Implementing Universal CNN Neuron

Luxia Xu, Fangyue Chen, and Han Huang
School of Science, Hangzhou Dianzi University, Hangzhou, Zhejiang, P. R. China

Abstract—The universal CNN neuron can realize arbitrary Boolean functions including both linearly separable Boolean functions (LSBF) and linearly not separable Boolean functions (non-LSBF). However, determining the optimal (or near-optimal) orientation vector and the parameters in the multi-nested discriminant function contained within a universal CNN neuron is still a difficult task. By the aid of the DNA-like learning algorithm proposed a few years ago on the feedforward binary neural networks, the bottleneck problem can be solved if the number of input variables is not large.

Keywords: Universal CNN neuron, Boolean function, multi-nested discriminant function, DNA-like learning algorithm

1. Introduction

A cellular neural network (CNN) [1,2] is a biologically inspired system where computation emerges from a collection of simple nonlinear locally coupled cells. An uncoupled CNN cell (or neuron) is a standard CNN cell described by the following equations [3]:

\[ \sigma = \sum_{k,l \in \{i-1,i,i+1\} \times \{j-1,j,j+1\}} b_{k,l} u_{k,l} = \sum_{i=1}^{9} b_i u_i \]  (1)

\[ \dot{x}_{i,j} = -x_{i,j} + a_{i,j} f(x_{i,j}) + \sigma + z \]  (2)

\[ y_{i,j} = f(x_{i,j}) = \frac{1}{2}(|x_{i,j} + 1| - |x_{i,j} - 1|) \]  (3)

where \{i, j\} are two integer labels indicating the position of the CNN cell \( C_{i,j} \) within a two-dimensional grid, \( \{k, l\} \) are similar indices indicating the position of the neighboring cells, \( u_{k,l} \) represents the "9" inputs coming from the cell itself, and from its eight neighbors, \( x_{i,j} \) is the scalar state variable associated with the CNN cell and \( y_{i,j} \) is the associated output. The scalar variable \( \sigma \) is called an excitation, and in the case of the standard CNN cell, it is computed as a linear correlation between the feed-forward (controlling) template vector \( b = (b_1, \ldots, b_n) \), which is a repacked version of the \( B \) template [1], and its associated input vector \( U = (u_1, u_2, \ldots, u_n) \), as defined in (1) (here \( n = 9 \)). The second notation, with the index "9" replacing the pair of indices \( \{k, l\} \) is more general and can be applied to arbitrary choices of CNN architectures and spheres of influence. From this perspective, \( n \) represents the number of cell inputs and \( R^9 \) is the cell input space.

It was proved in [4] that when the central feedback coefficient \( a_{i,i} > 1 \), and \( a_{i,j} = 0, i \neq j \), the cell dynamics starting from \( x_{i,j}(0) = 0 \) converges towards a stable steady state for which \( y_{i,j}(\infty) = \text{sgn}(\sigma + z) \). The "infinity" symbol here denotes the time for the dynamics to reach a steady state output, and represents a small transient period which depends on the implementation technology. It follows that an uncoupled CNN cell maps a continuous input space \( R^9 \) into a binary (Boolean) output space. In the special case where the inputs are binary, the cell can realize various Boolean functions.

The standard CNN cell (1) to (3) can be generalized to the following universal CNN neuron (or universal CNN cell) [5-7].

\[ \sigma = \sum_{i=1}^{n} b_i u_i \]  (4)

\[ \dot{x} = -x + a f(x) + \omega(\sigma) \]  (5)

\[ y = f(x) = \frac{1}{2}(|x + 1| - |x - 1|), \]  (6)

where the discriminant \( \omega(\sigma) \) is a multi-nested piecewise-linear function

\[ \omega(\sigma) = s(z_m + z_{m-1} + \cdots + z_1 + z_0 + \sigma \| \cdots \|), \]  (7)

and \( \{s, z_0, \cdots, z_m\} \) is an additional set of \( m+2 \) parameters, \( s = 1 \) or \( -1 \). This model includes the standard CNN cell as a special case \( \omega(\sigma) = s(\sigma + z) \) (there is no absolute value sign in formula (7)), and has the advantages of simplicity and tractability due to its piecewise-linear nature.

