Learning of the Non-Threshold Functions of Multiple-Valued Logic by a Single Multi-Valued Neuron With a Periodic Activation Function

Igor Aizenberg
Department of Computer Science
Texas A&M University-Texarkana
Texarkana, USA
igor.aizenberg@tamut.edu

Abstract—In this paper, a theory of multiple-valued threshold functions over the field of complex numbers is further developed. k-valued threshold functions over the field of complex numbers can be learned using a single multi-valued neuron (MVN). We propose a new approach for the projection of a k-valued function, which is not a threshold one, to m-valued logic (m > k), where this function becomes a partially defined m-valued threshold function and can be learned by a single MVN. To build this projection, a periodic activation function for the MVN is used. This new activation function and a modified learning algorithm make it possible to learn nonlinearly separable multiple-valued functions using a single MVN.

Keywords—multi-valued neuron; multiple-valued threshold function; learning; complex-valued weights

I. INTRODUCTION

In this paper, we further develop a theory of multiple-valued logic over the field of complex numbers and consider its new interesting application.

Multiple-valued threshold logic over the field of complex numbers is based on the following idea proposed in [1] and later presented in detail in [2] and [3]. Let M be an arbitrary additive group and its cardinality is not lower than k. Let \( A_k = \{ a_0, a_1, ..., a_{k-1} \}, A_k \subset M \) be a structural alphabet.

Definition 1 [1-3]. Let us call a function \( f : A^n_k \rightarrow A_k \) of \( n \) variables (where \( A^n_k \) is the \( n \)th Cartesian power of \( A_k \)) a function of \( k \)-valued logic over group \( M \).

Let us take the field of complex of complex numbers as a group \( M \) and a set of \( k \)th roots of unity \( E_k = \{ e^0, e, e^2, ..., e^{k-1} \} \), where \( e = e^{2\pi i/k} \) (i is an imaginary unity) is a primitive \( k \)th root of unity (see Fig. 1) as a set \( A_k \). Hence, any function of \( n \) variables \( f : E^n_k \rightarrow E_k \) is a function of \( k \)-valued logic over the field of complex numbers according to Definition 1.

Let \( K = \{ 0, 1, ..., k-1 \} \) be a set of values of a regular \( k \)-valued logic. Evidently, it is very easy to build one-to-one correspondence between set \( K \) and a set of the \( k \)th roots of unity \( E_k = \{ e^0, e, e^2, ..., e^{k-1} \} \). Thus, the regular \( k \)-valued logic becomes the one over the field of complex numbers and any multiple-valued function of \( k \)-valued logic becomes a multiple-valued function over the field of complex numbers [3]. Evidently, the function values and its arguments in this case are the \( k \)th roots of unity: \( e^j = e^{2\pi j/k}, \ j = 0, ..., k-1 \), where \( j \) is an imaginary unity.

Let us consider the following function (or \( k \)-valued predicate) proposed in [1]:

\[
P(z) = \exp(2\pi j/k), \text{ if } 2\pi j/k \leq \arg z < 2\pi (j + 1)/k,
\]

where \( \arg z \) is the argument of the complex number \( z \). Function (1) divides a complex plane onto \( k \) equal sectors and maps the whole complex plane into a set of \( k \)th roots of unity (see Fig. 1).

Fig. 1. Geometrical interpretation of the discrete MVN activation function
Definition 2 [2, 3]. A \( k \)-valued function \( f(x_1, ..., x_n) \) over the field of complex numbers is called a \( k \)-valued threshold function (a threshold function of \( k \)-valued logic) over the field of complex numbers if there exist such a complex-valued weighting vector \( W = (w_0, w_1, ..., w_n) \) that the following condition holds for all \( x_1, ..., x_n \) from the domain of this function:

\[
 f(x_1, ..., x_n) = P(w_0 + w_1x_1 + ... + w_nx_n), \tag{2}
\]

where \( P \) is function (1).

