1$  and  $C_3$  are formed by 2 compact clusters, which are denoted using multiple occurrences of class names. The presented partition can be produced by 2-layer multithreshold NN, which have 2-4-3 topology. The hidden layer of the network consists of 4 multithreshold neurons each of them is responsible of the recognition of patterns belonging to one of 3 classes. For the proper classification of patterns from class  $C_1$  a single neuron is sufficient. The same is true for  $C_2$ . The first class  $C_1$  requires the use of multithreshold neuron with 3 thresholds, whereas a single bithreshold neuron suffices for the second

class. The proper classification of patterns belonging to the third class  $C_3$  requires the use of 2 bithreshold neurons (one of them can be even replaced by a single-threshold neuron). The hyper-planes that form bounding surfaces of multithreshold neurons are drawn in solid lines.



**Figure 3.** Example of the performance of 2-layer multithreshold neural network

Each hidden neuron makes its own primary “slice” whose width depends on neuron two thresholds, which in turn are defined by the relative position of patterns. Every multithreshold neuron recognizes at least two patterns of corresponding class, with possibly some additional points. All primary slices are equidistant from their pivot lines depicted in Figure 3 in dashed line. For the neuron corresponding to the first class one more (unbounded) slice is required, which is performed using an additional third threshold. The corresponding line is marked in Figure 3 using an arrow. The label 0 should be assigned to this neuron, whereas all other neurons in the hidden layer would be labeled with 1. Every single-threshold neuron in the output level can be connected only with neurons corresponding to its own class and performs only the disjunction of its inputs.

Consider the network performance in the prediction mode. Let  $P_1$  and  $P_2$  are new instances presented to the network. Using (3) and the rule of the action of the output neurons we can conclude that network classifies  $P_1$  and  $P_2$  as members of  $C_1$  and  $C_3$ , respectively. It should be noted that classification of patterns  $P_3$  and  $P_4$  seems less clear. We shall return to this issue later.

### 3.3. Network synthesis algorithm

Consider the task of the synthesis of multithreshold NN. Essentially, multithreshold NN with  $l$  outputs computes a vector mapping  $\mathbf{F}: \mathbf{R}^n \rightarrow \{0,1\}^l$ . The question arises how to design of a computationally efficient algorithm for synthesis a network, which is able to produce a given partition  $(C_1, \dots, C_l)$  of the finite set  $C$  into  $l$  disjoint nonempty sets  $C_1, \dots, C_l$ , i.e., for all  $\mathbf{x} \in C$   $\mathbf{F}_i(\mathbf{x}) = 1 \Leftrightarrow \mathbf{x} \in C_i$ , where  $i = 1, \dots, l$ .

In the algorithm design the approach to NN synthesis shall be used that is the generalization of similar techniques used in [5] for the synthesis of bithreshold NN. This approach extends an idea of the synthesis of single-threshold NN [29] to the case of neurons with several thresholds.

The necessary number of such neurons would be inserted in the hidden layer that each of them is capable to recognize its own portion of patterns belonging to a particular class by separating them from patterns belonging to other classes using 2 or more parallel hyperplanes.

Let a collection of pairs  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$  be our training set, where  $m$  ( $m \geq n$ ) is its size,  $\mathbf{x}_i \in \mathbf{R}^n$  is a training pattern provided with its label  $y_i \in \{1, \dots, l\}$  and  $y_i = j$  implies that the corresponding pattern  $\mathbf{x}_i$  belongs to the class  $C_j$ ,  $i = 1, \dots, m$ . Suppose that all patterns in the sequence  $X = \mathbf{x}_1, \dots, \mathbf{x}_m$  are in general position in  $n$ -dimensional space—that is, every subset of  $n$  or fewer patterns consists of linearly independent elements. Assume also that  $X_i = X \cap C_i \neq \emptyset$ ,  $i = 1, \dots, m$ ,  $\mathbf{y} = (y_1, \dots, y_m)$ . Consider the following pseudocode of the network synthesis algorithm:

MultithresholdSynthesis( $X, \mathbf{y}, \alpha, p$ )

Stage 1. Synthesis of the hidden layer

- 1.1  $Z \leftarrow \{X_1, \dots, X_l\}$
- 1.2  $h \leftarrow 0$
- 1.3 while  $Z \neq \emptyset$ :
  - 1.4 select a random set  $X_i$  from  $Z$
  - 1.5  $h \leftarrow h + 1$
  - 1.6  $c[h] \leftarrow i$
  - 1.7  $r \leftarrow \min\{n, |X_i|\}$
  - 1.8 Move  $r$  randomly chosen patterns from  $X_i$  into the matrix  $A$
  - 1.9 Solve linear system  $\mathbf{w} \cdot A^T = \mathbf{1}$
  - 1.10  $\varepsilon \leftarrow \alpha \min\{|\mathbf{w} \cdot \mathbf{x} - 1| \mid \mathbf{x} \in X \setminus X_i\}$
  - 1.11  $X_i \leftarrow X_i \setminus \{\mathbf{x} \in X_i \mid |\mathbf{w} \cdot \mathbf{x} - 1| < \varepsilon\}$
  - 1.12  $\mathbf{t} \leftarrow (1 - \varepsilon, 1 + \varepsilon)$
  - 1.13  $\lambda \leftarrow 1$
  - 1.14 foreach  $\mathbf{a} \in X_i$ :
    - 1.15  $s \leftarrow \mathbf{w} \cdot \mathbf{a}$
    - 1.16  $\varepsilon_1 \leftarrow \alpha \min\{s - \mathbf{w} \cdot \mathbf{x} \mid \mathbf{x} \in X \setminus X_i, \mathbf{w} \cdot \mathbf{x} < s\}$
    - 1.17  $\varepsilon_2 \leftarrow \alpha \min\{\mathbf{w} \cdot \mathbf{x} - s \mid \mathbf{x} \in X \setminus X_i, \mathbf{w} \cdot \mathbf{x} \geq s\}$
    - 1.18  $B \leftarrow \{\mathbf{x} \mid \mathbf{x} \in X_i, \varepsilon_1 < \mathbf{w} \cdot \mathbf{x} - s < \varepsilon_2\}$
    - 1.19 if  $|B| \geq p$ :
      - 1.20  $X_i \leftarrow X_i \setminus B$
      - 1.21 if  $\varepsilon_1$  is well-defined: insert  $s - \varepsilon_1$  in the ordered vector  $\mathbf{t}$
      - 1.22 else:  $\lambda \leftarrow 0$
      - 1.23 if  $\varepsilon_2$  is well-defined: insert  $s + \varepsilon_2$  in the ordered vector  $\mathbf{t}$

1.24 Add multithreshold neuron  $(\mathbf{w}, \mathbf{t}, \lambda)$  in the hidden layer

1.25 if  $|X_i| = \emptyset$ :

1.26 remove  $X_i$  from  $S$

Stage 2. Synthesis of the output layer

2.1 for  $i \leftarrow 1$  to  $l$ :

2.2 for  $j \leftarrow 1$  to  $h$ :

2.3 if  $c[h] = i$ :  $v_{ij} \leftarrow 2$

2.4 else:  $v_{ij} \leftarrow 0$

2.5 Add threshold neuron  $(\mathbf{v}_i, 1)$  in the output layer

The presented algorithm has 2 parameters  $X$ —the sequence of training patterns,  $\mathbf{y}$ —the target vector of class labels for all patterns (is used only in step 1.1), and two hyperparameters  $\alpha$  and  $p$ . The synthesis process is divided by 2 stages. The main first stage defines the creation of the network hidden layer consisting of multithreshold neurons with variable number of thresholds. The second stage is very simple and defines the output layer consisting of single-threshold neurons.

