Hybrid 4-Layer Bithreshold Neural Network for Multiclass Classification

Vladyslav Kotsovsky [1]

[1] *State University "Uzhhorod National University", Narodna Square 3, Uzhhorod, 88000, Ukraine*

Abstract

The questions related to the improvement of the performance of bithreshold classifiers are treated in the paper. The new 4-layer neural network architecture is proposed whose first layer performs the normalization of inputs, the second layer is composed of bithreshold neurons, linear threshold units and winner-take-all neurons, and every linear threshold unit in last two layers has only small predefined weights. The synthesis algorithm for such networks is described and estimations of its time complexity and the size of the resulting network are given. The experiment results demonstrate that the implementation of the proposed approaches in the design of multiclass classifiers considerably improves their generalization capabilities.

Keywords

Artificial neural network, bithreshold neuron, synthesis algorithm, learning, multiclass classification

1. Introduction

Neural network technologies are extremely useful in intelligent systems [1] and machine learning [2]. They are embedded in many smart high-tech products [3, 4]. The tremendous capabilities of artificial neural networks (NN) are powered by task-oriented network architectures [5] and special hardware components [3].

The choice of the neuron model for the network is crucial in neural computations [6]. First neural networks used linear threshold units with threshold activation functions [5, 7] and binary outputs. Somewhat later, many kinds of more complicated models of neural units were proposed in order to increase the computational power and the capability of the networks [8] and make their learning easier.

One of the first among them was the model of the neural unit with a binary-valued multi-threshold activation function, which was introduced in the middle 1960s [9]. The simplest kind of such devices was a bithreshold neural unit or a *bithreshold neuron* (BN). Let us recall that the bithreshold neuron with the weight vector $\mathbf{w}=(w_1,\ldots,w_n)\in\mathbf{R}^n$ and thresholds $t_1, t_2 \in \mathbf{R}\ (t_1<t_2)$ is the computation unit with n inputs $x_1,\ldots,x_n$ and the single binary output y, which is calculated by the following rule:

$$y=\begin{cases}1, & \text{if } t_1<\mathbf{w}\cdot\mathbf{x}<t_2,\\ 0, & \text{otherwise.}\end{cases}$$

In the previous equation $\mathbf{x}=(x_1,\ldots,x_n)\in\mathbf{R}^n$ is an input vector. The activation function that is applied to the weighted sum of inputs $\mathbf{w}\cdot\mathbf{x}$ is called a bithreshold activation function. BN is completely defined by its *structure triplet* $(\mathbf{w},t_1,t_2)$. Bithreshold neurons outperform single-threshold ones [10], because they have the more sensible activation capable to trigger only in the case when the sum of weighted stimuli is within the given range, which is specified by thresholds of the neural unit [11].

COLINS-2023: 7th International Conference on Computational Linguistics and Intelligent Systems, April 20–21, 2023, Kharkiv, Ukraine
EMAIL: vladyslav.kotsovsky@uzhnu.edu.ua (V. Kotsovsky)
ORCID: 0000-0002-7045-7933 (V. Kotsovsky)

Bithreshold neurons can be organized in the consecutive layers of feed-forward neural networks, which are capable to solve classification tasks [12, 13]. But simulations revealed several weak points of 2-layer neural network classifier, which hidden layer consists of bithreshold neurons [14]. We will cover it later in the third section, but it is almost evident that before the practical applications, the model of bithreshold neural network classifier must be refined. Our research has *two main goals*:

- The design of the network architecture that would effectively use the advantages of bithreshold activation and mitigate the weakness of the basic model of the network from [14].
- The development of the synthesis algorithm for the classifier based on the designed architecture, as well as theoretical and experimental estimations of its performance.

The paper has the following structure. First, the works related to the topic of the study will be reviewed. Then, the basic model of 2-layer feed-forward bithreshold NN will be considered. We will discuss its advantages and consider carefully downsides related to the practical classification using this model. In the next section a new 4-layer hybrid architecture will be proposed, which enhances the network capability by imposing the locality constraints by using in the first hidden layer linear threshold units and winner-take-all neurons alongside with bithreshold ones. Then, the synthesis algorithm for such networks will be described, which application can significantly improve the classifier generalization capability and preserve the high memorization ability of the network. Next, we will treat the simulation results of the performance of 4-layer hybrid neural network multiclass classifier in the comparison with other popular classifiers provided by Sklearn library. Finally, two last paper sections contain the discussion of obtained results and conclusions.

2. Related Works

Recent research proved that bithreshold neural-like models have good perspectives of the successful application in the solving of important problems of data science [15], machine learning [16, 17] and forecasting [18].

The study of multithreshold neural units has a long history [9, 10]. Networks based on the layer of neurons with bithreshold activations have better capabilities in solving the classification tasks than single threshold ones [10, 13] and are better memorizers [8]. But the practical use of neural systems based on the bithreshold paradigm faces two problems: the estimation of the parameters of the network, which is capable to solve a given task [12] and the design of effective learning techniques for such systems [14].

Some approaches to the solution of the first problem were made in [12] where the magnitude of the weights and thresholds of the bithreshold neuron was established. Namely, it was shown that the largest coefficient of the bithreshold neuron with n binary inputs is bounded by $\sqrt{(n+2)\cdot(n+1)^{(n+1)}}$. The practical consequence follows: $O\left(n^2 \log n\right)$ memory bits are enough to store the structure triplet of BN. Moreover, this estimation is not very loose, because in average at least $\Omega\left(n^2\right)$ bits are required [12].

