On the Projection of k-Valued Non-linearly Separable Problems into m-Valued Linearly Separable Problems

Igor Aizenberg

Department of Computer Science, Texas A&M University-Texarkana
P.O. Box 5518, Texarkana, TX 75505-5518, U.S.A.
igor.aizenberg@tamut.edu

Abstract. In this paper, we observe a new approach to learn non-linearly separable problems using a single multi-valued neuron. It is shown that a *k*-valued problem, which is non-linearly separable in the *n*-dimensional space can be projected into an *m*-valued (where *m=kl*) linearly separable problem in the same space. This projection can be utilized through a periodic activation function for the multi-valued neuron. Then the initial problem can be learned by a single multi-valued neuron using its learning algorithm. This approach is illustrated by the examples of such problems as XOR, Parity *n*, mod *k* addition of *n* variables and some benchmarks using a single multi-valued neuron.

Keywords: Complex-valued neural networks, Multi-valued neuron, Derivative-free learning, XOR, Parity *n*, Mod *k* addition.

1 Introduction

It is commonly known that a linearly separable (threshold) Boolean function and a linearly separable function whose range is Boolean can be learned by a single neuron that is by the classical perceptron – a neuron with a threshold activation function 1]. It is also known that a linearly separable (threshold) multiple-valued function can also be learned by a single neuron – the multi-valued neuron 2].

However, it is commonly known that, for example, almost all real world classification problems are non-linearly separable, that is they are described by non-linearly separable (non-threshold) functions. Suppose we have to solve some *n*-dimansional *k*-class classification peoblem and this problem is described by some non-threshold function, which is either a discrete function of *k*-valued logic or a function with a descete *k*-valued range. The commonly used approach for solving such a problem is its consideration in the larger dimensional space. Actually, one of the ways to utilize this approach is a feedforward neural network, where hidden neurons form a new space, and a problem becomes linearly separable and solvable 1]. Another popular approach for solving this problem is the support vector machine (SVM) introduced in 3], where a larger dimensional space is formed using the support vectors and a problem becomes linearly separable and solvable in this new space.

In this paper, we will consider how the same problem can be approached from a different side. Suppose we have some *n*-dimansional *k*-valued non-linearly separable

K. Madani et al. (Eds.): Computational Intelligence, SCI 343, pp. 223–235.
springerlink.com

problem. This means that this problem is described by some non-threshold *k*-valued function of *n* variables. We will discover here how this non-threshold *k*-valued function of *n* variables can be projected into an *m*-valued partually defined threshold function of the same *n* variables (where $m > k$). This means that our *n*-dimansional *k*-valued non-linearly separable problem will become an *n*-dimansional *m*-valued linearly separable problem and therefore it will be possible to learn it using a single multi-valued neuron. Thus, we will reach a linear separation without extension of that initial *n*-dimansional space where our problem is defined.

A main tool, which will be used here to reach the declared goals, is the multi-valued neuron (MVN). The MVN was introduced in 4]. This neuron is based on the concept of multiple-valued logic over the field of complex numbers. This concept was introduced in 5], then presented in detail in 6], and further developed in 7]. The continuous MVN was introduced in 8] and then developed in 9]. A single MVN has significantly higher functionality than a single sigmoidal neuron. Another advantage of the MVN is its learning algorithm, which is derivative-free. The MVN's input/output mapping is always described by some threshold function of *k*-valued logic.

Another important tool, which will be used in this paper is the universal binary neuron (UBN). It was introduced in 10], then its theory was presented in detail in 2], and its modified learning algorithm is considered in 11]. The UBN is a Boolean neuron, its inputs and output are binary, thus it implements those input/output mappings, which are described by Boolean functions. The most important advantage of the UBN over traditional neurons is its ability to learn non-linearly separable Boolean functions. This is possible because the UBN employs complex-valued weights and the original activation function.