It follows from (5) and (6) that the steady state CNN output equation is

\[ y(\infty) = \text{sgn}(\omega(\sigma)) = \begin{cases} 1 & \text{if } \omega > 0 \\ -1 & \text{if } \omega \leq 0 \end{cases} \]  (8)

Model (4) to (6) is said an universal CNN neuron (UCNNN) means that every \( 2^n \) Boolean functions of \( n \) input variables \( (u_1, u_2, \ldots, u_n) \) can be realized by finding an optimal (or near-optimal) orientation vector \( b = (b_1, \cdots, b_n) \) and a set of appropriate parameters \( \{s, z_0, \cdots, z_m\} \).

Since Boolean functions always play a key role in information processing and computer science, large-scale effective realization of Boolean functions is very important but also extremely difficult [3,8]. For example, how to "learn" the suitable template values of a CNN to perform a given task is a "bottleneck" [9]. All \( 2^{256} = 256 \) Boolean functions of 3 input variables were realized via UCNNN in [6]. However, for \( n = 4 \), the problem is so complex that not only computing the orientation vector but also the additional
set of parameters in UCNN for a non-LSBF takes a lot of computation time [7].

By the aid of DNA-like algorithm of the feedforward binary neural networks [10,11], the issues will be easier to be solved. This paper focuses on implementing UCNN, an effective method to determine the optimal (or near-optimal) orientation vector and the set of parameters in the multi-nested piecewise-linear function in UCNN for 4 inputs Boolean functions is obtained.

2. Realization of Boolean Functions via UCNN

2.1 Orientation vector and excitation

According to the convention used in the CNN literature, a “0” (or false) logic level is coded with -1, while a “1” (or true) logic level is still coded with 1.

A Boolean function of n variables is defined as the following binary map from \(-1,1^n\) to \{-1,1\}: \(F(U) = v\), where \(U = (u_1,u_2,\cdots,u_n)\) is the input window and \(v\) is the output corresponding to \(U\) of the map. Let \(k = \sum_{i=1}^{n} u_i 2^{n-i}\), then \(U = \tilde{U}(k)\) is the decimal code of the input window \(U\). There are \(2^n\) different input windows denoted by \(U(k)\) ([5, 7], \(k = 0, 1, \cdots, 2^n - 1\)). Thus, the map \(F\) can be rewritten as \(F(U(k)) = v_k\) ([5, 7], \(k = 0, 1, \cdots, 2^n - 1\)). Obviously, such a map can generate an output symbol \([v_0,v_1,\cdots,v_{2^n-1}]\) consisting of \(2^n\) symbols “-1” and “1”.

Conversely, a symbol \([v_0,v_1,\cdots,v_{2^n-1}]\) completely determines a Boolean function.

An \(n\) inputs Boolean function \([v_0,v_1,\cdots,v_{2^n-1}]\) can be coded by a decimal integer \(N = \sum_{i=0}^{2^n-1} \tilde{v}_i 2^{2^n-i-1}\), where \(\tilde{v}_i = 1\) if \(v_i = 1\) or else 0. For example, the decimal code of \([1, -1, 1, -1, 1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1]\) is 43567.

A Boolean function \([v_0,v_1,\cdots,v_{2^n-1}]\) is realized by a UCNN is equivalent to determining a vector \(b = (b_1,b_2,\cdots,b_n)\) and a set of parameters \(\{s_3, s_0, \cdots, s_m\}\) such that \(sgn(\omega(s_k)) = v_k\) in the UCNN (4) to (6), where \(s_k = \sum_{i=1}^{n} b_i u_i^{(k)}\) (if \(v_i = 0\), \(1\), \(2^n - 1\)), \(s_k = \sum_{i=1}^{n} b_i u_i^{(k)}\) (if \(v_i = 0\), \(1\), \(2^n - 1\)).

