

The Logical Processing of Association Matrix for Pattern Recognition

Kaiquan-Chen
Shenzhen Bos Soft co.,Ltd
415 Overseas Chinese high-Tech Venture Park
Longgang, Shenzhen China
E-mail:chenkaiquan@21cn.com
Tel/Fax: +86-755-28957858. Post:518172

Abstract

To reduce memory size and to solve the saturation problem arising from using the association matrix in the pattern recognition process, I introduce some developm-ents based on logical operations, By presenting examples, we can illustrate that these methods are more efficient and pragmatic.

Keywords: Pattern recognition, neural network, association matrix, and logical operation

1. Introduction

The main function of a neural network is to perform pattern recognition. That is. when an input pattern is presented it produces an output pattern that is related in some way to the input. To inspire us with some ideas. we begin this section by briefly reviewing some neural networks. In the following sections. we will analyze these networks and put forward some developments .

Boolean neural networks are built from conventional Boolean logic gates .Using these devices the input pattern (or image)is divided into smaller sub-sections, and each section is connected to an electronic memory device[1]. Increasing the divided sections results in smaller memory size and allows the system to have much generation ability .In the meantime, it unfortunately becomes easy to get saturated, and this problem severely affects the correctness of the output. Assuming that each divided image has n pixels and the total number of pixels is p , the memory size is

$$m = \frac{p \times 2^n \times d}{n}$$

where d is the length of the discriminator, and n has a value of anything between 1 and p.

As n increases,so does memory size, but this decreases the ability to generate On the other hand, when n reduces to 1 or 2,the memory reaches minimum value,but quickly become useless because of saturation. it is worth noticing that when n=0, we can define the memory size as

$$w = p \times d$$

In this case,the structure of the Boolean neural network reduces to the associative memory. An early example of associative memory is known as the learning matrix. The Hopfield network [2] and the bi-directional associative memory are central to these ideas Since the patterns to be associated could be anything, they actually have much wider application.

Compared with Boolean neural network, the associative memory has smaller memory size and to some extent can solve the saturation problem using the following learning rule[3]:

$$w = \eta[x] * [y] \quad (1)$$

Where x and y are the input pattern matrix and the output pattern matrix respectively and we often make $\eta = 1$ for simplicity .The number of the rows of the matrix w is equal to the length of the input pattern .

During both the training and test processes, the rows can be addressed by the corresponding 1's in the input pattern. Thus, it has fewer addresses than the Boolean neural network.

2. Analysis of the Pattern Recognition Process Performed by the Associative Memory

Using the learning rule(1), we can solve the saturation problem, but solving some problems causes others. For instance, let

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$w = [x]^T * [y] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 \\ 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix}$$

This learning phase is quite short since it requires only one calculation. Then, the threshold T needs to be calculated.

$$T = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

When $x_2 = [0 \ 1 \ 0 \ 1]$ is presented as the input, we get

$$[0 \ 1 \ 0 \ 1] \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 \\ 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix} = [0 \ 4 \ 2 \ 4]$$

Threshold the output, we have

$$[0 \ 1 \ 1 \ 1]$$

which is not the original x_2 .

The ability of the auto associative memory is that when it is shown a corrupted pattern, it still produces the correct response. For hetero associative memory, we don't know a direct way to calculate the threshold. To avoid undesirable stable states in Hopfield network, the number of neurons should be increased. For n neurons network, only about $p=0.15n$ patterns can be stored[4]. If the extra bits are added to patterns to increase the value of n, then these bits have to be chosen in such a way to increase the difference between the patterns. However, we can see no specific procedure for doing this. Another requirement is the choice of patterns to be stored. A purely arbitrary set of patterns is unlikely to produce a good response.

3. Transformation of the Association Matrix

After the learning phase, the association matrix is set up. The elements, or weights like in one layer network, of the matrix are going to be transformed into their reverse thermometer codes. Value i and its 4-bit reverse thermometer code is defined as the following:

i	reverse thermometer code
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 1 1 1
4	1 1 1 1

After processing the codes, we need to convert them back to the normal form. Compared with author's early work[5], this saves a large amount of memory.

4. Processing the Transformed Association Matrix

The transformed association matrix is processed between some certain rows in the matrix. Which rows participate in the operations is determined by the specific positions of 1's in the input pattern. Then, the following operations are performed:

- Execute logical AND operations between these rows
 - Execute the logical NOT operations on the other rows determined by the positions of 0's in the input pattern.
 - Execute logical AND operations between the pattern resulted from the first step and those from second step.
- The final result is the desired output pattern. To illustrate, consider the example from section 2, which resulted in an incorrect response. Now, convert the matrix w

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 \\ 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix}$$

into the following form

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 11 & 01 & 11 \\ 0 & 01 & 01 & 01 \\ 0 & 11 & 01 & 11 \end{bmatrix}$$

When $x_2=[0 \ 1 \ 0 \ 1]$ is presented as input, execute logical AND operations between the second and the fourth rows and then we have a pattern

$$[0 \ 11 \ 01 \ 11]$$

The complementary pattern of the third row is

$$[11 \ 10 \ 10 \ 10]$$

After performing the logical AND operations between these two patterns, we get

$$[00 \ 10 \ 00 \ 10]$$

It should be converted back as the pattern $[0 \ 1 \ 0 \ 1]$, which is the correct answer.

Two things are worth noticing. First, any row(s) containing only 0's will never take part in the calculation. Second, if a complementary row participates the operation but results in all 0's, omit the operation.

Now, consider the example again. If the input pattern $x_1=[0 \ 1 \ 1 \ 1]$ is corrupted either by replacing the first bit by 1 or replacing the last bit by 0, we arrive at the correct answer $[0 \ 1 \ 1 \ 1]$ again. Why? Because in the first situation we don't use the first all 0's row, and in the second case the complementary of the last row results in a pattern with all bits being 0's.

To show how hetero associative works, consider the following example:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$w = [x]^T * [y] = \begin{bmatrix} 0 & 2 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Convert w into the following form

$$\begin{bmatrix} 00 & 11 & 00 & 01 & 01 \\ 01 & 01 & 01 & 01 & 00 \\ 01 & 01 & 01 & 00 & 01 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Taking x_3 as input pattern, we get

$$[01 \ 01 \ 01 \ 00 \ 00]$$

The complementary of x_1 is

$$\bar{x} = [11 \ 00 \ 11 \ 10 \ 10]$$

These two patterns result in the following correct pattern y_3

$$[0 \ 1 \ 0 \ 1 \ 0 \ 0]$$

5. Conclusion

The calculations we present in this paper use less memory to help overcome the saturation problem in the pattern recognition process. Other than carefully choosing the training patterns, or adding some extra bits to patterns, it adopts some logical operations and proves to be a more pragmatic method.

References

- [1] Phil Picton, (2000), Neural networks, palgrave.
- [2] Hopfield, J. J. (1984) Neurons with graded responses have collective computational properties like those of two-state neurons, Proceedings of the National Academy of Science, United States of America, Biophysics, 81,3088-92.
- [3] Patrick K. S. (1990), Artificial neural systems, Pergamon Press.
- [4] John H, Anders K, Richard G. P, (1990) Introduction to the theory of neural computation, Addison—Wesley Publishing company.
- [5] Zhou J. Z. Densely coded matrix model of associative memory, Hildesheimer Informatik-Berichte, Germany.