LEARNING OF TWO-LAYER BITHRESHOLD NEURAL NETWORKS

Kotsovsky V. M.[1], Batyuk A. Ye.[2], Melnychenko T. V.[1]

[1] *IMST Department, Uzhhorod National University, Uzhhorod, 88000, Ukraine, kotsavlad@gmail.com*
[2] *ACS Department, Lviv Polytechnic National University, Lviv, 79013, Ukraine, abatyuk@gmail.com*

A bithreshold approach in neural computations provide us with the more powerful neural units than classical thresholds ones [1, 2]. The advantage of the expressive power of multithreshold elements was estimated in [3]. We consider 2-layer neural networks consisting of the simplest multithreshold elements, namely, bithreshold units. We show that the use of bithreshold neurons instead of single-threshold units allows to halve the size of the hidden layer.

A computation unit with n real inputs $x_1,\dots,x_n$ and single output y is said to be a *bithreshold neuron* (BN) with the weight vector $\mathbf{w}=(w_1,\dots,w_n)\in \mathrm{R}^n$ and thresholds $t_1,t_2 \in \mathrm{R}\ (t_1<t_2)$ if $y=f_{t_1,t_2}(\mathbf{w}\cdot\mathbf{x})$ [2], where $\mathbf{w}\cdot\mathbf{x}$ is the dot product of the vectors $\mathbf{w}$ and $\mathbf{x}$, f_{t_1,t_2} is the bithreshold activation function defined in the following way:

$$f_{t_1,t_2}(z)=\begin{cases}-1, & \text{if } t_1<z<t_2,\\ 1, & \text{otherwise}\end{cases}$$

The BN is completely defined by the *structure vector* $(\mathbf{w},t_1,t_2)$.

Let us consider a two-layer feedforward fully-connected *bithreshold neural network* (see Figure 1). It has n nodes in the input layer, several bithreshold neurons in the hidden layer and a single threshold neuron in the output layer. A bithreshold neuron can be conveniently represented graphically as shown in Figure 1.

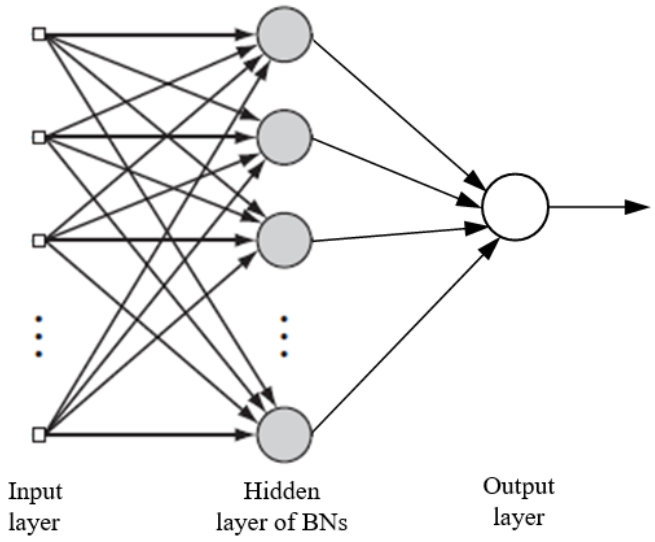

Fig. 1. A two-layer bithreshold neural network with a single output

We are interested in finding a computationally efficient algorithm for training network to compute an arbitrary *dichotomy* (A^+, A^-) of the finite set A [4]. We adopt Baum's idea [5] in the algorithm design.

Suppose that all patterns of the finite set A are in the general position in n-dimensional space (i.e., every subset of n or fewer vectors is linearly independent) and $|A| = m$. Without loss of generality we can assume that $A^+\neq\varnothing,\ A^-\neq\varnothing$. Consider the following algorithm:

Stage 1. Initialization

1.1. $k\leftarrow 0$

1.2. if $|A^-|<|A^+|$ then

$B \leftarrow A^-$, $C \leftarrow A^+$, $v \leftarrow 1$

else

$B \leftarrow A^+$, $C \leftarrow A^-$, $v \leftarrow -1$

Stage 2. Construction of the hidden layer

2.1. $k \leftarrow k+1$

2.2. $r \leftarrow \min\{n, |B|\}$

2.3. Extract r pattern from B to X

2.4. Find a solution $\mathbf{w}$ of the linear system $\mathbf{x}^i \cdot \mathbf{w} = 1$, $\mathbf{x}^i \in X$, $i = \overline{1,r}$

2.5. $\varepsilon \leftarrow \alpha \min\left\{|\mathbf{w} \cdot \mathbf{x} - 1| \,\middle|\, \mathbf{x} \in C\right\}$

2.6. $B \leftarrow B \setminus \left\{\mathbf{x} \in B \,\middle|\, |\mathbf{w} \cdot \mathbf{x} - 1| < \varepsilon\right\}$

2.7. Add the BN with the structure $(\mathbf{w}, 1-\varepsilon, 1+\varepsilon)$ in the hidden layer

2.8. If $B \neq \varnothing$, then go back to step 2.1

Stage 3. Definition of the output neuron

Add to the network the output threshold unit with k-dimensional weight vector $(v,\ldots,v)$ and threshold $(k-0.5)v$.

In our algorithm k denotes the index of the BN, which will be added in the network. The parameter α can be considered as a tolerance measure of our bithreshold networks. If $0 < \alpha < 1$, then the network performs well on training set (training error is near 0) but can be "overfitted" and less effective outside the training set. If α is slightly greater than 1, the network can have the better generalization ability.

Proposition. *An arbitrary dichotomy of m-set of n-dimensional vectors in general position can be computed by two-layer bithreshold neural network with at most* $\lceil m/(2n) \rceil$ *neurons in the hidden layer for which there exists* $O(m^2 + mn^2)$*-time learning algorithm.*

It should be noted that our algorithm is not valid in the degenerate case. Numerous modification of the stage 2 can be proposed. For example, we can add some noise to our data for the purpose of avoiding the degeneracy.

Note that our algorithm appears to be more like synthesis than learning, because we change the structure of the network on each iteration of the stage 2. But we can predefine $\lceil m/(2n) \rceil$ nodes in the hidden layer and use the parallel implementation of the stage 2.

REFERENCES

1. Deolalikar, V.: A two-layer paradigm capable of forming arbitrary decision regions in input space. IEEE Transactions on Neural Networks **13**(1), 15–21 (2002)
2. Kotsovsky, V., Geche, F., Batyuk, A.: On the computational complexity of learning bithreshold neural units and networks. In: Lytvynenko, V. et al. (eds.) Lecture Notes in Computational Intelligence and Decision Making. ISDMCI 2019. Advances in Intelligent Systems and Computing, vol. 1020, pp. 189-202. Springer, Cham (2020)
3. Olafsson, S., Abu-Mostafa, Y. S: The capacity of multilevel threshold function. IEEE Transactions on Pattern Analysis and Machine Intelligence 10(2), 277–281 (1988)
4. Anthony, M.: Discrete Mathematics of Neural Networks: Selected Topics. SIAM, Philadelphia (2001)
5. Baum, E. B.: On the capabilities of multilayer perceptrons. Journal of Complexity, **4**(3), 193–215 (1988).