Abstract—Discrete Multi-valued Neuron (MVN) was proposed for solving classification problems. The neuron has an activation function which is used to create an output value for an input instance. The learning algorithm associated with discrete MVN was designed for multi-class classification. However, the algorithm may have difficulties in convergence for the cases of two-class classification. In this paper, we propose a revised activation function to overcome this difficulty. A concept of tolerating areas is included. Another scheme adopting new targets is also proposed to work with discrete MVN. Simulation results show that the proposed ideas can improve the performance of discrete MVN.

Index Terms—Classification, complex-valued neuron, discrete MVN, activation function.

I. INTRODUCTION

Classification is an important and interesting topic in the field of artificial intelligence. Many approaches for doing classification have been proposed, e.g., Neural Networks [1], [2] and k-nearest neighbor [3]. Aizenberg and Aizenberg proposed the discrete multi-valued neuron (MVN) as a model of associative memory for multi-class classification [4], [5]. Later, Aizenberg proposed another model [6], MVN-P, which incorporates periodic activation function and a modified learning algorithm, to improve the performance of MVN. Both MVN and MVN-P are based on a complex-valued neuron. Input instances are converted to complex values which are located on a unit circle, and the output value of the neuron is decided by the location of the weighted sum.

The learning algorithm associated with discrete MVN was designed for multi-class classification. However, the algorithm may have difficulties in convergence for the cases of two-class classification. In this paper, we propose a revised activation function to overcome this difficulty. Another scheme adopting new targets is also proposed to work with discrete MVN. We compare our proposed ideas with discrete MVN and MVN-P by running them on several benchmark datasets. The simulation results show that the proposed ideas can improve the performance of discrete MVN.

This paper is organized as follows. Section II introduces discrete MVN and MVN-P. Section III presents a revised activation function. Section IV introduces the concept of tolerating areas. Section V presents the idea of new targets. Simulation results for demonstrating the effectiveness of our proposed ideas are presented in Section VI. Finally, concluding remarks are given in Section VII.

II. RELATED WORK

Aizenberg and Aizenberg proposed the discrete multi-valued neuron (MVN) [4], [5], as shown in Figure 1, for multi-class classification. In this figure, \(x_0, x_1, \ldots, x_n\) are the inputs and \(f\) is the output of the neuron. The first input \(x_0\) is fixed to one, and \(x_1, \ldots, x_n\) are the complex values converted from the real feature values \(i_1, \ldots, i_n\) associated with an instance as follows:

\[
\phi_j = \frac{i_j - a_j}{b_j - a_j},
\]

\[
x_j = e^{i\phi_j},
\]

where \(b_j\) and \(a_j\) represent the maximum and minimum values, respectively, of feature \(j\), \(i\) is the imaginary unit, \(0 < \beta < 2\pi\), and \(j = 1, 2, \ldots, n\) with \(n\) being the number of features in the instance. The weights \(w_0, w_1, \ldots, w_n\) are also complex values, and \(z\) is the weighted sum written as

\[
z = w_0 + w_1 x_1 + \ldots + w_n x_n
\]

The output \(y\) of the neuron is

\[
y = P(z)
\]

\[
= \exp(2\pi j / k) \equiv e^j
\]

where \(P\) is the activation function, \(k\) is the number of categories we are concerned with, and \(j\) is an integer satisfying

\[
2\pi j / k \leq \arg(z) < 2\pi(j + 1) / k
\]

For convenience, the \(k\) categories are named as category 0, category 1, \ldots, and category \(k - 1\), respectively. Note that the activation function divides the complex plane into \(k\) sectors of equal size. The \(j\)th sector, \(S_j\), is defined by

\[
S_j = \{z | 2\pi j / k \leq \arg(z) < 2\pi(j + 1) / k\}
\]
for \( j = 0, 1, \ldots, k - 1 \). The resulting \( k \) sectors are shown in Figure 2. If an input instance belongs to category \( j \), \( j = 0, \ldots, k - 1 \), then the corresponding \( z \) is located in sector \( S_j \) and the neuron output is \( \varepsilon^j \). Conversely, if the neuron output observed for an input instance is \( \varepsilon^j \), then the corresponding \( z \) is located in sector \( S_j \) and the input instance is classified to category \( j \).