The vector \(b = (b_1,b_2,\cdots,b_n)\) is called an orientation vector of UCNN \([5, 7]\), \(2^n\) excitations \(s_k = b \cdot U(k)^T\) (if \(v_i = 0\), \(1\), \(2^n - 1\)) is a projection from \(n\) input windows \(U(k)\) (if \(v_i = 0\), \(1\), \(2^n - 1\)) to a scalar one-dimensional projection axis. These excitations \(\{s_k\}_0^{2^n-1}\) form a projection tape on the projection axis, and is a sequence which possesses many interesting properties such as symmetry, self-reproduction and self-similarity [10,12,13]. Furthermore, \(\{s_k\}_0^{2^n-1}\) is a DNA-like sequence in which only \(n\) values \(\{s_0,s_1,s_2,s_2,\cdots,s_{2^n-3},s_{2^n-2}\}\) are independent, and \(\{s_k\}_0^{2^n-1}\) can be obtained by coping the \(n\) excitation values [11,14]. There is a relationship between \(b = (b_1,b_2,\cdots,b_n)\) and \(\{s_0,s_1,s_2,s_2,\cdots,s_{2^n-3},s_{2^n-2}\}\) in the sequence \(\{s_k\}_0^{2^n-1}\) [10,13]:

\[
\begin{pmatrix}
   b_1 \\
   b_2 \\
   b_3 \\
   \vdots \\
   b_{n-1} \\
   b_n
\end{pmatrix} = \begin{pmatrix}
   ((n-3)s_0 - \sum_{i=0}^{n-2} s_{2i})/2 \\
   (s_{2(n-2)} - s_0)/2 \\
   (s_{2(n-3)} - s_0)/2 \\
   \vdots \\
   (s_2 - s_0)/2 \\
   (s_1 - s_0)/2
\end{pmatrix}
\]

2.2 Minimum number of transitions of Boolean function and DNA-like learning algorithm

Formula (9) shows that training orientation vector \(b = (b_1,b_2,\cdots,b_n)\) of a CNN is equivalent to training its \(n\) excitation values \(\{s_0,s_1,s_2,s_2,\cdots,s_{2^n-3},s_{2^n-2}\}\).

For a given orientation vector \(b = (b_1,b_2,\cdots,b_n)\), the corresponding excitation sequence \(\{s_k\}_0^{2^n-1}\) certainly has an order relationship: \(s_{i_0} \leq s_{i_1} \leq \cdots \leq s_{i_{2^n-1}}\), where \(\{i_0,i_1,\cdots,i_{2^n-1}\}\) is a replacement of \(\{0,1,2,\cdots,2^n-1\}\). A Boolean function \([v_0,v_1,\cdots,v_{2^n-1}]\), if it satisfies \(v_i = v_j\) when \(s_i = s_j\) (\(i \neq j\)), then say it conforms to the orientation vector \(b\), or the vector \(b\) conforms the Boolean function. Furthermore, if \(v_{i_{k_0}} \neq v_{i_{k_{0+1}}}\) when \(s_{i_{k_0}} < s_{i_{k_{0+1}}}\), where \(i_{k_0},i_{k_{0+1}} \in \{i_0,i_1,\cdots,i_{2^n-1}\}\), then say the Boolean function has a transition with respect to the vector \(b = (b_1,b_2,\cdots,b_n)\), denoted by \(T_{i_{k_0}}(s_{i_{k_{0+1}}})\). It is clear that \(N_{T_{v_0}}\), the number of transitions of a Boolean function, depends on the orientation vector \(b\), i.e., a Boolean function may have different numbers of transitions for different orientation vectors. Therefore, the orientation vector which guarantees the number of transitions of a given Boolean function is minimum certainly is an optimal or near-optimal orientation vector.