In this paper, we use the idea, which is behind the UBN activation function, to modify the MVN activation function. This idea is the periodicity of an activation function. A periodic activation function, which is used in the UBN makes it possible to project the Boolean (2-valued) logic into an *m*-valued logic, where $m > 2$. This makes it possible to project any non-linearly separable Boolean function of *n* variables into a partially defined (only on Boolean inputs) linearly-separable function of the same *n* variables. This approach can be generalized for non-linearly separable functions of *k*-valued logic for $k > 2$, and such functions can be projected into partially defined functions of *m*-valued logic, where $m > k$. This paper extends the author's recent work 12], where some preliminary results were reported.

2 Multi-Valued Neuron (MVN)

2.1 Discrete and Continuous MVN

The discrete MVN was proposed in 4]. It is based on the principles of multiple-valued threshold logic over the field of complex numbers. The discrete MVN performs a mapping between *n* inputs and a single output. This mapping is described by a multiple-valued (*k*-valued) threshold function of *n* variables $f(x_1, \ldots, x_n)$. It is important to specify that we consider here multiple-valued logic over the field of

complex numbers 2], 6]. While in traditional multiple-valued logic its values are encoded by the integers from the set $K = \{0,1,...,k-1\}$, in the one over the field of complex numbers they are encoded by the kth roots of unity $E_k = \{\varepsilon^0, \varepsilon, \varepsilon^2, ..., \varepsilon^{k-1}\}$, where $\varepsilon^j = e^{i2\pi j/k}$, $j = 0,...,k\text{-}1$, (i is an imaginary unity). A k-valued threshold function describing the MVN input/output mapping, can be represented using $n+1$ complex-valued weight as follows

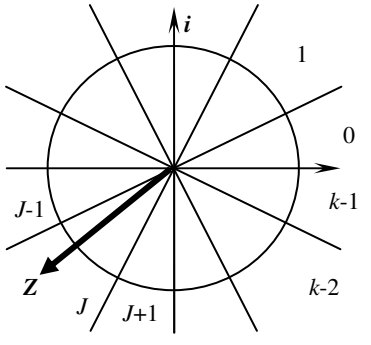
$$f(x_1, \ ..., \ x_n) = P(w_0 + w_1 x_1 + ... + w_n x_n), \tag{1}$$

where $x_1, ..., x_n$ are the variables (neuron inputs) and $w_0, w_1, ..., w_n$ are the weights. The values of the function and of the neuron inputs are the k^{th} roots of unity: $\varepsilon^j = e^{i2\pi j/k}$, $j \in \{0,1,...,k\text{-}1\}$, i is an imaginary unity. P is the activation function

$$P(z) = e^{i2\pi j/k}, \text{ if } 2\pi j / k \le \arg z < 2\pi(j+1)/k, \tag{2}$$

where j=0, 1, ..., k-1 are values of k-valued logic, $z = w_0 + w_1 x_1 + ... + w_n x_n$ is the weighted sum, arg z is the argument of the complex number z. It is important to mention that function (2), which was introduced in 5], is historically the first known complex-valued activation function. Function (2) divides a complex plane into k equal sectors and maps the whole complex plane into a set of k^{th} roots of unity (see Fig. 1).

This approach was later generalized for the continuous case. The continuous MVN has been proposed in 8] and then developed in 9]. The continuous case corresponds to $k \to \infty$ in (2). If the number of sectors $k \to \infty$ (see Fig. 1), then the angular size of



$$P(z) = \exp\left(j \cdot i2\pi / k\right)$$

Fig. 1. Geometrical interpretation of the discrete MVN activation function.

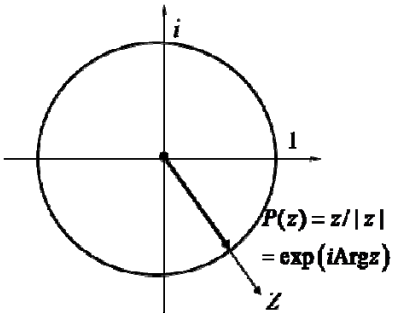


Fig. 2. Geometrical interpretation of the continuous MVN activation function.

a sector tends to 0. Hence, an activation function in this case becomes simply a projection of the weighted sum $z = w_0 + w_1 x_1 + ... + w_n x_n$ onto the unit circle:

$$P(z) = \exp(i\ (\arg z)) = e^{iArg\ z} = z/\mid z \mid, \tag{3}$$

where z is the weighted sum, Arg z is a main value of its argument (phase) and $|z|$ is the absolute value of the complex number z.