In the above algorithm  $h$  is the counter responsible for the current number of hidden multithreshold neurons. After the completion of the first stage  $h$  equals the size of the hidden layer. Inside a while loop the portion of unhandled patterns is selected, which belong to the same class. Then  $r$  patterns are moved into rows of auxiliary matrix  $A$ . In step 1.9 the vector of length  $r$  is denoted by  $\mathbf{1}$ , which consists solely of 1's. Steps 1.8–1.12 are intended to determine the parameter of the “main slice” of multithreshold neuron which is defined by the weight vector  $\mathbf{w}$  from step 1.9—a normal vector of the two parallel hyperplanes that perform this slice using thresholds  $1 - \varepsilon$  and  $1 + \varepsilon$ , respectively. Such slices are crucial for MultithresholdSynthesis, because each of them properly classifies at least  $n$  patterns of  $i$ th class (when patterns are in general position). The nested loop in steps 1.14–1.23 is intended to try candidates for secondary slices. Each such slice is taken into account only if it detects at least  $p$  new properly classified patterns, where  $p$  is the hyperparameter of the synthesis algorithm.

Note also that  $\varepsilon_1$  and  $\varepsilon_2$  determine the width of corresponding secondary slice and can be undefined. In this case at most a single threshold will be added into the threshold vector. It can result in multithreshold neurons with odd number of thresholds. If  $\varepsilon_1$  is undefined, then 0 is assigned to the neuron label. It is useful in the case when all remaining class patterns are only training instances lying in the “negative” half-space produced by corresponding hyperplane  $\mathbf{w} \cdot \mathbf{x} = s$ .

As mentioned in [5], the hyperparameter  $\alpha$  can be considered as a tolerance measure of multithreshold NN. If  $\alpha \in (0, 1]$ , then the network performs well on training set (training error is almost 0) but it can be “overfitted” and less effective outside the training data. If  $\alpha$  is slightly greater than 1, then the network can have the better generalization ability. Moreover, the use of  $\alpha > 1$  can result in the reduction of the number of neurons in the hidden layer.

In step 2.5 we denote by  $(\mathbf{v}_i, 1)$  the  $i$ th single-threshold neuron with the weight vector  $\mathbf{v}_i = (v_{i1}, \dots, v_{ih})$  and the unit threshold.



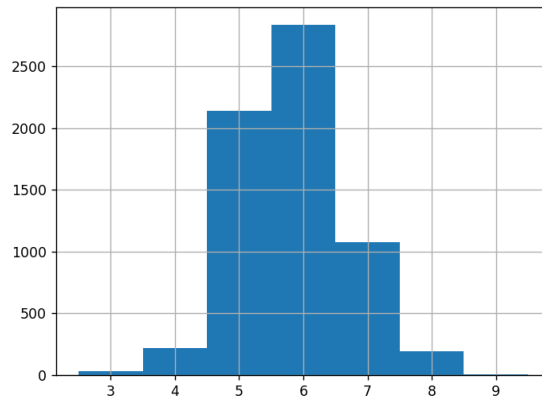
The following proposition estimates the complexity of the considered algorithm.

**Proposition.** An arbitrary partition of  $m$ -set  $X$  into  $l$  disjoint nonempty sets  $X_1, \dots, X_l$  of  $n$ -dimensional vectors can be performed by a 2-layer multithreshold NN with at most  $\lceil m/n \rceil + l$  nodes in the hidden layer and  $l$  neurons in the output layer, whereas the network can be synthesized using  $\text{MultithresholdSynthesis}(X, \mathbf{y}, \alpha, p)$  in  $O(m^3)$  time.

## 4. Experiment

Let us consider the capability of  $\text{MultithresholdSynthesis}$  to synthesis a multithreshold multiclass classifier to solve the classification problems compared to classical classifiers, such as  $k$ -nearest neighbor, decision tree, random forest, RBF SVC (support vector classifier with RBF kernel),  $\text{MLPClassifier}$  (multilayer perceptron or fully-connected feed-forward NN, using softmax function for estimates the probability of the target class over all possible target classes and cross-entropy loss function). Classifiers were tested on the “Wine Quality” dataset [31], related to red and white variants of the Portuguese “Vinho Verde” wine provided by “UC Irvine Machine Learning Repository” [32], as well as other popular repositories, e.g., [33]. This medium-sized dataset contains 6497 instances, 4898 of which are related to white wine, and other 1599—to red one [31].