A discrete multi-valued neuron (MVN) was introduced in [4]. This is a neuron with complex-valued weights, activation function (1) and inputs/output, which are \( k \)-th roots of unity. Thus, MVN may learn and implement those input/output mappings, which are multiple-valued threshold functions of \( k \)-valued logic. Two learning algorithms for the MVN are presented in [2] and [3], respectively. For any \( k \), a set of \( k \)-valued threshold functions is quite big and this makes the MVN more functional than other neurons. There are many applications where the MVN has shown its high efficiency. We can mention among others several associative memories with a different topology: the cellular memory [4], the Hopfield-like memories [5]-[8], the memories for storing medical images [9], [10], and the memory with random connections [3]. In [11], [12], it was suggested to use the MVN as a basic neuron in a feedforward neural network (a multilayer neural network with multi-valued neurons – MLMVN), which outperforms a traditional feedforward neural network in terms of functionality, learning speed, and classification/prediction rate when solving both benchmark and real world problems. For example, in [13], the MLMVN was used to learn the genetic code as a multiple-valued function. Hence, the MVN has confirmed its high efficiency in many applications, and multiple-valued threshold logic over the field of complex numbers has also confirmed its importance.

However, there is still a very interesting question: what we can do with those functions of \( k \)-valued logic, which are not the threshold ones? How it is possible to learn them? On the one hand, a single MVN may learn only threshold functions of \( k \)-valued logic and considering this problem in the straightforward way, we may conclude that to learn non-threshold functions, it is necessary to use networks, for example, the MLMVN. On the other hand, we may take a look at this problem from the different side. If some \( k \)-valued function is not a threshold function in \( k \)-valued logic, is it possible that it is a partially defined threshold function in \( m \)-valued logic, where \( m > k \)? In fact, a positive answer to this question is already given for \( k=2 \). It is shown in [3] and [14] that a Boolean function of \( n \) variables, which is not a threshold function, can be learned as a partially defined (only on the Boolean inputs) \( m \)-valued function, where \( m = 2l \), and \( l \) is a positive integer. Thus, a universal binary neuron (UBN), which can learn non-threshold Boolean functions, was suggested [15] and then developed [3]. A positive answer to this question for \( k=2 \) might give a very attractive opportunity to learn those \( k \)-valued functions, which are not threshold in \( k \)-valued logic, using a single MVN. In this case, we will be able to learn them not in \( k \)-valued logic, but in \( m \)-valued logic (\( m > n \) must be substituted for \( k \) in (1)). In [16] a periodic activation function was suggested for multi-valued neuron. It was also shown that this makes it possible to learn non-threshold multiple-valued functions using a single neuron.

In this paper, we further develop this approach. It will be shown that a \( k \)-valued function, which is not threshold in \( k \)-valued logic, may be projected into \( m \)-valued logic, where it becomes a partially defined threshold function. This makes it possible to learn this function using a single MVN. A mechanism of this projection is utilized through a periodic activation function and a modified learning algorithm developed for the MVN. We will also show how this approach can be used to learn using a single MVN those problems that were always considered non-linearly separable and not learnable using a single neuron.

II. \( L \)-REPETITIVE \( k \)-PERIODIC ACTIVATION FUNCTION FOR MULTI-VALUED NEURON

Let \( E_k = \{1, e_1, e_2, ..., e_{k^2}\} \) (where \( e_k = e^{2\pi i/k} \) 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 \( f : E^* \rightarrow K \) or \( f : O^* \rightarrow K \). Hence, the range of \( f \) is discrete, while its domain is either discrete or continuous. In general, its domain may be even hybrid. In fact, if some function \( f(y_1, ..., y_n), y_j \in [a_j, b_j] \), \( a_j, b_j \in \mathbb{R}, j=1, ..., n \) is defined on the bounded subdomain \( D^* \subset \mathbb{R}^n \) \( \{ f : D^* \rightarrow K \} \), then it can be easily transformed to \( f : O^* \rightarrow K \) by a simple linear transformation applied to each variable:

\[
y_j \in [a_j, b_j] \Rightarrow \quad \phi_j = \frac{y_j - a_j}{b_j - a_j} \quad 2\pi \in [0, 2\pi], j = 1, 2, ..., n, \tag{3}
\]

and then

\[
x_j = e^{i\phi_j} \in O, j = 1, 2, ..., n \quad \text{is the complex number located on the unit circle. Hence, we obtain the function}
\]

\[
f(x_1, ..., x_n) : O^* \rightarrow K.
\]