Activation function (3) is illustrated in Fig. 2. It maps the whole complex plane into the unit circle. Evidently, a hybrid MVN (with continuous inputs/discrete output, discrete inputs/continuous output, hybrid inputs and discrete or continuous output) can also be easily considered.

2.2 MVN Learning

MVN learning algorithm is identical for both discrete and continuous neurons. The most important property of MVN learning is that it is derivative-free. There are two MVN learning algorithms. One of them, which was proposed in 2], is computationally more efficient. It is based on the error-correction learning rule. If T is the desired neuron's output and Y is the actual one, then $\delta = T - Y$ is the error (see Fig. 3) that determines that adjustment of the weights, which is performed as follows:

$$W_{r+1} = W_r + \frac{C_r}{(n+1)} \delta \overline{X}\ , \tag{4}$$

or, as it was suggested in 9], as follows

$$W_{r+1} = W_r + \frac{C_r}{(n+1)|z_r|} \delta \overline{X}\ , \tag{5}$$

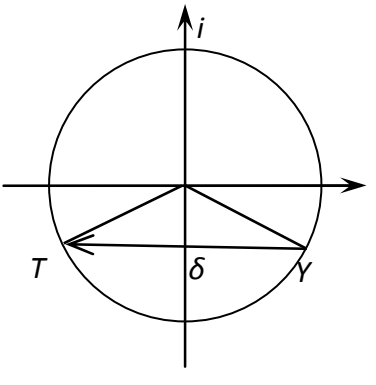


Fig. 3. Geometrical interpretation of the MVN learning rule

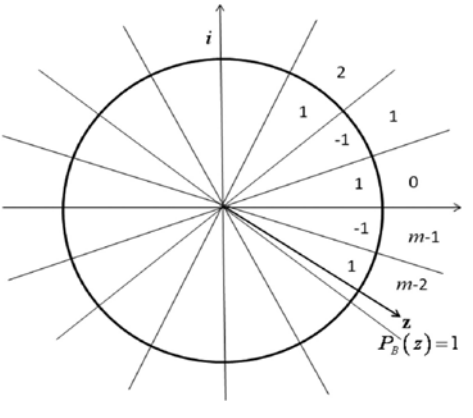


Fig. 4. Geometrical interpretation of the UBN activation function

where $\overline{X}$ is the vector of inputs with the components complex-conjugated, n is the number of neuron inputs, δ is the neuron's error, r is the number of the learning iteration, W_r is the current weighting vector (to be corrected), W_{r+1} is the following weighting vector (after correction), C_r is the constant part of the learning rate (it may always be equal to 1), and $|z_r|$ is the absolute value of the weighted sum obtained on the r^{th} iteration. A factor $1/|z_r|$ in (5) is a variable part of the learning rate. The use of it can be important for learning highly nonlinear input/output mappings.

3 Universal Binary Neuron

The UBN was introduced in 10]. It is presented in detail in 2], and its modified learning algorithm was recently considered in 11]. A key idea behind the UBN is the use of complex-valued weights and an original activation function for learning non-linearly separable Boolean functions.

If k=2 in (2) then the complex domain is divided into two parts (the top semiplane ("1") and the bottom semiplane ("-1")):

$$P(z)=\begin{cases} 1, & 0 \le \arg(z) < \pi \\ -1, & \pi \le \arg(z) < 2\pi. \end{cases}$$

However, this activation function does not increase the neuron's functionality: although the weights are complex, the neuron still can only learn linearly separable Boolean functions [2]. It was suggested in 11] to use the following periodic activation function

$$P_B(z) = (-1)^j, \text{if } \; 2\pi j / m \le \arg(z) < 2\pi(j+1)/m; \; m = 2l, l \in \mathbb{N}, \tag{6}$$

where l is some positive integer, j is a non-negative integer $0 \le j < m$. Activation function (6) is illustrated in Fig. 4. It separates the complex plane into $m = 2l$ equal sectors and determines the neuron's output by the alternating periodic sequence of 1, -1, 1, -1,…1, -1 depending on the parity of the sector's number.

