Letters

An approach for construction of Boolean neural networks based on geometrical expansion

Di Wang*, Narendra S. Chaudhari
School of Computing Engineering, Nanyang Technological University, Singapore 639798, Singapore

Abstract

We propose a fast covering learning algorithm (FCLA) for construction of Boolean neural networks. We visualize a neuron in terms of a hypersphere. To expand this hypersphere, we introduce three different radii. The construction process makes use of three concentric hyperspheres based on these radii, and is illustrated using an example. FCLA results in a simple neural network; further the resulting network structure is less sensitive to the order in which the input is given.

© 2003 Elsevier B.V. All rights reserved.

Keywords: Binary neural networks; Hard limiter neuron; Geometric learning; Linear separability

1. Introduction

Many algorithms used in the construction of binary neural networks (BNNs) are known. They include Boolean-like training algorithm (BLTA) [1], expand-and-truncate learning (ETL) [2], improved ETL (IETL) [7], constructive set covering learning algorithm (CSCLA) [5,6] and multi-core learning (MCL) [3,4]. In general, ETL, IETL, CSCLA and MCL have no generalization capability. BLTA has generalization capability but needs more hidden neurons. ETL, IETL, MCL are core-based training algorithms. The network structures constructed by ETL, IETL and MCL vary a lot in terms of different input sequences.

Extending the concepts in these approaches, we propose a fast covering learning algorithm (FCLA) for BNNs. FCLA constructs a two-layered BNN using hard-limiter

* Corresponding author.
E-mail addresses: wangdi@pmail.ntu.edu.sg (D. Wang), asnarendra@ntu.edu.sg (N.S. Chaudhari).
neurons. In our approach, we decide the region (half-space) of a neuron by visualizing it in terms of its equivalent hypersphere.

2. Preliminaries

A set of $2^n$ binary patterns $(0,1)^n$ can be considered as an $n$-dimensional unit hypercube. Each pattern is located on one vertex of this hypercube, hence on the surface of the following reference hypersphere (RHP):

$$\left(x_1 - \frac{1}{2}\right)^2 + \left(x_2 - \frac{1}{2}\right)^2 + \cdots + \left(x_n - \frac{1}{2}\right)^2 = \frac{n}{4}.$$ (1)

A Boolean function (of $n$ variables) classifies $2^n$ vertices into two classes: a set of true vertices, and a set of false vertices. If a Boolean function is linearly separable, then all true vertices lie inside or on some hypersphere (HP), and all false vertices lie outside [2]

$$\sum_{i=1}^{n} (x_i - c_i)^2 = r^2,$$ (2)

where $(c_1,c_2,\ldots,c_n)$ is the center of HP and $r$ is the radius. The intersection of HP and RHP is the separating hyperplane [2]

$$\sum_{i=1}^{n} (2c_i - 1)x_i = \left\{\sum_{i=1}^{n} c_i^2\right\} - r^2.$$ (3)

The true vertices, lie on one side of this hyperplane (having left-hand side expression in (3) being $> =$ right-hand side).

3. Our approach: neuron visualized in terms of a hypersphere

A typical hyperplane in a BNN has its corresponding hypersphere, with center $c$, and radius $r$. While constructing a BNN, suppose that $\{x_1^1,x_2^1,\ldots,x_v^v\}$ are $v$ (true) vertices included in one hypersphere. In terms of these vertices, we define the center $c = (c_1,c_2,\ldots,c_n)$, as

$$c_i = \frac{\sum_{k=1}^{v} x_{ki}^i}{v}.$$ (4)

To restrict our discussion for integer valued weights, we multiply both sides of (3) by $v$,

$$\sum_{i=1}^{n} \left(\left\{2 \sum_{k=1}^{v} x_{ki}^i\right\} - v\right)x_i = v \left[\sum_{i=1}^{n} c_i^2\right] - r^2.$$ (5)

We represent (5) using a hard-limiter model for a neuron, which has output zero when its input is less than the threshold, and has output one for values greater than or equal to the threshold. If a Boolean function is not linearly separable, it can be
Fig. 1. The process of geometrical expansion.

represented by some $m$-dimensional hyperspheres (and hence, hyperplanes). Each hidden neuron in BNN represents one (of these $m$) hyperplane. Different approaches for construction of BNNs use different methods for determining these hyperplanes. We illustrate our approach below.