The second problem is a part of the greatest challenge in the threshold logic, which consists in the design of training algorithms for network with discrete-valued activations. It is stated in [8] that if we consider the network with the fixed topology (i.e., a predefined number of layers and number of the neurons in each layer), then there are no effective approaches to learn such systems even for networks with the single-threshold activation functions (i.e., simple 2-layer networks consisting of classical linear threshold units [19]). It is shown in [12] that similar conclusion is also true for any bithreshold network, no matters how many layers it has, because the corresponding task of the learning such networks is NP-complete. In order to avoid this hardness another way of dealing with the bithreshold neural networks was explored in [14], where the synthesis approach was proposed consisting in the dynamic growth of layers of the network during the synthesis process. This approach ensures that network exactly memorizes every learning pattern. Thus, it satisfies the following paradigm: training neural networks consists in to train the networks until they fit the training dataset perfectly [20].

The synthesis algorithm originates from ideas exposed in [21] and can be extended to the design of multiclass classifiers. Estimations on the network size proved in [14] show that in the case of the memorization of patterns presented by points in general position (i.e., none $n+1$ of them lies on the

same hyperplane) in the case of two and more classes the number of bithreshold neurons in the hidden layer is bounded by $\lceil m/(2n) \rceil$ and $\lceil m/n \rceil$, respectively, where n is the dimension of input patterns, m is the size of the design set. It should be also noted that new deeper architectures were recently proposed in [20, 22, 23], which use tricky approaches to increase the memorization capability of networks in the case of ReLU [22] and threshold [20, 23] activations, respectively.

It has already mentioned in Introduction, that the classifier performance outside the training set was not quite satisfying both for binary and multiclass classification. Properties of bithreshold activation and network working principles will be discussed later that are responsible for the network poor ability to generalization.

3. Methods

3.1. 2-Layer Bithreshold Neural Network Classifier

Let us consider issues related to the bithreshold neural networks and their application in the neural network classifiers. First, let us review in brief the model of 2-layer bithreshold neural network.

3.1.1. Basic 2-Layer Network Architecture

Consider the architecture of 2-layer feed-forward fully-connected neural network, which was proposed in [14] to solve the task of multiclass classification in the case of several disjoint classes. The conceptual graph of the network is shown in Figure 1.

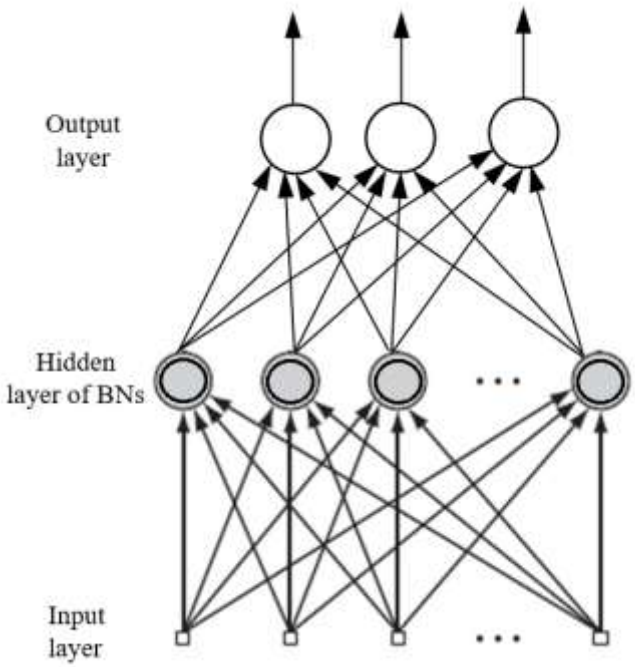

Figure 1: Architecture of 2-Layer Bithreshold neural network

The hidden layer consists of bithreshold neurons and each output node is the single-threshold neuron corresponding to the one of the given classes. Note that concentric circles are used to mark bithreshold neurons and distinguish them from single-threshold ones.

The synthesis (learning) algorithm for such networks was proposed in [14]. It is intended to learn the network classify exactly all patterns from the design (training) set $S = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_m, t_m)\}$, where m is the size of training set, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ is the set of input patterns, which are represented by n-

dimensional vectors and t_i are corresponding class labels, where $t_i \in \{1,2,\ldots,l\}$, $l>1$, $i=1,\ldots,m$. Let $X_i = \{\mathbf{x}_j \mid 1 \le j \le m,\ t_j = i\}$ be the subset of training samples belonging to *i*th class. The main geometrical idea behind the learning algorithm is the separation of X_i from all other patterns by the means of pairs of parallel hyperplanes with respect to the distribution of the labels of patterns, which can be regarded as targets.