Activation function (6) determines the most important advantage of the UBN. It is its ability to learn non-linearly separable Boolean functions.

For example, the favorite non-linearly separable problems of the research community - XOR and Parity n become linearly separable and can be learned by a single UBN with activation function (6). Table 1 shows how the XOR problem can be solved using a single UBN with the activation function (6) ($l = 2, m = 4$).

The UBN learning algorithm is presented in detail in 11]. It is based on the same learning rules (4) and (5) as the MVN learning algorithm. The choice of the desired output is based on the closeness of the current weighted sum to the right or left adjacent sector.

Table 1. Solving the **XOR** problem using a single UBN with the weighting vector (0, *i*, 1)

#	x_1	x_2	$z = w_0 + w_1 x_1 + w_2 x_2$	$P_B(z)$	XOR= $= x_1 \underset{\text{mod}\,2}{\oplus} x_2$
1)	**1**	**1**	1 + *i*	**1**	**1**
2)	**1**	**-1**	-1 + *i*	**-1**	**-1**
3)	**-1**	**1**	1 - *i*	**-1**	**-1**
4)	**-1**	**-1**	-1 - *i*	**1**	**1**

Let us analyze activation function (6). It is a periodic function with the period 2. Comparing this function with (2), we may conclude that (6) projects the Boolean logic into an *m*-valued logic. Activation function (6) may also be written in the following form:

$$P_B(z) = j \bmod 2, \text{ if } 2\pi j / m \le \arg z < 2\pi (j+1)/m, \quad j = 0,1,...,m-1;\ m = 2l, l \ge 2. \tag{7}$$

It was distinguished in 11] that if some non-threshold Boolean function can be learned using a single UBN with activation function (6) and its weighting vector $(w_0, w_1, ..., w_n)$ results from the learning process, then there exists a partially defined (on the Boolean variables) *m*-valued threshold function, which can be implemented by a single MVN with the same weighting vector. This means that a non-threshold Boolean function, which cannot be learned using a traditional perceptron, can be projected to a partially defined *m*-valued threshold function, which can be learned using a single MVN. Let us consider how the same approach can be used to learn using a single neuron not only non-linearly separable Boolean functions, but also non-linearly separable multiple-valued functions.

4 A Periodic Activation Function for the MVN

Let $E_k = \{1, \varepsilon_k, \varepsilon_k^2, ..., \varepsilon_k^{k-1}\}$ (where $\varepsilon_k = e^{i2\pi/k}$ is the primitive k^{th} root of unity) be the set of the k^{th} roots of unity. Let *O* be the continuous set of the points located on the unit circle. Let $K = \{0,1,...,k-1\}$ be the set of the values of *k*-valued logic. Let $f(x_1,...,x_n)$ be a function and either $f: E_k^n \to K$ or $f: O^n \to K$. Hence, the range of f is discrete, while its domain is either discrete or continuous. In general, its domain may be even hybrid. If some function $f(y_1, ..., y_n), y_i \in [a_j, b_j], a_j, b_j \in \mathbb{R}, j = 1,...,n$ is defined on the bounded subdomain $D^n \subset \mathbb{R}^n$ ($f: D^n \to K$), then it can be easily transformed to $f: O^n \to K$ by a simple linear transformation applied to each variable:

$y_j \in [a_j, b_j] \Rightarrow \varphi_j = \dfrac{y_j - a_j}{b_j - a_j}\alpha \in [0, 2\pi]$, and then $x_j = e^{i\varphi_j} \in O, j = 1, 2, ..., n$ is the $j = 1, ..., n; 0 < \alpha < 2\pi$,
complex number located on the unit circle. Hence, we obtain the function $f(x_1, \ ..., \ x_n) : O^n \to K$.

If this function $f(x_1, ..., x_n)$ is not a *k*-valued threshold function, it cannot be learned by a single MVN with activation function (2).