We consider three radii $r_1, r_2, r_3$ ($r_1 < r_2 < r_3$) with the same center $c$. The radius $r_1$ denotes the minimum value such that all the $v$ vertices (in “match region” in Fig. 1) are exactly in or on the hypersphere. A new vertex (say $x$) within the “claim region” in Fig. 1(a) (defined using $r_2$) causes an immediate expansion of the hypersphere, to get the new vertex on the surface of the hypersphere after expansion (Fig. 1(b)). A new vertex within the “boundary region” (defined using $r_3$) cannot be included in the hypersphere by immediate expansion, but is expected to be included in the next (or later) construction iteration. A new vertex (say $y$) (Fig. 1(b)) beyond $r_3$ is too far from this hypersphere. So it generates a new hypersphere (a new hidden neuron) (Fig. 1(c)).

These three radii are defined as follows:

\[
\begin{align*}
    r_1^2 &= \max_k \sum_{i=1}^n (x_i^k - c_i)^2, \\
    r_2^2 &= r_1^2 + \sigma_1^2, \\
    r_3^2 &= r_2^2 + \sigma_2^2,
\end{align*}
\]

where $\sigma_1$ and $\sigma_2$ are the parameters to capture “geometric expansion”. One method for their choice is shown in Fig. 2.

In Fig. 2, the claim region is defined by vertices with Hamming distance one from the match region. So we choose $\sigma_1 = 1$. Similarly, if we define the boundary region by vertices with Hamming distance one from the claim region, we choose $\sigma_2 = 1$. Other values of these parameters are possible, and are important for studying the behavior of our method; however, we do not consider this aspect here.

FCLA constructs a BNN whose structure is illustrated in Fig. 3. Since our method has a “generalization capability”, the expanded hyperspheres may include false vertices. Such a false vertex needs correction, which can be taken care of by modifying the
construction of the output layer (e.g., by using the approach in [2]). To illustrate the strength of our approach for “nicely clustered” example(s), these details are, however, not necessary.

4. Formulae for weights and a threshold value of a neuron

After visualizing a neuron (represented by an hyperplane) in terms of its equivalent hypersphere having the radius \( r_1 \), from (5), it follows that our operation of “expansion of hypersphere” by changing the radii to \( r_2 \), and \( r_3 \) get translated to change of thresholds for the corresponding neuron (the expansion of a hypersphere does not result in a change of the weights of the neuron). From (5), the (integer valued) weights for neuron are

\[
w_i = v(2c_i - 1) = 2 \sum_{k=1}^{v} x_i^k - v \quad (i = 1, \ldots, n)
\]

and, the threshold \( t_1 \) (corresponding to \( r_1 \), given in (5) above) is

\[
t_1 = v \left( \sum_{i=1}^{n} c_i^2 - r_1^2 \right) = v \min_{k=1} \left\{ \sum_{i=1}^{n} (v(2c_i x_i^k - (x_i^k)^2)) \right\},
\]
where $x_k^i$ is in $\{0, 1\}$, so $x_k^i = (x_k^i)^2$. Hence,

$$t_1 = \min_{k=1}^{v} \left\{ \sum_{i=1}^{n} (\nu(2c_i - 1)x_k^i) \right\} = \min_{k=1}^{v} \sum_{i=1}^{n} w_i x_k^i.$$  (10)

The threshold $t_2$, corresponding to $r_2$, is

$$t_2 = t_1 - v\sigma_1^2 = \min_{k=1}^{v} \sum_{i=1}^{n} w_i x_k^i - v\sigma_1^2.$$  (11)

And, the threshold $t_3$, corresponding to $r_3$, is

$$t_3 = t_2 - v\sigma_2^2 = \min_{k=1}^{v} \sum_{i=1}^{n} w_i x_k^i - v\sigma_1^2 - v\sigma_2^2.$$  (12)