3.1.2. Synthesis Algorithm for 2-Layer Bithreshold Networks

The following algorithm is the subtle simplification of the learning algorithm, which was proposed in [14] for the synthesis of 2-layer neural network classifier:

BasicSynthesis(*S*)

1. $k \leftarrow 0$
2. `for` $i \leftarrow 1$ `to` l**:**
3. Add threshold neuron TN[*i*] with bias -1 in the output layer of NN
4. Select X_i from *S*
5. $A_i \leftarrow X_i$
6. `while` $A_i \neq \varnothing$ **:**
7. $k \leftarrow k+1$
8. $r \leftarrow \min\{n, |A_i|\}$
9. Put *r* patterns from A_i into the matrix *A*
10. Solve linear system $\mathbf{w} \cdot A^T = \mathbf{1}$
11. $\varepsilon \leftarrow \min\{\,|\mathbf{w}\cdot\mathbf{x}-1| \mid \mathbf{x} \in X_j,\ j \neq i\}$
12. $A_i \leftarrow A_i \setminus \{\,\mathbf{x} \in A_i \mid |\mathbf{w}\cdot\mathbf{x}-1| < \varepsilon\}$
13. Add BN[*k*] with the structure $(\mathbf{w}, 1-\varepsilon, 1+\varepsilon)$ in the hidden layer of NN
14. Connect BN[*k*] with TN[*i*] with the weight 2
15. `return` NN

The only input parameter of BasicSynthesis is the design set *S*. The above algorithm returns the synthesized network NN. The resulting network is not fully-connected, because each bithreshold neuron from the hidden layer is connected only to the single output node corresponding to the class for the recognition of whose members this bithreshold neuron was created.

Figure 2 shows the example of the classification task in 2 dimensions. The network has classify patterns belonging to 3 given classes, which are marked by green squares, yellow diamonds and blue circles, respectively. The geometrical illustration of how the network works is shown in Figure 2. Every bithreshold neuron from the hidden layer performs its own "slice" whose "width" depends on the position of the points of other classes. BN recognizes at least two patterns of its own class, with possibly some other representatives. Two parallel lines perform a slice and they are equidistant from the dotted line—the "central hyperplane", which is defined by 2 patterns through which this line is drawn (in Figure 2 they are H_1, H_2 and H_3, respectively). For our example three BNs are sufficient to classify exactly all patterns. Thus, the corresponding 2-layer network has 2-3-3 architecture.

As it was mentioned in the introduction, the basic synthesis algorithm from [14] yields the network that memorizes precisely patterns from the design set but its behavior can be poor outside this set. Consider some reasons explaining it in details:

1. Outside the design set the classifier is often indecisive and does not attributes new patterns to any class (e.g., in Figure 2 all points P_1, P_2 and P_4 are not classified although it seems that P_1 would be attributed to the second class and P_2 and P_4—to the third one. This disadvantage results in low values of the recall metric for classes [5].

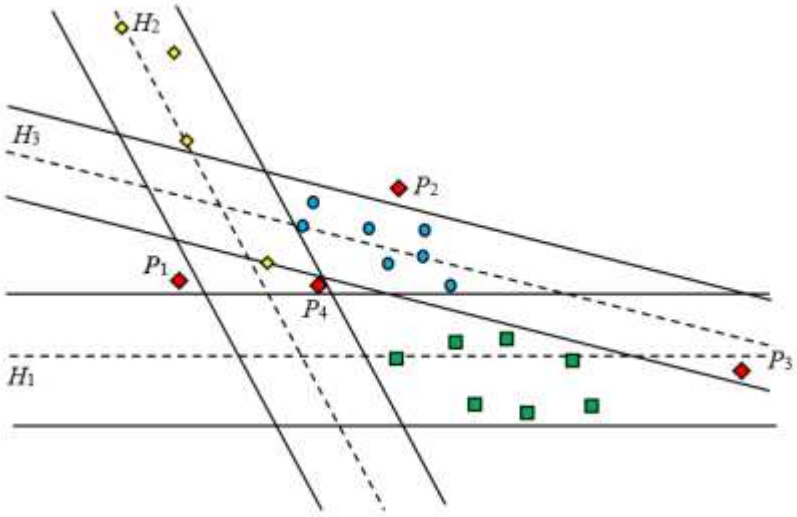

Figure 2: Illustration of the performance of 2-layer bithreshold neural network

2. The network often attributes a new pattern to the class all whose training points are very distant from this pattern (e.g., in Figure 2 pattern P_3 is classified as a member of the third class despite it is too far from the class members). This drawback is related to the nonlocal nature of the bithreshold activation function, which follows from the fact that the decision region of the bithreshold unit is unbounded.

Above drawbacks result in the poor generalization ability of classifiers. If we split the design set into training set and testing set and run synthesis algorithm only on the training set, then we can obtain the classifier with low accuracy on the testing set.

3.2. Hybrid 4-Layer Neural Network

3.2.1. Network Architecture

The disadvantages of the basic classifier and its synthesis algorithm mentioned at the end of the previous section force to look for ways to improve the network performance. Consider some of them.

The main reason of the first drawback is the narrow gap between bounding hyperplanes. It can be reduced if we relax the restriction of the equidistance of the decision hyperplanes from central hyperplane, which is imposed in step 10 of synthesis algorithm. We can divide this step by two smaller steps by finding the upper and lower threshold independently. This can lead to the wider decision regions and decreases the degree of the indecisiveness of the classifiers. E.g., by shifting the left bound line of the second BN in Figure 2, we can force the network to classify the pattern P_1 as a member of the second class. Sometimes the separate search of the biases from the central hyperplane can result in the undefined (or infinite) value(s) of threshold(s). E.g., it is true for our example in Figure 2. In such cases we can restrict the corresponding bias by using the bias of the pattern that is most distant from the central line of its class.

Experiments show that just mentioned modification can improve the accuracy on the testing set by several percent. But it cannot eliminate the lack of the decisiveness of classifiers. In order to reduce the downsides of the basic classifier we can change the topology of the network by replacing the single BN, which is responsible for the recognition of the portion of patterns of the fixed class, by the block consisting of a single BN, two additional neurons in the same layer and one neuron in the next layer.