Let us project the *k*-valued function $f(x_1, ..., x_n)$ into an *m*-valued logic, where $m = kl$, l is integer and $l \geq 2$. To do this, let us define the following new discrete activation function for the MVN:

$$P_l(z) = j \bmod k, \text{ if } 2\pi j / m \leq \arg z < 2\pi (j+1) / m, \quad j = 0, 1, ..., m-1; \ m = kl, l \geq 2. \tag{8}$$

This definition is illustrated in Fig. 5. Activation function (8) separates the complex plane onto m equal sectors and $\forall t \in K$ there are exactly l sectors, where (8) equals to t.

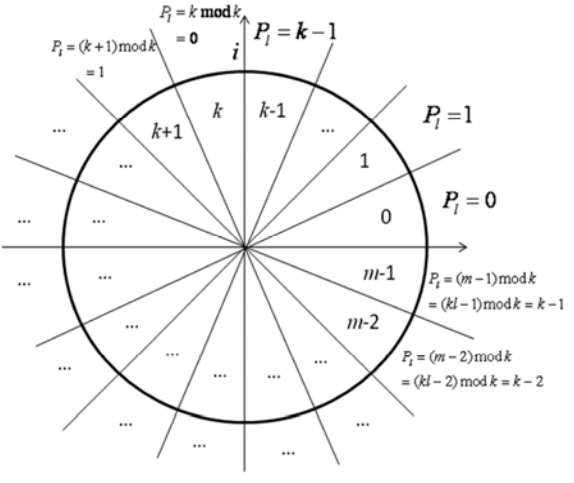


Fig. 5. Geometrical interpretation of the *l*-repeated multiple discrete-valued MVN activation function (7)

This means that activation function (8) establishes mappings from E_k into $E_m = \{1, \varepsilon_m, \varepsilon_m^2, ..., \varepsilon_m^{k-1}\}$ and from K into $M = \{0, 1, ..., k-1, k, k+1, ..., m-1\}$, respectively. Since $m = kl$ then each element from M and E_m has exactly l prototypes in K and E_k, respectively. In turn, this means that the MVN's output,

depending on which one of the m sectors (whose ordinal numbers are determined by the elements of the set M) the weighted sum is located in, is equal to

$$\underbrace{\underbrace{0,1,...,k-1}_{0},\underbrace{0,1,...,k-1}_{1},...,\underbrace{0,1,...,k-1}_{l-1}}_{lk=m}. \tag{9}$$

Hence the MVN's activation function in this case becomes k-periodic.

Activation function (8) projects a k-valued function $f(x_1,...,x_n)$ into m-valued logic. This projection becomes very attractive, if $f(x_1,...,x_n)$, being a non-threshold function in k-valued logic, becomes a partially defined threshold function in m-valued logic. The latter means that $f(x_1,...,x_n)$ can be learned it using a single MVN. It will be shown below that this is definitely the case.

It is important to mention that if $l=1$ in (8) then $m=k$ and activation function (8) coincides with activation function (2) accurate within the interpretation of the neuron's output (if the weighted sum is located in the j^{th} sector then according to (2) the neuron's output is equal to $e^{ij2\pi/k}=\varepsilon^j \in E_k$, which is the j^{th} k^{th} root of unity, while in (8) it is equal to $j \in K$). Evidently, the multiple-valued activation function (8) is a generalization of the binary activation function (7) for the multiple-valued case.

5 Learning Algorithm for the MVN with a Periodic Activation Function

To make the approach proposed in Section 3 active, it is necessary to develop an efficient learning algorithm for the MVN with activation function (8). Such an algorithm will be presented here.

As it was mentioned above (Section 2), one of the MVN learning algorithms is based on either of error-correction learning rules (4) or (5). Let us adapt this algorithm to activation function (8).

Let $f(x_1,...,x_n)$ be a function of k-valued logic and $f: E_k^n \to K$ or $f: O^n \to K$. Let this function be non-threshold and therefore it is not possible to learn it using a single MVN with activation function (2). Let us try to learn it in m-valued logic using a single MVN with activation function (8). Thus, the expected result of this learning process is the representation of $f(x_1,...,x_n)$ according to (1), where the activation function P_l determined by (8) substitutes for the activation function P determined by (2).