The data includes only 12 features, 11 physicochemical properties as inputs (see [34]), wine color and one sensory quality score as the output from 0 to 10 (actually, for red wine the quality is between 3 and 8, and for white wine—between 3 and 9). The distribution of qualities is shown in Figure 4.



**Figure 4.** Distribution of qualities of wines

Note that only 9 instances have the quality value 9 and they are almost invisible in Figure 4. The most frequent quality value is 6 with 2836 representatives. Thus, the classes are very imbalanced. Moreover, not all input features are supposed to be quite relevant [31]. Thus, the training task for this dataset can be considered hard enough to estimate classifier ability [34].

Dataset contains only numerical data (except the color value). Thus, the feature extraction was very simple. Data were standardized by removing the mean and scaling to unit variance using standard scaler from Scikit Learn [2, 35]. 80% of instances of every dataset were used as the

training set (namely, 5198 instances), and the rest 20%—as the test set for the final evaluation of classifier results. In order to avoid the dependence of the selection of patterns for the training, random sub-sampling validation was used [2], i.e., randomly chosen testing instances keeping the same proportion 1:3 between instances concerning red and white wines, respectively.

For every classifier stratified 5-fold cross-validation was used in order to evaluate the classifier performance and to tune its hyperparameters. The average performance measure over all 5 splits and grid search was employed for this purpose.

For  $k$ -nearest neighbor classifier only the number of neighbors was explored. For multilayer perceptron Adam optimizer was used and the number of hidden layer as well as their sizes, the kind of activation function and batch size were investigated. For support vector classifier with RBF kernel different combinations of values were tried of regularization parameter  $C$ , degree of the polynomial kernel function and kernel coefficient  $\gamma$  [35]. In the case of random forest classification, the number of trees in the forest and the maximum depth of the tree were chosen. The last parameter also was studied for decision tree classifier.

Both hyperparameters  $\alpha$  and  $p$  were explored for MultithresholdSynthesis( $X$ ,  $\mathbf{y}$ ,  $\alpha$ ,  $p$ ).  $\{0.7, 0.8, 0.85, 0.9, 0.95, 1, 1.05, 1.1, 1.15, 1.2\}$  and  $\{2, 3, 4, 5, 6, 7\}$  ranges were used for  $\alpha$  and  $p$ , respectively. During the experiments we used implementations of 5 above-mentioned popular classifiers provided by Scikit-Learn library [35].

Two classification metrics were used during simulation. The first was the accuracy, the second—top-2 score (successful when the target class is within the top 2 predictions) [35].

## 5. Results

The following two tables contain results of experiments in the case of first and second metrics, respectively. For first 5 classifiers best between all specified combinations of hyperparameter values are not shown for sake of brevity (and because their study is beyond the main scope of the paper). Note that all classifiers were retrained one more time on full training set after the finish of cross-validation and grid search. That is why the values of metrics on the test set are often higher than average across all splits during cross-validation.

**Table 1**

Classifiers performance on “Wine Quality” dataset. Accuracy metric (in %)

| Classifier                 | Cross-validation |            | Test set |
|----------------------------|------------------|------------|----------|
|                            | Train score      | Test score |          |
| 7-Nearest Neighbor         | 65.45            | 54.43      | 56.01    |
| Decision tree              | 100              | 57.68      | 60.92    |
| Random forest              | 100              | 65.62      | 67.87    |
| Multilayer perceptron      | 91.71            | 63.11      | 62.13    |
| Support vector machine     | 67.55            | 56.77      | 58.15    |
| 2-layer multithreshold NN1 | 100              | 49.02      | 54.28    |
| 2-layer multithreshold NN2 | 93.02            | 56.24      | 60.95    |

Results for two versions of multithreshold 2-layer NN are presented in last two rows in both tables. In the synthesis of NN1  $\alpha = 1$  was used. NN2 was synthesized using the best value  $\alpha^* = 1.05$  found by the grid search. In both networks the best value  $p^* = 4$  was used.