First, let us explore how we can reduce the first network drawback by using alternative decision rules when bithreshold neurons fail to attribute a new pattern to any class. The simplest solution is based on the slight modification of the nearest-neighbor rule. We can classify the unclassified pattern **x** as a member of the class whose member is nearest to **x**. The entire dataset can be large and to remember all

patterns is too expensive and can decrease the classifier performance. Thus, some derived class characteristics would be used. For example, we can use the center of the class. It is a good choice for compact classes. But it fails for complex-shaped class (e.g., ring-like), because its center may be far away from most class members). The possible tradeoff is the *center* (centroid) of the set of all patterns from the design set that are "properly classified" by the fixed BN in the hidden layer of the network. Another solution consists in the use of the subset of "just classified" patterns that are properly attributed by the current BN for the first time during the synthesis.

Let BN[k] be a kth bithreshold neuron in the hidden layer of the network and Z_k be the set of its "properly classified" (or "just classified") patterns. Then the corresponding centroid can be calculated using the following equation:

$$\mathbf{c}_k = \frac{1}{|Z_k|}\sum_{\mathbf{x}\in X_k}\mathbf{x} \tag{1}$$

We call $\mathbf{c}_k$ a *center* of BN[k]. For example, in Figure 3 $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$ are centroids corresponding to 3 bithreshold neurons in the hidden layer (note also wider gap between hyperplanes of the second class).

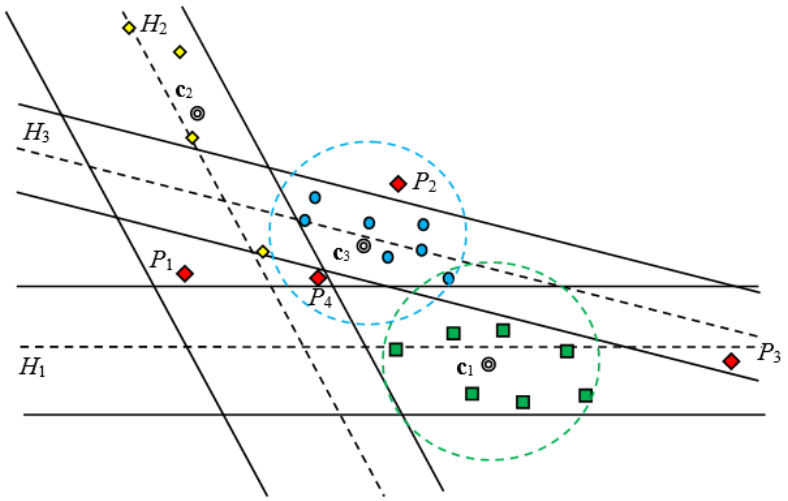

Figure 3: Example of the network performance

Let $k^* = \arg\min_k \|\mathbf{x}-\mathbf{c}_k\|$, where $\mathbf{x}$ is a given input pattern and the index k ranges over all possible indices of centers. Then the network classifies $\mathbf{x}$ as a member of the class t_{k^*}, where t_{k^*} is the target label of the patterns, which were used to calculate the center $\mathbf{c}_{k^*}$. For example, such approach allows us to properly classify new patterns P_2 and P_4 in Figure 3 in the third class.

Let us consider the possible implementation of the nearest-neighbor rule using the neural computations. Since

$$\|\mathbf{x}-\mathbf{c}_k\|^2 = \|\mathbf{x}\|^2 - 2\mathbf{c}_k\cdot\mathbf{x} + \|\mathbf{c}_k\|^2, \tag{2}$$

the minimization of the distance between vectors is equivalent to the minimization of the right part in the last equation. Under the assumption that $\|\mathbf{x}\| = \text{const}$ this task is equivalent to the task of the maximization of the value of expression $2\mathbf{c}_k\cdot\mathbf{x} - \|\mathbf{c}_k\|^2$ over all centroids corresponding to the bithreshold neurons in the second layer of the network. We can treat the last equation as the biased weighted sum of the neuron WTA[k] with the weight vector $2\mathbf{c}_k$ and the bias $-\|\mathbf{c}_k\|^2$. The sublayer of "winner-take-all" (WTA) neurons ensures the desired behavior of the classifier.

Notice that the input normalization is required for the successive application of above approach. But the normalization can result in the undesirable coincidence of nearly collinear patterns. We can avoid

it by augmenting input patterns with a new component (e.g., $x_{n+1}=1$). The need to perform these transformations requires us the add the additional normalization layer at beginning of the network.

Consider the second downside of the basic classifier, which is caused by the unboundedness of the decision region of bithreshold neurons. Note also that the WTA layer has even worse property, because it has not only unbound decision region but is always decisive. The above downside can be reduced by rejecting patterns, which are too distant from patterns that were used during the computation of the coefficients of bithreshold neurons in the hidden layer. Let

$$r_k = \max_{\mathbf{x}\in Z_k} \|\mathbf{x}-\mathbf{c}_k\|. \tag{3}$$

We call r_k the radius of BN[k] and require that classifier accepts only such patterns $\mathbf{x}$ that the inequality $\|\mathbf{x}-\mathbf{c}_k\| \le \beta r_k$ holds, where $\beta \in (0,+\infty)$. We can consider that the coefficient β is responsible for the degree of sensibility of the neuron. For example, if $\beta=1$, then only the interior of the green and blue dotted circles in Figures 3 can be attributed to the first and third classes, respectively. It is evident, that

$$\|\mathbf{x}-\mathbf{c}_k\| \le \beta r_k \Leftrightarrow \|\mathbf{x}-\mathbf{c}_k\|^2 \le (\beta r_k)^2 \Leftrightarrow \beta^2 r_k^2 - \|\mathbf{x}-\mathbf{c}_k\|^2 \ge 0.$$