This learning process may be based on either of the same learning rules (4) or (5), but applied to $f(x_1,...,x_n)$ as to the m-valued function. To use these learning rules, it is necessary to determine a desired output. Unlike the case of the MVN with activation function (2), a desired output in terms of m-valued logic cannot be determined unambiguously for the MVN with activation function (8). According to

(8), there are exactly l sectors on the complex plane out of m, where this activation function equals to the given desired output $t \in K$. Therefore, there are exactly l out of m^{th} roots of unity that can be used as the desired outputs in (4) or (5). Which one of them should we choose? We suggest using the same approach that was used in the error-correction learning algorithm for the UBN 2], 11]. UBN's activation function (6) determines an alterning sequence with respect to sectors on the complex plane. Hence, if the actual output of the UBN is not correct, in order to make the correction, we can "move" the weighted sum into either of the sectors adjacent to the one where the current weighted sum is located. It was suggested to always move it to the sector, which is closest to the current weighted sum (in terms of the angular distance). The convergence of this learning algorithm was proven in 2].

Let us use the same approach here. Activation function (8) determines k-periodic sequence (9) with respect to sectors on the complex plane. Suppose that the current MVN's output is not correct and the current weighted sum is located in the sector $s \in M = \{0,1,...,m-1\}$. Since $l \geq 2$ in (8), there are l sectors on the complex plane, where function (8) takes the correct value. Two of these l sectors are the closest ones to sector s (from right and left sides, respectively). From these two sectors, we choose sector q whose border is closest to the current weighted sum z. Then either of learning rules (4) or (5) can be applied. Hence, the learning algorithm for the MVN with activation function (8) can be described as follows. Let us have N learning samples for the function $f(x_1,...,x_n)$ to be learned and $j \in \{1,...,N\}$ be the number of the current learning sample (initially, $j = 1$). One iteration of the learning process consists of the following steps:

1) Set r=1, j=1, and Learning="False".
2) Check (1) with the activation function (8) for the learning sample j.
3) If (1) holds then set $j = j + 1$, otherwise set Learning='True' and go to Step 5.
4) If $j \leq N$ then go to Step 2, otherwise go to Step 9.
5) Let z be the current value of the weighted sum and $P(z) = \varepsilon^s, s \in M$, $P(z)$ is the activation function (2), where m is substituted for k. Hence the MVN's actual output is $P_l(z) = s \bmod k$. Find $q_1 \in M$, which determines the closest sector to the s^{th} one, where the output is correct, from the right, and find $q_2 \in M$, which determines the closest sector to the s^{th} one, where the output is correct, from the left (this means that $q_1 \bmod k = d$ and $q_2 \bmod k = d$).
6) If $\left(\arg z - \arg\left(e^{i(q_1+1)2\pi/m}\right)\right) \bmod 2\pi \leq \left(\arg\left(e^{iq_2 2\pi/m}\right) - \arg z\right) \bmod 2\pi$ then $q = q_1$ else $q = q_2$.
7) Apply the learning rule (4) or (5) to adjust the weights.
8) Set $j = j + 1$ and return to Step 4.

9) If Learning='False' then go to Step 10, otherwise set $r=r+1$, $j=1$, Learning='False' and go to Step 2.
10) End.

Since according to this learning algorithm the learning of a k-valued function is reduced to the learning of a partially defined m-valued function, the convergence of the learning algorithm may be proven in the similar manner as the convergence of the learning algorithm based on rule (4) and of the UBN learning algorithm were proven in 2].

6 Simulation Results

To confirm the efficiency of the proposed activation function and learning algorithm, they were checked for the following three problems using a software simulator written in Borland Delphi 5.0 running on a PC with the Intel® Core™2 Duo CPU.

6.1 Wisconsin Breast Cancer (Diagnostic)

This famous benchmark database was downloaded from the UC Irvine Machine Learning Repository 13]. It contains 569 samples that are described by 30 real-valued features. There are two classes ("malignant" and "benign") to which these samples belong.