5. An approach for construction of a network

We construct a neural network having two layers, a hidden layer, and an output layer (Fig. 3). To do this, we examine whether a given “true” vertex, say $x^k$, can be covered by one of the hidden neurons. A hidden neuron has a weight vector, say $w$, and three thresholds ($t_1, t_2, t_3$). If $w \cdot x^k \geq t_1$, this true vertex is already covered by the neuron, so nothing needs to be done. If $t_2 \leq w \cdot x^k < t_1$, the vertex $x^k$ is within the “claim region” of the neuron; so we update the neuron parameters (i.e., its weight vector, and threshold values) to include $x^k$ in the neuron using (9)–(12). A new neuron with center as $x^k$ is added if $t_3 \leq w \cdot x^k < t_2$, the vertex $x^k$ is within the “boundary region” of the neuron; hence, we first examine whether other available neurons can “claim” it. If it cannot be included in any other available neuron, we “put aside” for re-consideration after other vertices are processed. Inclusion of other vertices to existing neurons results in expansion of “match” and “claim” regions; the vertices, put aside, may be claimed by such “expanded” neuron(s) later.

However, if the vertices “put aside” get trapped only within the boundary regions of existing neurons, we update $t_3$ to $t_2$ (corresponding to “shrinking” of radius $r_3$ to $r_2$).

The output layer has only one neuron, which collects all hidden neuron outputs with weight 1, and has threshold value of 1. (As stated in Section 3, in general, we also need to assign negative valued weights for taking care of false vertices that have become included in earlier neurons; however, we do not consider this aspect here.) The construction process ensures that, the values of connection weights for all neurons are integers.

6. An example

To illustrate the construction process in the above section, we take an example of the circular region representation using 6-bit space quantization, the well-known problem considered in [1–4, 7]. We set $\sigma_1 = \sigma_2 = 1$. For this problem, we illustrate the construction process in Table 1 below.
Table 1
The FCLA construction process of circular region (represented using 6-bit space quantization)

<table>
<thead>
<tr>
<th>Input</th>
<th>Neuron</th>
<th>Center</th>
<th>( r_1 )</th>
<th>( r_2 )</th>
<th>( r_3 )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
<th>( w_4 )</th>
<th>( w_5 )</th>
<th>( w_6 )</th>
<th>( t_1 )</th>
<th>( t_2 )</th>
<th>( t_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,100</td>
<td>1</td>
<td>(1,0,0,1,0,0)</td>
<td>0</td>
<td>1</td>
<td>( \sqrt{2} )</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>011100</td>
<td>2</td>
<td>(0,1,1,1,0,0)</td>
<td>0</td>
<td>1</td>
<td>( \sqrt{2} )</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>010011</td>
<td>3</td>
<td>(0,1,0,0,1,1)</td>
<td>0</td>
<td>1</td>
<td>( \sqrt{2} )</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>011010</td>
<td>4</td>
<td>Put aside</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100,010</td>
<td>5</td>
<td>Put aside</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101,011</td>
<td>4</td>
<td>(1,0,1,0,1,1)</td>
<td>0</td>
<td>1</td>
<td>( \sqrt{2} )</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>100,111</td>
<td>4</td>
<td>(1,0,1,2,0,1,1)</td>
<td>1/2</td>
<td>( \sqrt{3}/2 )</td>
<td>3/2</td>
<td>2</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>011101</td>
<td>2</td>
<td>(0,1,1,1,0,1,2)</td>
<td>1/2</td>
<td>( \sqrt{3}/2 )</td>
<td>3/2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>010100</td>
<td>2</td>
<td>(0,1,2,3,1,0,1/3)</td>
<td>( \sqrt{3}/3 )</td>
<td>( \sqrt{14}/3 )</td>
<td>( \sqrt{23}/3 )</td>
<td>-3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>-3</td>
<td>-1</td>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>101,100</td>
<td>1</td>
<td>(1,0,1,2,1,0,0)</td>
<td>1/2</td>
<td>( \sqrt{3}/2 )</td>
<td>3/2</td>
<td>2</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>100,101</td>
<td>1</td>
<td>(1,0,1,3,1,0,1/3)</td>
<td>( \sqrt{3}/3 )</td>
<td>( \sqrt{14}/3 )</td>
<td>( \sqrt{23}/3 )</td>
<td>3</td>
<td>-3</td>
<td>1</td>
<td>3</td>
<td>-3</td>
<td>-1</td>
<td>5</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>011011</td>
<td>3</td>
<td>(0,1,1,2,0,1,1)</td>
<td>1/2</td>
<td>( \sqrt{3}/2 )</td>
<td>3/2</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>011010</td>
<td>3</td>
<td>(0,1,2,3,0,1,2/3)</td>
<td>( \sqrt{3}/3 )</td>
<td>( \sqrt{14}/3 )</td>
<td>( \sqrt{23}/3 )</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>-3</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>100,010</td>
<td>4</td>
<td>(1,0,1,3,0,1,2/3)</td>
<td>( \sqrt{3}/3 )</td>
<td>( \sqrt{14}/3 )</td>
<td>( \sqrt{23}/3 )</td>
<td>3</td>
<td>-3</td>
<td>-1</td>
<td>-3</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