Suppose that a \( k \)-valued function \( f(x_1, ..., x_n) \) is not a threshold function. Let us project this \( k \)-valued function \( f(x_1, ..., x_n) \) into \( m \)-valued logic, where \( m = kl \), \( l \) is integer
and \( I \geq 2 \). Let us implement this projection using a periodic activation function, which was recently proposed for MVN in [16]:

\[
P(z) = j \mod k,
\]

if \( 2\pi j / m \leq \arg z < 2\pi (j + 1) / m, \)

\( j = 0, 1, \ldots, m - 1; m = kl, l \geq 2. \)

This definition is illustrated in Fig. 2. Activation function (3) separates the complex plane into \( m \) equal sectors and \( \forall t \in K \) there are exactly \( l \) sectors, where (3) equals to \( t \).

![Fig. 2. Geometrical interpretation of the \( k \)-periodic \( l \)-repeated activation function (3)](image)

This means that activation function (3) establishes mappings from \( E_k \) into \( E_m = \{ 1, e_m, e_m^2, \ldots, e_m^{k-1}, e_m^k, e_m^{k+1}, \ldots, e_m^{m-1} \} \) (where \( e_m \) is a primitive \( m \)th root of unity) and from \( K \) into \( M = \{ 0, 1, \ldots, k - 1, k + 1, \ldots, m - 1 \} \), respectively. Evidently, these mappings project a \( k \)-valued function \( f(x_1, \ldots, x_n) \) into \( m \)-valued logic. This projection can be very interesting from the practical point of view if \( f(x_1, \ldots, x_n) \), being a non-threshold function in \( k \)-valued logic, becomes a partially defined threshold function in \( m \)-valued logic. The latter means that this function can be learned by a single MVN with the activation function (3). 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 in which one of the \( m \) sectors (whose ordinal numbers are determined by the elements of the set \( M \)) the weighted sum is located, is equal to \( 0, 1, \ldots, k - 1, 0, 1, \ldots, k - 1, \ldots, 0, 1, \ldots, k - 1 \).

\[
(4)
\]

Thus, function (3) is \( k \)-periodic (its period is \( k \) and \( l \)-repetitive (each its value is repeated exactly \( l \) times). In \( m \)-valued logic over the field of complex numbers our function takes, respectively the values from the set \( E_m \):

\[
e_m^0 = 1, e_m^1, \ldots, e_m^{k-1}, e_m^k, e_m^{k+1}, \ldots, e_m^{2k-1}, \ldots, e_m^m, e_m^m, \ldots, e_m^{kl-1}, e_m^{kl+1}, e_m^{kl-1}
\]

It is important to mention that if \( l = 1 \) in (3) then \( m = k \) and activation function (3) coincides with activation function (1) accurate within the interpretation of the neuron’s output (if the weighted sum is located in the \( j \)-th sector then in (1) then the neuron’s output is equal to \( e_j^{j/2\pi/k} = e_j \in E_k \), which is the \( j \)-th of \( k \)-th root of unity, while in (3) it is equal to \( j \in K \). Hence (3) projects \( k \)-valued logic into \( m \)-valued logic and simultaneously it re-projects values of any \( k \)-valued function \( f(x_1, \ldots, x_n) \) back into \( k \)-valued logic. To consider our \( k \)-valued function in \( m \)-valued logic, without this re-projection, we should just substitute \( m \) for \( k \) in (1).

III. LEARNING RULES AND STRATEGIES

As we have seen, function (3) projecting a \( k \)-valued function \( f(x_1, \ldots, x_n) \) into \( m \)-valued logic, projects \( E_k \) into \( E_m \). However, function (3) itself does not establish any mechanism of this projection. This mechanism can be established by the MVN with the activation function (3) by learning the corresponding function \( f(x_1, \ldots, x_n) \) in \( m \)-valued logic. This means that the correspondence between sets \( E_k \) and \( E_m \) or in other words a way in which \( E_k \) is projected into \( E_m \) can be established by a learning algorithm in the self-adaptive way.