A single MVN with activation function (2) with $k=2$ fails to learn the entire data set because it is non-linearly separable. However, a single MVN with activation function (8) with $k=2, l=2, m=4$ learns the problem with no errors. Ten runs of the learning process started from different random weights resulted in convergence after 649-1300 iterations, which took a few seconds.

We have also tested the ability of a single MVN to solve classification problem. 10-fold cross validation was used as it is recommended in 13]. Every time the data set was divided into a learning set (400 samples) and a testing set (169) samples. After the learning set was learned completely with no errors, the classification of the testing set samples was performed. A classification rate of 96.5%-97.5% was achieved. This is comparative to the best known result (97.5%) 13]. However, it is important to note that our method solves the problem using just a single neuron, while in the previous works either different networks or linear programming methods were used.

6.2 Sonar

This famous benchmark database was also downloaded from the UC Irvine Machine Learning Repository 13]. It contains 208 samples that are described by 60 real-valued features. There are two classes ("mine" and "rock") to which these samples belong.

A single MVN with activation function (2) with $k=2$ fails to learn the entire data set even after 1,000,000 iterations. However, the single MVN with activation function (8) with $k=2, l=2, m=4$ learns the problem with no errors. Ten runs of the learning process started from different random weights resulted in convergence after 65-180 iterations, which took a few seconds.

We have also tested the ability of a single MVN to solve the classification problem. This set is initially divided by its creators into learning (104 samples) and testing (104 samples) subsets. After the learning set was learned completely with no errors, the classification of the testing set samples was performed. The classification rate of 88.1%-91.5% was achieved. This is comparative to the best known results reported in 14] – 94% (Fuzzy Kernel Perceptron), 89.5% (SVM), and in 9] - 88%-93% (MLMVN). However, here the problem was solved using just a single neuron, while in the previous works different neural and kernel-based networks were used.

6.3 k-Valued Non-threshold Function

Let us consider the following fully defined function of 3 variables of 4-valued logic $f(x_1, x_2, x_3) = (x_1 + x_2 + x_3) \bmod 4$ (see the first four columns of Table 2). This function, which is the analogue of parity 3 function in the Boolean logic, is non-threshold in 4-valued logic and cannot be learned using a single MVN with activation function (2) with $k = 4$. However, this function can be learned by a single MVN with activation function (8) with $k = 4, l = 8, m = 32$ (see columns 5-6 of Table 2). The learning process converges after 584-43875 iterations (ten independent runs).

It is interesting that every time the learning process has converged to different weighting vectors, but to the same type of a resulting monotonic *m*-valued function (see the 5th column of the Table 2). This confirms that the learning of a non-threshold *k*-valued function may be reduced to the learning of a partially defined *m*-valued threshold function.

7 Conclusions

We have considered in this paper the original approach for projecting a *k*-valued non-linearly separable function of *n* variables to a partially defined *m*-valued threshold function of *n* variables. This makes it possible to learn such no-linearly separable multiple-valued functions using a single multi-valued neuron like it is possible to learn non-linearly separable Boolean functions using a single universal binary neuron. This means that non-linearly separable classification problems, which have been considered unsolvable using a single neuron, can now be learned using a single multi-valued neuron. The projection of a non-linearly separable problem to a linearly separable problem described in the paper is reached by using a new periodic activation function for the multi-valued neuron and by employing a modified learning algorithm for this neuron.

Acknowledgements. This work is supported by the National Science Foundation under Grant 0925080.

Table 2. Non-threshold function of 3 variables of 4-valued logic and the results of its learning