Aizenberg proposed MVN-P which incorporates periodic activation function and a modified learning algorithm [6]. MVN-P differs from MVN in the activation function used, making it more effective than MVN for multi-class classification. In MVN-P, \( k \) values are mapped to \( m \) values by requiring \( m = kl \) and \( l \geq 2 \). Therefore, the complex plane is divided into \( m \) sectors of equal size. Each sector \( S_j \) is defined as

\[
S_j = \{ z | 2\pi j/m \leq \arg(z) < 2\pi (j+1)/m \} \tag{7}
\]

for \( 0 \leq j \leq m - 1 \). The \( m \)-value activation function for MVN-P is defined to be:

\[
P_l(z) = j \mod k \tag{8}
\]

where \( 2\pi j/m \leq \arg(z) < 2\pi (j+1)/m \), \( j = 0, 1, \ldots, m-1 \). Figure 3 gives an interpretation of the above definition for \( m \) sectors. Note that the output of MVN-P is \( j \mod k \), instead of \( \varepsilon^j \). Therefore, the outputs of MVN-P are both \( l \)-periodic and \( k \)-multiple. If an instance belongs to category \( j \), its corresponding \( z \) value can be located in sector \( S_j \), sector \( S_{k+j} \), \ldots, or sector \( S_{k(l-1)+j} \). On the contrary, if the neuron output observed for an input instance is \( j \), then the corresponding \( z \) is located in sector \( S_j \), sector \( S_{k+j} \), \ldots, or sector \( S_{k(l-1)+j} \), and the input instance is decided to belong to category \( j \).

Given the architecture of MVN-P described above, the learning algorithm for the weights \( w_0, w_1, \ldots \) and \( w_n \) works as follows. For a training instance, if the output value is identical to the target of the instance, weights are not changed. However, if the output value is different from the target of the instance, weights have to be changed. Let \( S_h \) be the sector in which the \( z \) value of the instance is located, and \( S_q \) be the target sector nearest to \( z \) [6]. The weight vector \( W = [w_0, w_1, \ldots, w_n] \) is updated by

\[
W_{\text{new}} = W_{\text{old}} + C/(n+1)\delta\bar{X} \tag{9}
\]

\[
\delta = \varepsilon^q - \varepsilon^s \tag{10}
\]

where \( \bar{X} \) is the conjugate of the input vector \( X = [1, x_1, \ldots, x_n] \) and \( C \) is the learning rate which usually equals one.

III. REVISED ACTIVATION FUNCTION

Firstly, we describe why discrete MVN may have difficulties in convergence in binary classification. In two-class classification, the complex plane is divided into two sectors, as shown in Figure 4. Notice that the horizontal axis is the boundary of the two sectors. The relationship between the old \( z \) and the new \( \tilde{z} \), by referring to Eq.(9), can be described
By applying Eq.(12), the complex plane is divided into two sectors labeled by \(+i\) and \(-i\), respectively. If the weighted sum of an instance is located at an angle between 0 and \(\pi\), the output value will be a positive real number. In this case, the instance is labeled as \(+i\). If the weighted sum of an instance is located at an angle between \(\pi\) and 2\(\pi\), the output value will be a negative real number. In this case, the instance is labeled as \(-i\). Eq.(12) can be rewritten as follows:

\[
\tilde{\mathbf{z}} = \tilde{w}_0 + \tilde{w}_1 x_1 + \ldots + \tilde{w}_n x_n \\
= (w_0 + 1/(n+1)\delta) + (w_1 + 1/(n+1)\delta x_1) x_1 \\
+ \ldots + (w_n + 1/(n+1)\delta x_n) x_n \\
= w_0 + w_1 x_1 + \ldots + w_n x_n \\
+ 1/(n+1)\delta + \ldots + 1/(n+1)\delta \\
= z + (n+1)/(n+1)\delta \\
= z + \delta.
\]