Example 1: For 4 inputs Boolean function: \([v_0,v_1,\cdots,v_{15}] = [-1,1,1,1,1,1,1,1,-1,1,1,1,1,1,1,1]\), its decimal code is \(\tilde{v}_{15} = 32504\), where \(\tilde{v}_i = 1\) if \(v_i = 1\) or else 0. If take the orientation vector \(b = (1,3,1,1)\), then one has \(s_0 = -6\), \(s_1 = s_2 = s_8 = -4\), \(s_3 = s_9 = s_{10} = -2\), \(s_4 = s_{11} = 0\), \(s_5 = s_6 = s_{12} = 2\), \(s_7 = s_{13} = s_{14} = 4\), \(s_{15} = 6\). Thus, the order relationship of the excitations \(\{s_k\}_0^{15}\) is \(s_0 < s_1 < s_2 < s_3 < s_4 = s_{10} < s_4 < s_{11} < s_5 = s_6 < s_{12} < s_7 = s_{13} = s_{14} < s_{15}\). It is easy to prove that the orientation vector \(b = (1,3,1,1)\) conforms to the Boolean function. Since \(v_0 \neq v_1\) \(s_0 < s_1\), \(v_12 \neq v_7\) \(s_{12} < s_7\). Thus, the Boolean function has two transitions, i.e., \(N_{T_{v_0}} = 2\) for the orientation vector \(b\).

If take another orientation vector \(b = (7,3,2,1)\) which also conforms to the Boolean function, then one has \(s_0 = -13\), \(s_1 = -11\), \(s_2 = -9\), \(s_3 = s_4 = -7\), \(s_5 = -5\), \(s_6 = -3\), \(s_7 = -1\), \(s_8 = 1\), \(s_9 = 3\), \(s_{10} = 5\), \(s_{11} = s_{12} = 7\), \(s_{13} = 9\), \(s_{14} = 11\), \(s_{15} = 13\). Thus, their order relationship is \(s_0 < s_1 < s_2 < s_3 = s_4 < s_5 < s_6 < s_7 < s_8 < s_9 < s_{10} < s_{11} = s_{12} < s_{13} < s_{14} < s_{15}\).
It follows that \(v_0 \neq v_1 (\sigma_0 < \sigma_1), v_0 \neq v_7 (\sigma_0 < \sigma_7), v_7 \neq v_8 (\sigma_7 < \sigma_8), v_12 \neq v_{13} (\sigma_{12} < \sigma_{13})\). Thus, it has four transitions, i.e., \(N_{|T_4} = 4\) for the orientation vector \(b = (7, 3, 2, 1)\).

The DNA-like learning algorithm is an effective algorithm to train the excitation sequence \(\{\sigma_k\}_{0}^{n-1}\) and compute the minimum number of transitions for a Boolean function [11,14]. In fact, the minimum number \(\text{Min}\{N_{|T_r}\}\) of transitions for all 4 inputs Boolean functions can be calculated by using the DNA-like algorithm, the detail is shown in Table 1. From the table, one can find that 4 inputs Boolean function have no more than 5 transitions. Obviously, the Boolean function which only has one transition is linearly separable. It was known that there were 1882 LSBFs in the set of 4 inputs Boolean functions [10,11,13], the result is consistent with the conclusion in Table 1.

| Table 1: Minimum number of transitions of 4 inputs Boolean functions |
|----------------|----------------|
| \(\text{Min}\{N_{|T_r}\}\) | 65536 Boolean functions with 4 inputs |
| Amount of BF | 1 | 2 | 3 | 4 | 5 |
| 1882 | 14244 | 30216 | 19002 | 192 |
| Percentage (%) | 2.87 | 21.73 | 46.11 | 28.99 | 0.29 |

2.3 Designing optimal multi-nested discriminant function

The \(m\)-nested discriminant function

\[
\omega(\sigma) = s(z_m + |z_{m-1} + \cdots + z_1 + |z_0 + \sigma| |\cdots||) \]

is a piecewise-linear function on variable \(\sigma\), its number of roots is \(2^m\), and the number doubles on \(m\) [7]. Thus, for a suitable set of parameters \(\{s, z_0, z_1, \cdots, z_m\}\), the \(\sigma\) axis can be divided into \(2^m + 1\) parts by the curve of the discriminant function.