Thus, using (2), we obtain the following equation for normalized inputs:

$$\|\mathbf{x}-\mathbf{c}_k\| \le \beta r_k \Leftrightarrow 2\mathbf{c}_k \cdot \mathbf{x} + \beta^2 r_k^2 - 1 - \|\mathbf{c}_k\|^2 \ge 0.$$

Therefore, single-threshold neuron TN[2, k] with the weight vector $2\mathbf{c}_k$, the bias $\beta^2 r_k^2 - 1 - \|\mathbf{c}_k\|^2$ and the step binary-valued activation can be used to verify whether a given pattern $\mathbf{x}$ is in the βr_k-neighborhood of the center $\mathbf{c}_k$ of kth bithreshold neuron.

Three units BN[k], WTA[k] and TN[2, k] are placed in the second layer of the network and are together responsible for the classification of patterns lying in the neighborhood of the kth centroid $\mathbf{c}_k$. Notice that the unit TN[2, k] is more "important" than other two units. Its activation is necessary for the successful classification of an input pattern as a member of the corresponding class. The decision of such classification is made in the case when at least one of remaining two units is activated. To implement the above classification rule using neural network approach, we can use a mini-block consisting of BN[k], WTA[k] and TN[2, k] in the second network layer and the single threshold neural unit TN[3, k] in the third layer, which has the bias -5 and whose three inputs are outputs of BN[k], WTA[k] and TN[2, k] taken with the weights 2, 2, and 4, respectively. The output of each mini-block is connected to the corresponding single-threshold node TN[4, i] in the output layer, where i is a number of the class for the classification of whose patterns the kth mini-block is intended.

Therefore, we obtain 4-layer neural network architecture in which the first layer is responsible for the augmentation and normalization of inputs, the second hybrid hidden layer consists of 3 kinds of neurons (BN, WTA and TN), the third hidden layer consisting of single-threshold neurons with only 3 inputs and the output layer is composed of l single-threshold neurons, where l is the number of classes. The second and third layers are crucial for this architecture and grow dynamically according to the design set and implementation details of the synthesis algorithm. The size of the second layer is 3 times greater than the size of the third layer.

Consider an example of the conceptual graph of the network that is designed employing the above 4-layer architecture. Let the network be designed to solve the binary classification task in two dimensions and let training patterns of the first class can be separated using 2 mini-blocks and a single mini-block suffices to recognize patterns belonging to the second class. The graph of the network is depicted in Figure 4, where threshold neurons are denoted by circles, bithreshold neurons—by concentric circles and winner-take-all neurons—by square boxes, all possible pairs of which are joined by lateral connections. The normalization and augmentation layer has 3 outputs, which are denoted in similar way as network inputs x_1 and x_2. Each mini-blocks consisting of 4 neurons is depicted inside a frame. The outputs of the first two mini-blocks are passed to inputs of the first output neurons, which corresponds to the first class. The last mini-block is connected to the output neuron, which is responsible of the recognition of patterns of the second class.