Suppose two instances are put into discrete MVN, and their weighted sums are \(z_1\) and \(z_2\), respectively. Assume that these two instances are wrongly classified, so the weights must be updated by Eq.(9). Note that \(\delta = \varepsilon^1 - \varepsilon^0\) and \(\delta = \varepsilon^0 - \varepsilon^1\), respectively, with the directions being rightward or leftward. Therefore, their weighted sums after update can never be located in the correct sectors. As a result, the learning algorithm may have difficulties in convergence in the cases of binary classification.

We propose a revised model, called MVN-Sin, with the following activation function:

\[
P(z) = \sin(\arg(z))
\]

which is shown pictorially in Figure 5. Note that the function value of an angle between 0 and \(\pi\) is positive, while the function value of an angle between \(\pi\) and 2\(\pi\) is negative. By applying Eq.(12), the complex plane is divided into two

![Figure 5. The revised activation function.](image)

weights will not be updated. Note that 0 and \(\pi\) are located on the boundary of the sectors.

With the new activation function, we have

\[
\begin{cases}
\varepsilon^i = +i, \varepsilon^q = -i, & \text{if } \sin(\arg(z)) > 0 \\
\varepsilon^i = -i, \varepsilon^q = +i, & \text{if } \sin(\arg(z)) < 0 \\
o - update, & \text{if } \sin(\arg(z)) = 0
\end{cases}
\]

The direction of \(\delta\) in discrete MVN can only be rightward or leftward, unable to move updated weighted sums to correct sectors. However, by applying the new activation function, the directions of \(\delta\) are different. The direction of \(\delta\) are upward or downward, being able to move updated weighted sums to correct sectors, as shown in Figure 6. Let \(z_1\) and \(z_2\) be two instances that are wrongly classified, and have \(\delta = (-i) - (+i)\) and \(\delta = (+i) - (-i)\), respectively. After the application of Eq.(9), the updated weighted sums can be moved to correct sectors, respectively. Obviously, MVN-Sin can converge for the cases in binary classification. In summary, the learning algorithm of MVN-Sin can be described as below.

**procedure** MVN-Sin

Randomly generate initial values for the weights;

learning = false;

while(learning == false) {

for(i = 1; i \(\leq N; i + +\))

Compute the weighted sum \(z\) by Eq.(3);

\(x = P(z)\) by Eq.(12);

if(\(x \neq \) target)

Update the weights by Eq.(9);

learning = true;

endfor;

if(learning == false)

break;

else

learning = false;

endwhile;

end procedure

In the for loop, the instances are considered one by one from instance 1 to instance \(N\). As usual, the completion of one
for loop is called an epoch. The learning process continues until all the training instances are correctly classified.

The learning algorithm of MVN-Sin is easier than MVN-P. MVN-P must decide where the target is since each category is associated with 1 sectors and the total number of sectors is \( m = kl \). The sector which is nearest to the weighted sum \( z \) is chosen as the target. In contrast, the target for update in MVN-Sin is an easy choice, i.e., the opposite sector. Therefore, MVN-Sin can be faster than MVN-P in the training phase.

Another revised version, which we call MVN-Cos, adopts the following activation function:

\[
P(z) = \cos(\arg(z))
\]  

(13)

The sectors of MVN-Cos are rotated 90 degrees from those of MVN-Sin. The boundary of the two sectors is the longitudinal axis. The activation function can be rewritten as

\[
P(z) = \begin{cases} 
+1, & \text{if } \cos(\arg(z)) > 0 \\
-1, & \text{if } \cos(\arg(z)) < 0
\end{cases}
\]

and we have

\[
\varepsilon^q = +1, \varepsilon^s = -1, \text{ if } \cos(\arg(z)) > 0 \\
\varepsilon^q = -1, \varepsilon^s = +1, \text{ if } \cos(\arg(z)) < 0 \\
\text{no-update, if } \cos(\arg(z)) = 0
\]

The directions of \( \delta \) are shown in Figure 7. As matter of fact, MVN-Cos behaves in the same way as MVN-Sine. Therefore, we won’t consider it any more in the rest of the paper.