The first column gives the input vertices. The second column represents the index of the corresponding hidden neuron. The third column is the center (obtained using (4)) of the corresponding separating hypersphere(s), and columns 4–6 are the three radii obtained using (6)–(8) respectively. The connection weights \( w_i \) \( (i=1,\ldots,6) \) in Table 1 are obtained using (9). The three thresholds in Table 1 are obtained using (10)–(12).

The first twelve rows in Table 1 correspond to 12 true vertices (say, \( x_1,\ldots,x_{12} \)) in our Boolean function (first column in these rows specify these vertices). The vertex \( x_1 = 100,010 \) causes creation of neuron 1, with weights (obtained using (9)) \( w_1 = (1,-1,-1,1,-1,-1) \) and thresholds (obtained using (10)–(12)) given in the first row. For the vertex \( x_2 = 011100 \), \( w_1 \cdot x_2 = -1 \); so it is outside any region of interest for neuron 1, and causes neuron 2 to be created. Similarly, vertex \( x_3 = 010,011 \) causes neuron 3 to be created. Next, for vertex \( x_4 = 011010 \), we have \( w_1 \cdot x_4 = -3 \), so it is outside any region of interest for neuron 1; however, it is in the boundary region for both neurons 2 as well as 3, \( (w_2 \cdot x_4 = 1, w_3 \cdot x_4 = 1) \). Since none of the existing neurons can claim \( x_4 \), we “put aside” this vertex for re-consideration at a later stage. This is done by adding it in the shaded portion of Table 1, in row 13. Similarly, \( x_5 = 100,010 \) results in being put aside for re-consideration at a later stage, thereby resulting in row 14. The vertices \( x_6,\ldots,x_{12} \) are in the claim region of one of the existing neurons; so they cause the parameters of neurons to be updated. After we finish these 12 vertices, we continue the process of claiming these two vertices put aside earlier. Neurons 3 and 4 claim these vertices. Modified parameters (weights and threshold values) of these neurons are shown in rows 13 and 14.

7. Concluding remarks

For the example discussed in Section 6, earlier core-vertex-based methods, e.g., ETL, and IETL, need up to eight neurons for some orders in which input values are
given. However, for FCLA, we get the same neural network structure having four hidden neurons for any order in which input values are given. Thus, all the earlier core-based methods are sensitive to different sequences of inputs for such problems. Our method FCLA is not core-based and is not sensitive to different sequences of inputs for the example of circular region representation using 6-bit space quantization, which represents a class of “nicely clustered” problems.

To summarize, we propose a fast covering learning algorithm (FCLA) for Boolean neural networks based on geometrical expansion. FCLA has several advantages compared with previous learning algorithms for BNNs: (i) it possesses the generalization capability, but the previous learning algorithms (except for BLTA) do not, (ii) it is not core-based, and (iii) it is less sensitive to different input sequences of the same training set.

References