Let \( f(x_1, \ldots, x_n) \) be a function of \( k \)-valued logic and \( f : E_k^+ \rightarrow K \) or \( f : O^+ \rightarrow K \). Suppose this function cannot be learned by a single MVN with activation function (1). Let us try to learn it in \( m \)-valued logic using a single MVN with activation function (3). Thus, the expected result of this learning process is the representation of \( f(x_1, \ldots, x_n) \) according to (2), where activation function (3) substitutes for the activation function (1).

The learning process, which we consider here, can be based on the error-correction learning rule, which was proposed for the MVN in [3]:

\[
W_{r+1} = W_r + \frac{C_r}{(n+1)} (e^t - e^r) \tilde{X},
\]

or on the modified learning rule, which was suggested in [11]:

\[
W_{r+1} = W_r + \frac{C_r}{(n+1)} (e^t - e^r) \tilde{X},
\]
\[ W'_{r+1} = W_r + \frac{C_r}{(n+1)|z_r|} \left( \varepsilon^d - \varepsilon^t \right) \bar{X}, \]  

where \( \bar{X} \) is the input vector with the components complex-conjugated, \( n \) is the number of neuron inputs, \( \varepsilon^d \) is the desired output of the neuron, \( \varepsilon^t = P(z) \) is the actual output of the neuron, \( r \) 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 (6) is a variable part of the learning rate. The use of it can be important for learning highly nonlinear input/output mappings with a number of high irregular jumps. However, it should not be used for learning smooth, non-spiky functions, where rule (5) can be used.

The learning rule (5) requires that a desired neuron’s output is pre-determined. Unlike the case of the regular MVN with the activation function (1), a desired output in terms of \( m \)-valued logic cannot be determined unambiguously for the MVN with activation function (3) for \( l \geq 2 \). According to (3), there are exactly \( l \) sectors out of \( m \) on the complex plane, where this activation function equals to the given desired output \( t \in K \). Therefore, there are exactly \( l \) of \( m^l \) roots of unity that can be used as the desired outputs in learning rule (5). By using two self-adaptive learning strategies, the desired output may be determined during the learning process every time, when the neuron’s output is incorrect.

The first strategy is based on the same idea that the one for the UBN. Since the MVN with activation function (3) is a generalization of the UBN for \( k \geq 2 \), we suggest using here the same learning strategy that was used in the UBN error-correction learning algorithm [3]. Let \( l \geq 2 \) in (3). Activation function (3) determines the \( k \)-periodic and \( l \)-multiple sequence (4) with respect to sectors on the complex plane. Suppose that the current actual output is not equal to the desired one and the current weighted sum is located in the sector \( s \in M = \{0,1,...,m-1\} \). Since \( l \geq 2 \) in (3), there are \( l \) sectors on the complex plane, where the function (3) takes a 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 \( z \) in terms of the angular distance. Then either learning rule (5) or learning rule (6) can be applied.

The second strategy is based on the following considerations. Activation function (3) divides the complex plane into \( l \) domains, and each of them consists of \( k \) sectors (Fig. 2). Since a function \( f \) to be learned as a partially defined function of \( m \)-valued logic ( \( m = lk \) ) is in fact a \( k \)-valued function, then each of \( l \) domains contains those \( k \) values, which may be used as the desired MVN outputs. Suppose that the actual output is not correct and the current weighted sum is located in the sector \( s \in M = \{0,1,...,m-1\} \). This sector in turn is located in the \( n \)th domain (of \( l, l = \lfloor s/k \rfloor \)). Since there are \( l \) domains and each of them contains exactly one correct output, we have \( l \) options to choose the desired output. Let us choose it in the same \( n \)th domain, where the current actual output is located. Hence \( q = tk + f_k(x_1,...,x_n) \), where \( f_k(x_1,...,x_n) \) is a desired value of function to be learned in terms of traditional multiple-valued logic

\[ f_k(x_1,...,x_n) \in K = \{0,1,...,k-1\} \]  

and respectively,

\[ f(x_1,...,x_n) \in E_q = \{1,e_k,e_k^2,...,e_k^{k-1}\} \]

Once \( q \) is determined, this means that \( \varepsilon^d \) be the desired output and either learning rule (5) or learning rule (6) can be applied.