**Table 2**

Classifiers performance on “Wine Quality” dataset. Top-2 score metric (in %)

| Classifier                 | Cross-validation |            | Test set |
|----------------------------|------------------|------------|----------|
|                            | Train score      | Test score |          |
| 7-Nearest Neighbor         | 92.2             | 84.31      | 85.01    |
| Decision tree              | 100              | 61.17      | 63.88    |
| Random forest              | 100              | 91.38      | 92.05    |
| Multilayer perceptron      | 98.46            | 85.6       | 87.97    |
| Support vector machine     | 88.84            | 77.14      | 77.8     |
| 2-layer multithreshold NN1 | 100              | 57.11      | 56.29    |
| 2-layer multithreshold NN2 | 98.02            | 83.92      | 84.36    |

By analyzing the performance results shown in Table 1 and Table 2, we can conclude that:

- The basic version of multithreshold network NN1 was good memorizer and performed perfectly on the training set.
- The generalization ability of NN1 was very poor. Its performance on the test set was worst for both considered metrics.
- The optimized version NN2 lost a bit in the accuracy on the training set, but performed considerable better on new patterns.
- The proper choice of the  $\alpha$  allow us to avoid the great overfitting.
- The test error during cross-validation varies significantly.
- The use of value of  $\alpha$  lying inside the half-interval  $(0, 1]$  results in the degradation of the generalization capability of the classifier.
- The most typical number of thresholds of the neuron in the hidden layer of NN2 was 3.
- The growth of the value of hyperparameter  $p$  results in almost bithreshold hidden layer.
- Using the modified software implementation and additional memory it is possible to decrease the time complexity of  $\text{MultithresholdSynthesis}(X, \mathbf{y}, \alpha, p)$  in Proposition from  $O(m^3)$  to  $O(m^2(n + \log m))$ .
- The estimated hidden layer size  $\lceil m/n \rceil + l$  from Proposition is too pessimistic. The actual size in the experiment was 231. But it seems plausible that the number of nodes in the hidden layer grows as  $m/n$ .

## 6. Discussions

The proposed model of 2-layer multithreshold NN and corresponding synthesis algorithm can be considered as the first step towards an application of multilayer NN based on the multithreshold paradigm in machine learning in the case of an arbitrary number of thresholds. Even the first applications of such networks in classification on the real-word datasets showed that they have the similar pros and cons that bithreshold neural networks considered in [5, 24].

The principal consequences that follow from concrete results of simulation concerning the pattern classification on synthetic [35] as well as real-world datasets [32, 33] using multithreshold 2-layer NN suggest that networks yielded by  $\text{MultithresholdSynthesis}$  or its more complicated

modifications have good performance on the design set but are rather inefficient on new instances. More precisely, as it was observed that for “Wine Quality” and other datasets:

Making prediction, classifier can attribute new patterns to a class whose all representatives in the training set are very distant from this pattern (“nonlocality” of classifier [24]).

The size of the hidden layer of the network strongly depends on the order in which the training patterns were presented to learner. Moreover, during the cross-validation the NN synthesized on different splits demonstrated the significant diversity of the size of the hidden layer.

The size of the hidden layer is too large in the case of the larger datasets.

This downsides of the basic multithreshold 2-layer NN classifier is illustrated in Figure 3. The network classifies pattern  $P_3$  as a member of first class despite it looks more likely be a member of one of two other classes (first drawback). Moreover, the network is undecided when it predicts a class for new pattern  $P_4$ , despite it is situated in the proximity of a compact cluster of members of the first class (second drawback).

A conjecture was stated in [5] that the reason for such two downsides is the nature of the multithreshold activation function. These downsides can be eliminated by using a localized modifications of activation function (1), e.g., the multithreshold modifications of smoothed local activations, which were proposed in [24]. Another solution consists in the use of modified network architectures with additional layers comprising nodes of different kinds, as it was proposed in [30] for bithreshold neural networks. The last drawback can be partially reduced using some additional hyperparameters in the MultithresholdSynthesis, e.g., the limitation of the hidden layer size.