IV. INTRODUCING TOLERATING AREAS

The boundaries between two distinct categories in MVN or MVN-P are rigidly specified, resulting in inflexibility and long training time. In this section, we introduce tolerating areas for MVN-Sin, resulting in a version called MVN-Sin-T. For this version, two tolerating areas are involved at the two ends of each sector. The size of a tolerating area is

\[
\Delta \theta = \pi/\alpha
\]

where \( \alpha \) is a positive integer. We set \( \alpha \) to be 20 in this paper. Figure 8 shows an example of MVN-Sin-T. Note that parts B and C are tolerating areas. Tolerating areas can offer several advantages, including shorter training times and higher testing accuracies.

V. CHANGING TARGETS FOR MVN-P

In this section, we consider a modified learning scheme to MVN-P for multi-class classification. In MVN-P, the target \( q \) of \( \delta \) is the boundary of a desired sector. Now, in our modified version called MVN-P-Ct, we change the target to be the center of a desired sector instead as follows:

\[
\delta = \varepsilon^q \cdot \varepsilon^{1/2} - \varepsilon^s
\]  

(14)

The learning algorithm of MVN-P-Ct is identical to that presented Section III, except with the following change of \( \delta \) in Eq.(9):

\[
\delta = \exp(i2\pi q/m) \cdot \exp(i2\pi/2m) - \exp(i2\pi s/m)
\]  

(15)

Figure 9 shows an example of MVN-P-Ct with \( m = kl = 2 \times 2 = 4 \). In this figure, an instance with the weighted sum \( z_1 \) is wrongly classified and its target is sector \( j = 1 \). The thick line at \( \varepsilon^q \cdot \varepsilon^{1/2} \) indicates the center of sector \( j = 1 \) and it becomes the target for \( \delta \). The lines marked with \( \delta \) and \( \delta' \)
We stop the training of MVN-P(2) and MVN-P experiences difficulty in convergence. For Australian spent by each method. This table shows that as respectively.

MVN-Sin get 97.01% in accuracy for the Cancer dataset, while MVN-P(2) gets 96.15% and MVN-P(3) gets 94.93%. However, MVN-Sin-T only spends 0.2264 seconds, showing three to eight times faster than MVN-P(2) and MVN-P(3). However, MVN-Sin-T only spends 0.2264 seconds, showing four times faster than MVN-Sin.

Table IV compares testing accuracies between MVN-P and MVN-P-Ct. In this table, MVN-P-Ct(2) means MVN-P-Ct with \( l = 2 \) and MVN-P-Ct(3) means MVN-P-Ct with \( l = 3 \). From this table, it seems that MVN-P-Ct and MVN-P provide equal testing accuracies. Table V compares training times between MVN-P and MVN-P-Ct. As can be seen, MVN-P-Ct and MVN-P run with about the same speed. The exception is for Parkinsons. MVN-P-Ct(3) takes 3.9569 seconds, while MVN-P(3) takes 8.4778 seconds.

Table V compares training times between MVN-Sin and MVN-P for two-class data sets. Note that in this table, MVN-P-Ct(2) means MVN-P-Ct with \( l = 2 \) and MVN-P-Ct(3) means MVN-P-Ct with \( l = 3 \). From this table, it seems that MVN-P-Ct and MVN-P provide equal testing accuracies.

Table IV. Comparison between MVN-Sin and MVN-P on testing accuracy (%).