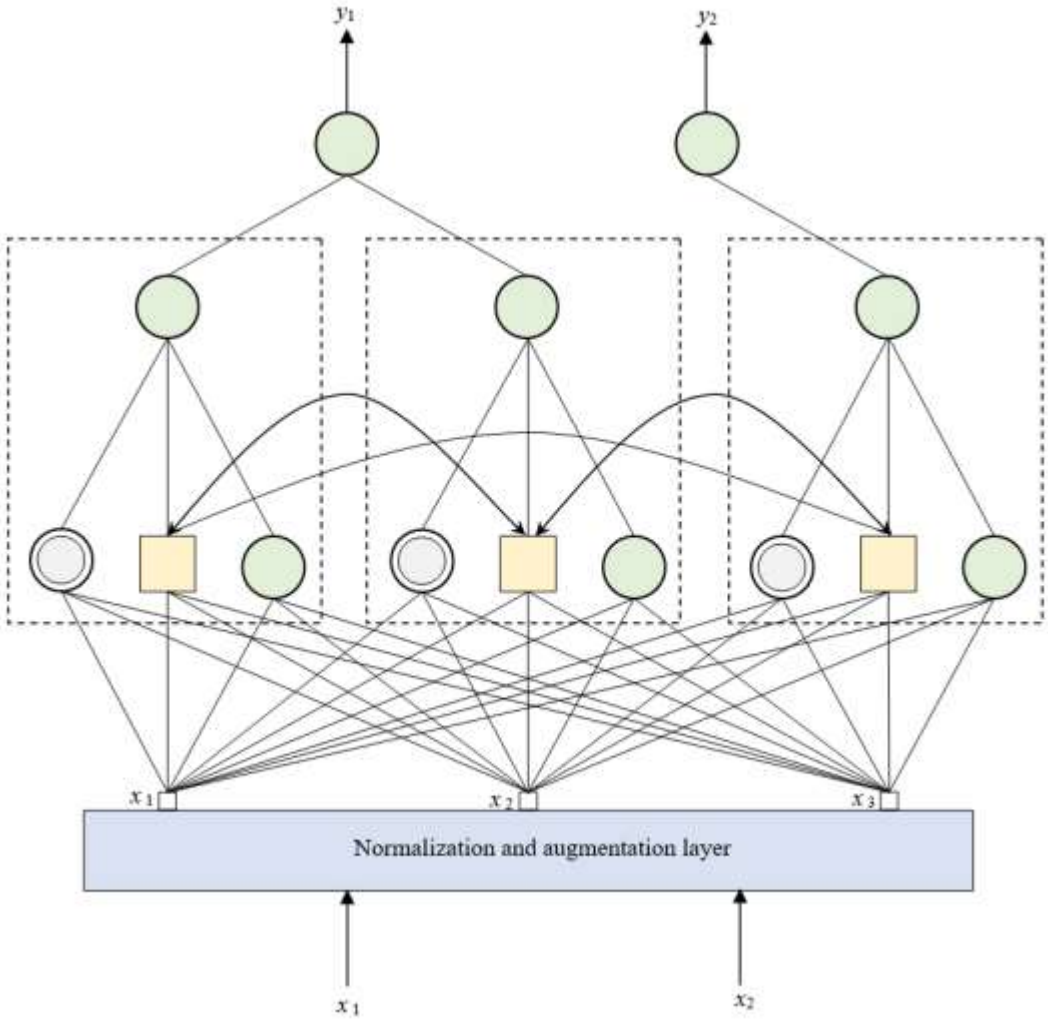

Figure 4: Example of 4-layer hybrid neural network Architecture

3.2.2. Synthesis Algorithm

The preceding explanation allows us to design the synthesis algorithm for a 4-layer neural network. Its principal part is the subroutine `Branch`, which synthesizes the part of the network responsible for the recognition of the patterns of the particular class (parameter *B*) by separating them from patterns from other classes (the set of all patterns is passed using parameter *A*). Let us describe it using the following pseudocode:

Branch(NN, OutputNode, *A*, *B*, α, β)

1. Copy patterns from *B* to *C*
2. `while` $C \neq \varnothing$:
3. $k \leftarrow k+1$
4. $q \leftarrow \min\{n, |C|\}$
5. Choice *q* random patterns from *C* to *D*
6. Find a solution of the system $\mathbf{w} \cdot \mathbf{x} = 1,\ \mathbf{x} \in D$
7. $\varepsilon_1 \leftarrow \alpha \max\{\ \mathbf{w} \cdot \mathbf{x} - 1 \mid \mathbf{x} \in A \setminus B,\ \mathbf{w} \cdot \mathbf{x} < 1\}$
8. $\varepsilon_2 \leftarrow \alpha \min\{\ \mathbf{w} \cdot \mathbf{x} - 1 \mid \mathbf{x} \in A \setminus B,\ \mathbf{w} \cdot \mathbf{x} > 1\}$
9. $X_k \leftarrow \{\ \mathbf{x} \in B \mid \varepsilon_1 < \mathbf{w} \cdot \mathbf{x} - 1 < \varepsilon_2\}$

10. $\mathbf{c}_k \leftarrow$ centroid of BN[k]
11. $r_k \leftarrow$ radius of BN[k]
12. $C \leftarrow C \setminus X_k$
13. Add BN[k] with the structure triplet $\left(\mathbf{w}, 1+\varepsilon_1, 1+\varepsilon_2\right)$ in the second layer
14. Add WTA[k] with the structure $\left(2\mathbf{c}_k, -\left\|\mathbf{c}_k\right\|^2\right)$ in the second layer
15. Add TN[2, k] with the structure $\left(2\mathbf{c}_k, \beta^2 r_k^2 - \left\|\mathbf{c}_k\right\|^2 - 1\right)$ in the second layer
16. Add TN[3, k] with the bias -5 in the third layer
17. Connect BN[k], WTA[k], TN[2, k] with TN[3, k] using weights 2, 2, 4
18. Connect TN[3, k] and OutputNode with the weight 1

Note that some steps of the previous subroutine might be explained in more detail. E.g., steps 10 and 11 are intended to search the center and radius of kth bithreshold neuron. We can employ here equations (1) and (3) using $Z_k = X_k$ or $Z_k = X_k \bigcap C$. Steps 7 and 8 correspond actually to step 11 of BasicSynthesis. They are useful to determine the independent shifts from central hyperplane $\mathbf{w} \cdot \mathbf{x} = 1$ and are intended to increase the gap between the decision hyperplanes of the current BN.

The main network synthesis algorithm can be described as follows:

HybridSynthesis(S, α, β)

1. `for` $i \leftarrow 1$ `to` l:
2. $B_i \leftarrow \varnothing$
3. `for` $i \leftarrow 1$ `to` m:
4. $\mathbf{x}_i \leftarrow \left(\mathbf{x}_i, 1\right) / \left\|\left(\mathbf{x}_i, 1\right)\right\|$
5. Add a small random noise to $\mathbf{x}_i$
6. Include $\mathbf{x}_i$ in A
7. Include $\mathbf{x}_i$ in B_{t_i}
8. Create 4-layer NN
9. $k \leftarrow 0$
10. `for` $i \leftarrow 1$ `to` l:
11. Add the node TN[4, i] in the output layer of NN and initialize its bias to zero
12. `Branch`(NN, TN[4, i], A, B_i, α, β)
13. `for` $i \leftarrow 1$ `to` $k-1$:
14. `for` $j \leftarrow i+1$ `to` k:
15. Set the lateral connection between WTA[i] and WTA[j]
16. `return` NN

Three parameters are used in HybridSynthesis(S, α, β): S—design set, α—a tolerance measure (see [8, 14]), and β—a factor by which the distance to the center is multiplied. Both α and β are suitable to adjust the classifier performance for a particular task. These hyperparameters alongside with the transformation in step 4 are designed to improve the generalization ability of the classifier. Both HybridSynthesis algorithm and its Branch subroutine use global counter of the network mini-blocks k. HybridSynthesis returns the resulting 4-layer neural network whose first layer performs normalization, the second layer consists of $3k$ neurons, the third layer—of k neurons, and the output layer—of l neurons.