The convergence of the learning algorithm based on either rule (5) or rule (6) and on either of the described learning strategies is proven in the following terms. If a \( k \)-valued function \( f(x_1,...,x_n) \) is a partially defined \( m \)-valued threshold function ( \( k < m \) ) and therefore it can be learned by a single MVN with activation function (3), then a weighting vector can be found after a finite number of learning steps determined by either rule (5) or rule (6) and either of two described strategies. This proof cannot be presented here just because the lack of space (we are planning to present it in a journal paper, where all results presented here will be generalized and developed). However, an idea behind this proof is the following. Applying either of learning rules (5) or (6), we adjust the weights every time when the actual neuron’s output does not coincide with the desired neuron’s output. Since we are given a condition that our function can be learned, this means that some weighting vector for this function exists. If we assume that the process of the weights adjustment is infinite, we face a contradiction of the fact that some weighting vector whose norm is finite exists, with the fact that this finite norm has to be greater than infinity. This means that the process of weights adjustment cannot be infinite and it always converges to the weighting vector.

IV. SIMULATION RESULTS

To confirm the efficiency of the proposed approach to learning of non-threshold \( k \)-valued multiple-valued function as a partially defined \( m \)-valued threshold function by a single multi-valued neuron, we used a software simulator written in Borland Delphi 5.0 running on a PC with the Intel® CoreTM2 Duo CPU. The following problems have been considered.

A. mod k Addition of n k-Valued Variables

This problem may be considered as a generalization of the famous and popular Parity \( n \) problem for the \( k \)-valued case. In fact, Parity \( n \) problem is a \( \text{mod} 2 \) addition of \( n \) variables. It is well known that this is a classical non-linearly separable problem. However, it may be learned by a single UBN [14].
As it was mentioned in [2], \( \text{mod} \, k \) addition of \( n \) variables is a non-threshold multiple-valued function for any \( k \) and any \( n \). Therefore, it cannot be learned by a single MVN with activation function (1). To our best knowledge, there is no evidence that this function may be learned by any other single neuron. However, as we will see now, it is not a problem to learn this function using a single MVN with activation function (3).

A universal solution of \( \text{mod} \, k \) addition problem in terms of the relationship between \( k, n \) on the one side and \( l \) in (3) on the other side, is beyond the scope of this paper. We just want to show here that this multiple-valued problem is really solvable at least for those \( k \) and \( n \), for which we have performed experimental testing. The experimental results are summarized in Table 1. Since the first learning strategy showed better performance for this problem (fewer learning iterations and time), all results are given only for this strategy. In all experiments we have chosen the smallest \( l \) in (3), for which the convergence was reached after a reasonable number of iterations (200,000 or less).

![Table 1: Simulation results for mod k addition of n k-valued variables](image)

<table>
<thead>
<tr>
<th>( k ) (value of ( k )-valued logic) in (3)</th>
<th>( n ) (number of variables/inputs)</th>
<th>( l ) (periodicity coefficient) in (3)</th>
<th>Rule (5)</th>
<th>Rule (6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>21</td>
<td>140</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>9700</td>
<td>9700</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>665</td>
<td>41</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>11102</td>
<td>648</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>10</td>
<td>80895</td>
<td>184236</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>13</td>
<td>17027</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td>191105</td>
<td>182015</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>12</td>
<td>58552</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>17</td>
<td>1525</td>
<td>4242</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>4</td>
<td>929</td>
<td>315</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>12</td>
<td>587</td>
<td>751</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>5</td>
<td>7895</td>
<td>5315</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>11</td>
<td>8311</td>
<td>8279</td>
</tr>
</tbody>
</table>