<table>
<thead>
<tr>
<th></th>
<th>MVN-Sin</th>
<th>MVN-P(2)</th>
<th>MVN-P(3)</th>
<th>MVN-Sin-T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancer</td>
<td>97.01</td>
<td>96.15</td>
<td>94.93</td>
<td>97.01</td>
</tr>
<tr>
<td>Heart</td>
<td>77.57</td>
<td>76.04</td>
<td>74.56</td>
<td>81.63</td>
</tr>
<tr>
<td>Sonar</td>
<td>89.47</td>
<td>88.52</td>
<td>77.89</td>
<td>88.49</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>87.15</td>
<td>88.75</td>
<td>84.70</td>
<td>88.91</td>
</tr>
<tr>
<td>Australian</td>
<td>82.90</td>
<td>82.30</td>
<td>82.60</td>
<td>83.34</td>
</tr>
<tr>
<td>Heart(2)</td>
<td>78.89</td>
<td>77.04</td>
<td>77.04</td>
<td>78.15</td>
</tr>
</tbody>
</table>

Table I. Characteristics of eight benchmark sets.

<table>
<thead>
<tr>
<th></th>
<th># of classes</th>
<th># of features</th>
<th># of training</th>
<th># of testing</th>
<th># of patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancer</td>
<td>2</td>
<td>31</td>
<td>452</td>
<td>117</td>
<td>569</td>
</tr>
<tr>
<td>Heart</td>
<td>2</td>
<td>44</td>
<td>212</td>
<td>55</td>
<td>267</td>
</tr>
<tr>
<td>Sonar</td>
<td>2</td>
<td>60</td>
<td>164</td>
<td>44</td>
<td>208</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>2</td>
<td>22</td>
<td>152</td>
<td>43</td>
<td>195</td>
</tr>
<tr>
<td>Australian</td>
<td>2</td>
<td>14</td>
<td>548</td>
<td>132</td>
<td>690</td>
</tr>
<tr>
<td>Heart(2)</td>
<td>2</td>
<td>13</td>
<td>216</td>
<td>54</td>
<td>270</td>
</tr>
<tr>
<td>Wine</td>
<td>3</td>
<td>13</td>
<td>136</td>
<td>42</td>
<td>178</td>
</tr>
<tr>
<td>Iris</td>
<td>3</td>
<td>4</td>
<td>120</td>
<td>30</td>
<td>150</td>
</tr>
</tbody>
</table>

are for Eq.(10) and Eq.(14), respectively, and the dotted lines represent the directions of update.

VI. SIMULATION RESULTS

In this section, we present some simulation results to demonstrate the effectiveness of our proposed ideas. All the algorithms involved in this section are implemented in MATLAB R2011b on a PC with the AMD Athlon(tm)II X2 CPU. Six two-class and two three-class data sets, Cancer, Heart, Sonar, Parkinsons, Australian, Heart(2), Wine and Iris are taken from the established UCI repository [15] for simulation. The characteristics of these data sets are shown in Table I. For example, in the Cancer data set there are 2 classes and 31 features. The total number of patterns is 569 of which 452 are used for training and the rest are for testing.

A five-fold cross-validation is adopted for all the benchmark sets.

We do not compare discrete MVN since it never converges in binary cases. We compare with MVN-P which is more powerful than discrete MVN and is able to converge in two-class cases. For MVN-P, the number for \( l \) has to be chosen. The classification accuracies obtained from MVN-P and MVN-Sin are shown in Table II for two-class data sets. Note that in this Table, MVN-P(2) means MVN-P with \( l = 2 \) and MVN-P(3) means MVN-P with \( l = 3 \). For easy investigation, Table II is also shown in Figure 10. As can be seen, MVN-Sin and MVN-Sin-T run faster than MVN-P(2) and MVN-P(3) for most of the cases. For example, both MVN-Sin get 97.01% in accuracy for the Cancer dataset, while MVN-P(2) gets 96.15% and MVN-P(3) gets 94.93%, respectively.