The following proposition gives the characteristics of the performance of the HybridSynthesis.

Proposition. An arbitrary learning sample S consisting of m training pairs, each of which contains a n-dimensional pattern and its target class label, can be completely memorized by 4-layer hybrid neural network multiclass classifier with at most $k = \left(\left\lfloor m / \left(n+1\right)\right\rfloor + l\right)$ neurons in the third layer, $3k$ neurons in the second layer and l output binary threshold neurons if training patterns are in general position. The

network can be synthesized using HybridSynthesis algorithm in time $O\left(mn^2 + m^2\right)$, where l is the number of classes and $m \geq n$.

The proof is similar to the proof of the related proposition in [14] and is omitted here.

4. Experiment and Results

The capability of 2-layer and 4-layer neural network classifiers were tested on the medium-sized "Pen-Based Recognition of Handwritten Digits" (pendigits) dataset provided by UC Irvine Machine Learning Repository [24]. The dataset contains 10992 instances, which were obtained by collecting 250 samples from 44 writers. After a resampling the patterns have 16 input attributes. The target is the class code 0..9. The classes contain 901, 912, 934, 830, 906, 842, 855, 912, 855, and 846 members, respectively. Thus, it was the multiclass classification task with training pairs spread out relatively uniformly across all 10 classes.

The quality of the performance of classifiers that have treated in the work was compared with the performance of 3 classifiers: RBF support vector classifier (SVC), decision tree classifier, multilayer perceptron (MLP). Three classifiers from the current paper were examined. The first of them (NN2) was the basic 2-layer NN with bithreshold hidden layer, which was synthesized using BasicSynthesis algorithm. The other two were 4-layer hybrid neural networks produced by Synthesis(S, α, β) algorithm with $\alpha = 0.95$ and $\beta = 1.33$. The first of them (NN4a) used only "just" classified patterns in steps 10 and 11 of Branch subroutine, meanwhile the second one (NN4b)—all "properly" classified.

During the simulation standard versions of first three classifiers from Scikit-Learn library [25] were used (with gamma = 'scale', C = 0.1 for RBF SVC, max_depth = 7 for decision tree, 100 and 20 nodes in hidden layers for MLP). NN2, NN4a and NN4b was implemented using Numpy and Scikit-Learn libraries [5, 25].

In order to estimate the generalization abilities of classifiers the dataset was split into a training set and a testing set. In each trial 20% of randomly chosen patterns (2198 patterns) was moved to the testing set, the rest of the dataset—to the training set (the design set for NN2, NN4a and NN4b). The dataset split and the training were repeated 100 times.

The simulation results are presented in Table 1, which contains the average accuracies over all 100 trials.

Table 1
Experiment results on pendigits dataset

Classifier	Accuracy on training set (in %)	Accuracy on testing set (in %)
RBF SVC	98.09	97.77
Decision tree	92.99	91.21
MLP	99.32	98.12
NN2	100	59.34
NN4a	99.76	97.65
NN4b	99.87	98.01

The analysis of the experiment results allows us to conclude that:

- As expected, BasicSynthesis algorithm ensured exact memorization on the training data for NN2 but the network performed poorly outside the design set.
- HybridSynthesis(S, α, β) algorithm yielded networks with the almost same accuracy on the training set as NN2 but the significantly better accuracy on the testing set. Moreover, NN4a and NN4b had in average 12% fewer bithreshold neurons in the hidden layer.
- 4-layer network NN4b had the better classification accuracy than NN4a and overperformed all considered classifiers except MLP on the testing set.

5. Discussions

Proposed 4-layer hybrid neural network architecture allows to reduce some downsides of the basic 2-layer bithreshold architecture. Moreover, the modification of the step 11 of the BasicSynthesis algorithm, which is performed in steps 7 and 8 of Branch subroutine has following consequences:

- The network is suitable for the classification in the case when some classes have a nonempty intersection.
- The bounding hyperplanes are shifted independently from the central one (defined by the condition $\mathbf{w} \cdot \mathbf{x} = 1$). This leads in practice to the wider gap between the decision surfaces and decreases the number of necessary bithreshold neurons.
- The use of hyperparameter α also permits to increase the mentioned gap. It allows the network make some mistakes on the design set but, in general, improves its generalization abilities.

The main HybridSynthesis algorithm call Branch on each iteration of the loop, which is executed through class labels. These calls are independent and admit the parallel implementation. It can considerably speed up the synthesis computation using GPUs [5] or other similar hardware [3].

The optimal values of hyperparameters α and β are task-dependent and can be found using the grid search [25] or the random search [5].

6. Conclusions

We considered the new 4-layer neural network architecture, which can be useful for the design of multiclass classifiers. The experiment results prove that the combination of the power of bithreshold activations with the nearest neighbor rule (provided by the sublayer of WTA-neurons) and the regularization of the distance to the centers allowed to reduce the network overfitting and considerably improved the classifier's generalization capabilities.