It should be noted that the more expensive part of MultithresholdSynthesis algorithm is computations in step 1.10, which have the  $O(m^2n)$  contribution in its time complexity. These computations can be performed simultaneously with solely synchronization of the shared variable  $\varepsilon$ . Therefore, MultithresholdSynthesis has massively parallel structure and can be effectively realized using multicores processors and graphical accelerators.

## 7. Conclusions

The ways of the use of multithreshold paradigm in neural computation were treated in the paper. The new modification of the model of binary-valued multithreshold neural unit was proposed, which allows often to reduce number of thresholds. The 2-layer neural network architecture was studied whose hidden layer consists of multithreshold neurons with different number of thresholds. The synthesis algorithm was developed for such multithreshold NN. The multiclass classifier was designed on the base of such architecture.

Experiment results of the performance of multithreshold classifier on “Wine Quality” real-world dataset were presented. These results clarify pros and cons of the proposed model compared to some popular classifiers. The impact of algorithm hyperparameters on the classifier ability to make a precise prediction, as well as and the ways to improve it were discussed. The main conclusion is that there exists a need in the development of localized modifications of the multithreshold activation function and additional hyperparameters in synthesis algorithm in order to regulate the network performance more precisely.

## 8. References

- [1] E.H. Houssein, M.E. Hosney, M.M. Emam, E.M. Younis, A.A. Ali, W.M. Mohamed, Soft computing techniques for biomedical data analysis: open issues and challenges, *Artificial Intelligence Review* 56 (2023): 2599–2649.
- [2] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media, Sebastopol, CA, 2022.
- [3] 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.
- [4] I. Izonin, R. Tkachenko, S. A. Mitoulis, A. Faramarzi, I. Tsmots, D. Mashtalir, Machine learning for predicting energy efficiency of buildings: a small data approach, in: *Procedia Computer Science*, volume 231, 2024, pp. 72–77. doi:10.1016/j.procs.2023.12.173.
- [5] V. Kotsovsky, A. Batyuk, Representational capabilities and learning of bithreshold neural networks, in: S. Babichev et al. (Eds), *Advances in Intelligent Systems and Computing*, volume 1246, Springer, Cham, 2021, pp. 499–514.
- [6] R. Tkachenko, An integral software solution of the SGTm neural-like structures implementation for solving different Data Mining tasks, in: S. Babichev, V. Lytvynenko (Eds.), *Lecture Notes on Data Engineering and Communications Technologies*, volume 77, Springer, Cham, 2022, pp. 696–713.
- [7] M. Havryliuk, N. Hovdysh, Y. Tolstyak, V. Chopyak, N. Kustra, Investigation of PNN optimization methods to improve classification performance in transplantation medicine, in: *CEUR Workshop Proceedings*, volume 3609, 2023, pp. 338–345.
- [8] 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.
- [9] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed, Pearson Education, Upper Saddle River, NJ, 2009.
- [10] O. Kuchanskyi et al., Gender-related differences in the citation impact of scientific publications and improving the authors' productivity, *Publications* 11.3 (2023): 37.
- [11] M. Anthony, J. Ratsaby, Large-width machine learning algorithm, *Progress in Artificial Intelligence* 9.3 (2020): 275–285.
- [12] V. Kotsovsky, F. Geche, A. Batyuk, Artificial complex neurons with half-plane-like and angle-like activation function, in: *Proceedings of the International Conference on Computer Sciences and Information Technologies, CSIT 2015, Lviv, Ukraine, 2015*, pp. 57–59.
- [13] B. Amirgaliyev, O. Kuchanskyi, Y. Andrashko, Building a dynamic model of profit maximization for a carsharing system accounting for the region's geographical and economic features, *Eastern-European Journal of Enterprise Technologies*, 2.4-116 (2022): 22–29.
- [14] M. Lupei, M. Shlahta, O. Mitsa, Y. Horoshko, H. Tsybko, V. Gorbachuk, Development of an interactive map within the implementation of actual state and public directions, in: *Proceedings of the 12th International Conference on Advanced Computer Information Technologies, ACIT 2022, Ruzomberok, Slovakia, 2022*, pp. 384–387.
- [15] S. Rajput, K. Sreenivasan, D. Papailiopoulos, A. Karbasi, An exponential improvement on the memorization capacity of deep threshold networks, in: *Advances in Neural Information Processing Systems*, volume 16, 2021, pp. 12674–12685.