Table III shows the comparison on the training time spent by each method. This table shows that as \( l \) increases, MVN-P experiences difficulty in convergence. For Australian and Heart(2), MVN-P(2) and MVP-P(3) cannot converge in 10,000 epoches. We stop the training of MVN-P(2) and MVP-P(3) after 10,000 epoches for these two datasets. As can be seen, MVN-Sin runs faster than MVN-P(2) and MVN-P(3). However, MVN-Sin-T can run even faster. For example, MVN-Sin spends 0.9734 seconds on the Parkinsons dataset, while MVN-P(2) spends 3.2785 seconds and MVN-P(3) spends 8.4778 seconds. This shows MVN-Sin is about three to eight times faster than MVN-P(2) and MVN-P(3). However, MVN-Sin-T only spends 0.2264 seconds, showing four times faster than MVN-Sin.

Table IV. Comparison between MVN-P and MVN-P-Ct on testing accuracy (%).

<table>
<thead>
<tr>
<th></th>
<th>MVN-P-Ct(2)</th>
<th>MVN-P-Ct(3)</th>
<th>MVN-P(2)</th>
<th>MVN-P(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancer</td>
<td>96.32</td>
<td>94.56</td>
<td>96.15</td>
<td>94.93</td>
</tr>
<tr>
<td>Heart</td>
<td>74.56</td>
<td>72.31</td>
<td>76.04</td>
<td>74.56</td>
</tr>
<tr>
<td>Sonar</td>
<td>82.29</td>
<td>75.49</td>
<td>88.52</td>
<td>87.89</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>90.83</td>
<td>86.03</td>
<td>88.73</td>
<td>84.70</td>
</tr>
<tr>
<td>Wine</td>
<td>94.34</td>
<td>94.93</td>
<td>93.28</td>
<td>93.17</td>
</tr>
<tr>
<td>Iris</td>
<td>96.00</td>
<td>96.00</td>
<td>96.00</td>
<td>96.00</td>
</tr>
</tbody>
</table>
Table V

<table>
<thead>
<tr>
<th></th>
<th>MVN-P-CT (2)</th>
<th>MVN-P-CT (3)</th>
<th>MVN-P (2)</th>
<th>MVN-P (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancer</td>
<td>0.7524</td>
<td>2.7402</td>
<td>0.7675</td>
<td>1.7274</td>
</tr>
<tr>
<td>Heart</td>
<td>1.8705</td>
<td>3.6093</td>
<td>1.7117</td>
<td>3.3008</td>
</tr>
<tr>
<td>Sonar</td>
<td>0.1412</td>
<td>0.3072</td>
<td>0.1573</td>
<td>0.2993</td>
</tr>
<tr>
<td>Parkinsons</td>
<td>2.1966</td>
<td>3.9569</td>
<td>3.2785</td>
<td>8.4778</td>
</tr>
<tr>
<td>Wine</td>
<td>0.1075</td>
<td>0.2354</td>
<td>0.1441</td>
<td>0.1955</td>
</tr>
<tr>
<td>Iris</td>
<td>5.7979</td>
<td>6.2607</td>
<td>6.1591</td>
<td>6.9697</td>
</tr>
</tbody>
</table>

VII. Conclusion

We have presented the architecture of the discrete Multi-valued Neuron with Sine activation function (MVN-Sin) which is an improved version of the discrete multi-valued Neuron (MVN) proposed by Aizenberg [6] for two-class problems. In MVN-Sin, the revised activation function is provided to change the direction of update and the location of the resulting sectors. We have presented MVN-Sin-T, having a tolerating zone provided between two distinct categories. We have also presented MVN-P-Ct which is MVN-P with targets modified to change the direction of $\delta$. Simulation results have shown that MVN-Sin can run faster than MVN-P and discrete MVN, MVN-Sin-T can run faster than MVN-Sin, and MVN-P-Ct has similar characteristics as MVN-P.

References