The proposed HybridSynthesis algorithm has high degree of parallelism and the resulting model of the classifier can be fine-tuned using two hyperparameters. The synthesis algorithm is flexible enough and yields the network, which size depends on the dimensionality and the complexity of the concrete classification task. It should be noted that additional efforts are required for the enhancement of the generalization ability of the classifier via the decreasing of the number of the units in hidden layers and the reduction of the influence of the order in which learning pairs are selected during the synthesis.

4-layer classifier can be implemented as the part of different computational intellectual tool, i.e., in the smart classification subsystem of the linguistic interactive map [26].

7. References

[1] M. Lupei, A. Mitsa, V. Repariuk, V. Sharkan, Identification of authorship of Ukrainian-language texts of journalistic style using neural networks, Eastern-European Journal of Enterprise Technologies 1.2 (103) (2020): 30–36.

[2] N. Melnykova, V. Melnykov, N. Shahovska, and N. Lysa, The investigation of artificial intelligence methods for identifying states and analyzing system transitions between states, in: Proceedings of the IEEE 15th International Conference on Computer Sciences and Information Technologies, CSIT 2020, Lviv, Ukraine, 2020, pp. 70–75.

[3] I. Tsmots, R. Tkachenko, V. Teslyuk, Y. Opotyak, V. Rabyk, Hardware Components for Nonlinear Neuro-like Data Protection in Mobile Smart Systems, in: Proceeding of the IEEE 17th International Conference on Computer Sciences and Information Technologies, CSIT 2022, Lviv, Ukraine, 2022, pp. 198–202.

[4] 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, vol. 77, Springer, Cham, 2022, pp. 696–713.

[5] 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.

[6] M. Lupei, A. Mitsa, V. Sharkan, S. Vargha, and V. Gorbachuk, The identification of mass media by text based on the analysis of vocabulary peculiarities using support vector machines, in: International Conference on Smart Information Systems and Technologies, SIST 2022, Astana, Kazakhstan, 2022, pp. 471–476.
[7] S. Haykin, Neural Networks and Learning Machines, 3rd ed. Pearson Education, Upper Saddle River, NJ, 2009.
[8] V. Kotsovsky, A. Batyuk, On-line relaxation versus off-line spectral algorithm in the learning of polynomial neural units, in: S. Babichev et al. (Eds.), Communications in Computer and Information Science, vol. 1158, Springer, Cham, 2020, pp. 3–21.
[9] D. R. Haring, Multi-threshold threshold elements, IEEE Transactions on Electronic Computers EC-15.1 (1966): 45–65.
[10] V. Deolalikar, A two-layer paradigm capable of forming arbitrary decision regions in input space, IEEE Transactions on Neural Networks, 13.1 (2002): 15–21.
[11] R. Takiyama, The separating capacity of a multithreshold threshold element, IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-7.1 (1985): 112–116.
[12] V. Kotsovsky, A. Batyuk, I. Mykoriak, The computation power and capacity of bithreshold neurons, in: Proceedings of the IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2020, Lviv, Ukraine, 2020, pp. 28–31.
[13] 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.
[14] V. Kotsovsky, F. Geche, and A. Batyuk, Bithreshold neural network classifier, in: Proceedings of the IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2020, vol. 1. Lviv, Ukraine, 2020, pp. 32–35.
[15] I. Izonin, R. Tkachenko, R. Pidkostelnyi, O. Pavliuk, V. Khavalko, A. Batyuk, Experimental evaluation of the effectiveness of ANN-based numerical data augmentation methods for diagnostics tasks, in: CEUR Workshop Proceedings, vol. 3038, 2021, pp. 223–232.
[16] M. Lupei, A. Mitsa, I. Povkhan, V. Sharkan, Determining the eligibility of candidates for a vacancy using artificial neural networks, in: Proceedings of the IEEE 3rd International Conference on Data Stream Mining and Processing, DSMP 2020, Lviv, Ukraine, 2020, pp. 18–22.
[17] V. Teslyuk, A. Kazarian, N. Kryvinska, I. Tsmots, Optimal artificial neural network type selection method for usage in smart house systems, Sensors 21.1 (2021).
[18] F. Geche, V. Kotsovsky, A. Batyuk, S. Geche, M. Vashkeba, Synthesis of time series forecasting scheme based on forecasting models system, in: CEUR Workshop Proceedings, vol. 1356, 2015, pp. 121–136.
[19] A. Blum, R. Rivest, Training a 3-node neural network is NP-Complete, Neural Networks, 5.1 (1992): 117–127.
[20] 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, vol. 16, 2021, pp. 12674–12685.
[21] E. B. Baum, On the capabilities of multilayer perceptrons, Journal of Complexity, 4.3 (1988): 193–215.
[22] C. Yun., S. Sra, A. Jadbabaie, Small ReLU networks are powerful memorizers: a tight analysis of memorization capacity, in: Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 15532–15543.
[23] R. Vershynin, Memory capacity of neural networks with threshold and rectified linear unit activations, SIAM Journal on Mathematics of Data Science 2.4 (2020): 1004–1033.
[24] D. Dua, C. Graff, UCI Machine Learning Repository, 2017. URL: http://archive.ics.uci.edu/ml.
[25] F. Pedregosa et al., Scikit-learn: machine learning in Python, Journal of Machine Learning Research 12 (2011): 2825–2830.
[26] M. Lupei, M. Shlahta, O. Mitsa, Y. Horoshko, H. Tsybko, and 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, Spišská Kapitula, Slovakia, 2022, pp. 384–387.