x_1	x_2	x_3	$f(x_1, x_2, x_3)$ 4-valued	$j \in M$, $M = \{0,1,...,31\}$	$P_l =$ $= j \bmod 4$
0	0	0	0	8	0
0	0	1	1	9	1
0	0	2	2	10	2
0	0	3	3	11	3
0	1	0	1	9	1
0	1	1	2	10	2
0	1	2	3	11	3
0	1	3	0	12	0
0	2	0	2	10	2
0	2	1	3	11	3
0	2	2	0	12	0
0	2	3	1	13	1
0	3	0	3	11	3
0	3	1	0	12	0
0	3	2	1	13	1
0	3	3	2	14	2
1	0	0	1	9	1
1	0	1	2	10	2
1	0	2	3	11	3
1	0	3	0	12	0
1	1	0	2	10	2
1	1	1	3	11	3
1	1	2	0	12	0
1	1	3	1	13	1
1	2	0	3	11	3
1	2	1	0	12	0
1	2	2	1	13	1
1	2	3	2	14	2
1	3	0	0	12	0
1	3	1	1	13	1
1	3	2	2	14	2
1	3	3	3	15	3
2	0	0	2	10	2
2	0	1	3	11	3
2	0	2	0	12	0
2	0	3	1	13	1
2	1	0	3	11	3
2	1	1	0	12	0
2	1	2	1	13	1
2	1	3	2	14	2
2	2	0	0	12	0
2	2	1	1	13	1
2	2	2	2	14	2
2	2	3	3	15	3
2	3	0	1	13	1
2	3	1	2	14	2
2	3	2	3	15	3
2	3	3	0	16	0
3	0	0	3	11	3
3	0	1	0	12	0
3	0	2	1	13	1
3	0	3	2	14	2
3	1	0	0	12	0
3	1	1	1	13	1
3	1	2	2	14	2
3	1	3	3	15	3
3	2	0	1	13	1
3	2	1	2	14	2
3	2	2	3	15	3
3	2	3	0	16	0
3	3	0	2	14	2
3	3	1	3	15	3
3	3	2	0	16	0
3	3	3	1	17	1

References

1. Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Prentice Hall, New Jersey (2009)
2. Aizenberg, I., Aizenberg, N., Vandewalle, J.: Multi-valued and universal binary neurons: theory, learning, applications. Kluwer Academic Publishers, Boston (2000)
3. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
4. Aizenberg, N.N., Aizenberg, I.N.: CNN Based on Multi-Valued Neuron as a Model of Associative Memory for Gray-Scale Images. In: The Second IEEE Int. Workshop on Cellular Neural Networks and their Applications, pp. 36–41. Technical University Munich, Germany (1992)
5. Aizenberg, N.N., Ivaskiv, Y.L., Pospelov, D.A.: About one generalization of the threshold function Doklady Akademii Nauk SSSR (The Reports of the Academy of Sciences of the USSR), 196(6) 1287–1290 (1971) (in Russian)
6. Aizenberg, N.N., Ivaskiv, Y.L.: Multiple-Valued Threshold Logic. Naukova Dumka Publisher House, Kiev (1977) (in Russian)
7. Aizenberg, I., Aizenberg, N., Vandewalle, J.: Multi-valued and universal binary neurons: theory, learning, applications. Kluwer Academic Publishers, Boston (2000)
8. Aizenberg, I., Moraga, C., Paliy, D.: A Feedforward Neural Network based on Multi-Valued Neurons. In: Reusch, B. (ed.) Computational Intelligence, Theory and Applications. Advances in Soft Computing. AISC, vol. XIV, pp. 599–612. Springer, Heidelberg (2005)
9. Aizenberg, I., Moraga, C.: Multilayer Feedforward Neural Network Based on Multi-Valued Neurons (MLMVN) and a Backpropagation Learning Algorithm. Soft Computing 11(2), 169–183 (2007)
10. Aizenberg, I.N.: A Universal Logic Element over the Complex Field. Kibernetika (Cybernetics and Systems Analysis) 27(3), 116–121 (in Russian) ; English version is available from Springer 27(3), 467–473 (1991)
11. Aizenberg, I.: Solving the XOR and Parity n Problems Using a Single Universal Binary Neuron. Soft Computing 12(3), 215–222 (2008)
12. Aizenberg, I.: A Multi-Valued Neuron with a Periodic Activation Function. In: International Joint Conference on Computational Intelligence, Funchal-Madeira, Portugal, October 5-7, pp. 347–354 (2009)
13. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. University of California, Irvine (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
14. Chen, J.-H., Chen, C.-S.: Fuzzy Kernel Perceptron. IEEE Transactions on Neural Networks 13, 1364–1373 (2002)