- [16] Z.-G. Zhang, Y.-L. Xiao, J. Zhong, Unitary learning in conditional models for deep optics neural networks, in: Proceedings of SPIE – The International Society for Optical Engineering, volume 12565, 2023, no. 1256543.
- [17] R. Takiyama, Multiple threshold perceptron, *Pattern Recognition* 10.1 (1978): 27–30.
- [18] 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, doi: 10.1109/CSIT61576.2023.10324129.
- [19] N. Jiang, Z. Zhang, X. Ma, J. Wang, Y. Yang, Analysis of nonseparable property of multi-valued multi-threshold neuron, in: Proceedings of 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 2008, pp. 413-419, doi: 10.1109/IJCNN.2008.4633825.
- [20] I. Prokić, Characterization of multiple-valued threshold functions in the Vilenkin-Chrestenson basis, *Journal of Multiple-Valued Logic and Soft Computing* 34.3-4 (2020): 223–238.
- [21] R. Takiyama, The separating capacity of a multithreshold threshold element, *IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-7.1* (1985): 112–116.
- [22] K. Ashenayi, J. Vogh, M.R. Sayeh, B. Karimi, T. Baradaran, Multiple threshold perceptron using sinusoidal function, *International Journal of Modelling and Simulation* 12.1 (1992): 22–26.
- [23] S. Olafsson, Y. S. Abu-Mostafa, The capacity of multilevel threshold function, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10.2 (1988): 277–281.
- [24] V. Kotsovsky, A. Batyuk, Feed-forward neural network classifiers with bithreshold-like activations, in: Proceedings of IEEE 17th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2022, Lviv, Ukraine, 2022, pp. 9–12.
- [25] 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.
- [26] V.K. Venkatesan, I. Izonin, J. Periyasamy, A. Indirajithu, A. Batyuk, M.T. Ramakrishna, Incorporation of energy efficient computational strategies for clustering and routing in heterogeneous networks of smart city, *Energies* 15.20 (2022): 7524.
- [27] Y. Andrashko et al., A method for assessing the productivity trends of collective scientific subjects based on the modified PageRank algorithm, *Eastern-European Journal of Enterprise Technologies*, 1.4 (121) (2023): 41–47.
- [28] V. Kotsovsky, A. Batyuk, V. Voityshyn, On the size of weights for bithreshold neurons and networks, in: Proceedings of IEEE 16th International Conference on Computer Sciences and Information Technologies, CSIT 2021, Lviv, Ukrain, 2021, volume 1, pp. 13–16.
- [29] E. B. Baum, On the capabilities of multilayer perceptrons, *Journal of Complexity* 4.3 (1988): 193–215.
- [30] V. Kotsovsky, “Hybrid 4-layer bithreshold neural network for multiclass classification,” in *CEUR Workshop Proceedings*, volume 3387, 2023, pp. 212–223.
- [31] P. Cortez, A.L. Cerdeira, F. Almeida, T. Matos, J. Reis., Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems* 47 (2009): 547–553.
- [32] M. Kelly, R. Longjohn, K. Nottingham, The UCI machine learning repository, 2023. URL: <http://archive.ics.uci.edu>.
- [33] OpenML: A worldwide machine learning lab, 2024. URL: <https://openml.org>.
- [34] D. C. Angus, Modeling wine quality from physicochemical properties, (2019). URL: <https://api.semanticscholar.org/CorpusID:209521363>.
- [35] Scikit-learn: Machine Learning in Python, (2024). URL: <https://scikit-learn.org>.