* - here and further "-" means that the convergence has not been reached after 200,000 iterations

B. Two Spirals

The two spirals problem is a well known non-linearly separable classification problem, where the two spirals points must be classified as belonging to the 1st or to the 2nd spiral. Thus, this is 2-dimensional, 2-class classification problem. The standard two spirals data set usually consists of 194 points (97 belong to the 1st spiral and other 97 points belong to the 2nd spiral). The following results are known as the best for this problem so far. The two spirals problem can be learned completely with no errors by the MLMVN [11] containing 30 hidden neurons on the single hidden layer and a single output neuron. This learning process requires about 800,000 iterations. The best known result for the standard backpropagation network with the same topology is 14% errors after 150,000 learning iterations [17]. For the cross-validation testing, where each second point of each spiral goes to the learning set and each other second point goes to the testing set, one of the best known results is reported in [18], where the classification accuracy up to 94.2% is shown there by the technique, which employs along with a traditional support vector machine (SVM) the merits of the kNN classifier.

A single MVN with activation function (3) \((l = 2, k = 2, m = 4)\) significantly outperforms all mentioned techniques. Just 2-3 learning iterations are required to learn the two spirals problem completely with no errors using the 1st learning strategy and learning rule (5). Just 3-6 iterations are required to achieve the same result using the 2nd learning strategy and the learning rule (6). These results are based on the ten independent runs of the learning algorithm. We also used ten independent runs to check the classification ability of a single MVN with activation function (3) \((l = 2, k = 2, m = 4)\) using the cross-validation. The two spirals data were divided into the learning set (98 samples) and testing set (96 samples). We reached the absolute success in this testing: 100% classification rate is achieved in all experiments.

C. Iris

This famous benchmark database was downloaded from the UC Irvine Machine Learning Repository [19]. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. Four real-valued (continuous) features are used to describe the data instances. It is known [19] that the first class is linearly separable from the other two but the latter are not linearly separable from each other. Thus, a regular single MVN with the activation function (1), as well as any other single artificial neuron cannot learn this problem completely. However, a single MVN with activation function (3) \((l = 3, k = 3, m = 9)\) learns the Iris problem completely with no errors. To transform the input
features into the numbers located on the unit circle, we have linearly transformed real numbers to the angle values, which determine points on the unit circle. The learning algorithm based on the second learning strategy and the learning rule (6), converges with the zero error. The convergence requires about $10^7$ iterations. Every time the error decreases very quickly and after 50-100 iterations there are stably 1 or just a few more samples, which still require the weights adjustment, but their final adjustment takes time (5-12 hours). Nevertheless, this result is very interesting, because to our best knowledge this is the first time when the “Iris” problem was learned using just a single neuron. It is interesting that after the convergence for the first class (known and referred to as “Iris Setosa” [19]) the weighted sums for all instances appear in the same single sector on the complex plane, for the second class “Iris Versicolour” sums for all instances appear in the same single sector on the complex plane, for the second class “Iris Setosa” [19] the weighted sums for all instances appear in the one sector, but about 1/3 appear in another one, located in the different “l-domain” (see (3) and Fig. 2). For the third class the weighted sums for all the instances excepting one appear in the same single sector on the complex plane, but for the one instance (every time the same) it appears in the different sector belonging to the different “l-domain”. In this way, the second and the third class, which initially are known as non-linearly separated, become separated.

V. CONCLUSIONS

In this paper, we further developed a theory of multiple-valued threshold functions over the field of complex numbers. $k$-valued threshold functions over the field of complex numbers can be learned using a single multi-valued neuron, while non-threshold functions cannot be learned. We propose a new approach for the projection of a $k$-valued function, which is not a threshold one, to $m$-valued logic ($m>k$), where this function becomes a partially defined threshold function and can be learned using a single MVN. To build this projection, a periodic activation function for the MVN is used. Then the projection to higher valued logic is utilized using a learning algorithm. The proposed approach makes it possible to learn non-linearly separable multiple-valued functions using a single MVN.

REFERENCES