In general, realizing a given \(n\) inputs Boolean function via a UCNN by designing an optimal multi-nested discriminant function should perform the following steps:

(1) determine the orientation vector \(b = (b_0, b_1, \cdots, b_n)\) which conforms to the function and the corresponding excitations sequence \(\{\sigma_k\}_{0}^{n-1}\) by using DNA-like learning algorithm;

(2) calculate all transitions of the function with respect to the vector \(b\);

(3) determine the number \(m\) based on the minimum number of transitions and the distribution of these transitions;

(4) calculate parameters \(\{z_0, z_1, \cdots, z_m\}\) and \(s\) based on the roots of the multi-nested discriminant function.

Example 2: For 4 inputs Boolean function \([v_0, v_1, \cdots, v_{15}] = [1, -1, 1, 1, 1, 1, 1, -1, 1, 1, 1, -1, 1, 1, 1, 1]\), whose decimal code is 32504 in Example 1, there are two transitions \(T_{|\{\sigma_0, \sigma_1\}\}}\) and \(T_{|\{\sigma_{12}, \sigma_{13}\}\}}\), one can take 1-nested discriminant function \(\omega(\sigma) = s(z_1 + |\sigma + z_0|)\). Let \(\omega(\sigma) = 0\), then its 2 roots are respectively \(\bar{\sigma}_1^{(1)} = -z_0 + z_1\) and \(\bar{\sigma}_1^{(1)} = -z_0 - z_1\). Take each of them as the middle value of two excitations that appear in the corresponding transition, i.e., \(-z_0 + z_1 = (\sigma_0 + \sigma_1) / 2\) and \(-z_0 - z_1 = (\sigma_{12} + \sigma_{13}) / 2\). Finally one obtains \(z_0 = 1, z_1 = -4\) and \(s = -1\), i.e., \(\omega(\sigma) = 4 - |1 + \sigma|\).

The two \(m\)-nested discriminant functions in Example 1 and 2 are shown in Fig. 1 and 2.

3. Conclusion

In literature [3], the founders, Prof. Chua and Roska of CNN and CNN-UM, once pointed "presently no theory exists which allows one to determine whether an arbitrary Boolean function is realizable by a CNN". Indeed, the UCNN (or universal CNN cell, UCNNC) is a good model to treat the difficult problem, which only needs a single neuron (cell). In this paper, the UCNN realizing a given Boolean function can effectively be implemented by using DNA-like algorithm if the number of the input variables is...
Fig. 1: Multi-nested discriminant function $\omega(\sigma) = 4 - |1 + \sigma|$ for 4 inputs Boolean function $[v_0, v_1, \cdots, v_{15}] = [-1, 1, 1, 1, 1, 1, -1, 1, 1, 1, -1, -1, -1, -1]$ in Example 1, the orientation vector is $b = (1, 3, 1, 1)$.

Fig. 2: Multi-nested discriminant function $\omega(\sigma) = -6 + | -10 + | -14 + \sigma||$ for 4 inputs Boolean function $[v_0, v_1, \cdots, v_{15}] = [1, -1, 1, -1, 1, -1, -1, 1, 1, 1, 1]$ in Example 2, the orientation vector is $b = (-8, -10, -2, 4)$.

not large. In future, further researches are needed such as on how to perform the Boolean function of bigger number of input variables via UCNNN, increase the computing speed, reduce the store space in a computer and so on.

Acknowledgment

This research was supported by NSFC (Grants 11171084 and 60872093).

References