A Neural Network Structure with Constant Weights to Implement Convex Recursive Deletion Regions

CHE-CHERN LIN
National Kaohsiung Normal University
Department of Industrial Technology Education
116 Ho Ping First Road, Kaohsiung,
TAIWAN, R.O.C.

Abstract: -A previous study has proposed a constructive algorithm to implement convex recursive deletion regions via two-layer perceptrons. However, the absolute values of the weights determined by the constructive algorithm become larger and larger when the number of nested layers of a convex recursive deletion region increases. The absolute values of the weights also depend on the complexity of the structure of the convex recursive deletion region. If the structure of the convex recursive deletion region is very complicated, the absolute values of the weights determined by the constructive algorithm could be very large. Besides, we still need to use the constructive procedure to get the parameters (weights and thresholds) for the neural networks. In this paper, we propose a simple three-layer network structure to implement the convex recursive deletion regions in which all weights of the second and third layers are all 1's and the thresholds for the nodes in the second layer are pre-determined according to the structures of the convex recursive deletion regions. We also provide the activation function for the output node. In brief, all of parameters (weights and activation functions) in the proposed structure are pre-determined and no constructive algorithm is needed for solving the convex recursive deletion region problems. We prove the feasibility of the proposed structure and give an illustrative example to demonstrate how the proposed structure implements the convex recursive deletion regions.

Key-words: - Multi-layer perceptrons, nested decision region, convex recursive deletion region.

1. Introduction

A multi-perceptron is a layered structure neural network where weights are used to connect the nodes between adjacent layers and to perform necessary computations to implement classification problems. Training algorithms are used to train the weights and get the desired mappings from inputs to outputs. However, using training algorithm to optimize the weight spends a lot of computational time. Some studies focused on the partitioning capabilities of multi-layer perceptrons [1-9]. It has been known that the first layer of a multi-layer perceptron produces decision boundaries for classifications and the rest of layers implement the mappings from the inputs to the outputs [1]. It has been known that single-layer percetrons can determine linearly separable decision regions, two-layer perceptrons can partition either convex open or closed decision regions, and three-layer perceptrons are capable of implementing of any shapes of decision regions [2]. Recent research also indicated that convex recursive deletion regions can be implemented by two-layer perceptrons [3]. A general study on the partitioning capabilities of two-layer perceptrons can be found in [4] where the authors presented the Weight Deletion/Selection Algorithm to examine the feasibility of implementation of decision regions. A constructive algorithm implementing convex recursive deletion regions using two-layer perceptrons has been proposed by [5] where the constructive algorithm served to determine the parameters of the two-layer perceptrons. Some intractable classification problems have been implemented by multi-layer perceptrons using space partitioning [6]. For a multi-layer perceptron, if one uses a constrain-based decomposition training architecture, the second layer and third layer of a three-layer perceptron function as logic "AND" and "OR", respectively [7].

The convex recursive deletion regions have been solved by the two-layer perceptrons where a constructive algorithm is used to determine the weights and the threshold for the two-layer perceptrons [5]. However, the absolute values of the weights determined by the constructive algorithm become larger and larger when the number of nested layers of a convex recursive deletion region increases. The absolute values of the weights also depend on the complexity of the structure of the convex recursive deletion region. If the structure of the convex recursive deletion region is very complicated, the absolute values of the weights determined by the constructive algorithm could be very large. This might probably cause computational overflowing problems for integer manipulations. Besides, we still need to use the constructive procedure to get the parameters (weights and thresholds) for the neural networks. In this paper, we propose a simple three-layer network structure to implement the convex recursive deletion regions in which all weights of the second and third layers are all 1's and the thresholds for the nodes in the second layer are pre-determined according to the structures of the convex recursive deletion regions. We also provide the activation function for the output node. In brief, all of parameters (weights and activation functions) in the proposed structure are pre-determined and no constructive algorithm is needed for solving the convex recursive deletion region problems. We also prove the feasibility of the proposed structure and give an illustrative example to demonstrate how the proposed structure implements the convex recursive deletion regions. Finally, we provide the conceptual diagram of the hardware implementation of the proposed network structure.

For the visual reason, in this paper, we use two-dimensional examples to explain how a multi-layer perceptron forms the decision boundaries, and how the proposed network structure implements the convex recursive deletion regions.

2. Preliminaries

2.1 Forming of Decision Regions

To explain how the first layer of a multi-layer perceptron forms decision boundaries, we present a two-class classification example implemented by a two-layer perceptron. This example is taken from [1,4] and shown in Fig. 1. Fig. 1(a) is a two-layer perceptron with two inputs, four nodes in the first layer (the hidden layer) and one node in the second layer (the output layer). Fig. 1(b) is the corresponding decision region where the two-dimensional input space is formed by the two inputs. In the figure, four partitioning lines linearly separate the decision region into 11 sub-regions which are numbered from 1 to 11. The shaded sub-regions belong to class A, while the blank ones belong to class B.

We use four nodes in the first layer to generate the four partitioning lines, respectively. In this paper, we use the same labels (z_1 to z_4) to represent the four nodes and their corresponding partitioning lines. Each of partitioning line divides the input space into two half spaces. We then use '0' and '1' to represent the two half spaces, respectively. The second layer makes a final decision by performing the mapping from inputs to outputs. The weights of the first layer are pre-determined for a given decision region. One only needs to determine the second layer weights of the two layer perceptrons.

We define θ value for a particular sub-region as follows [4, 10]

$$\theta_l = \sum_{k=1}^{r} w_k z_k \qquad (1)$$

where θ_l is the θ value of sub-region l, r is the number of partitioning lines in the decision region, and w_k is the weight linking first layer node z_k to the output node.

We use a hard limiter as the activation function, which is give by the following formula [4, 10]:

$$y = \begin{cases} 1 \ \ (\text{class A}) & \text{if} \quad \theta_l \geq \theta_h \\ 0 \ \ (\text{class B}) & \text{if} \quad \theta_l < \theta_h \end{cases} \qquad (2)$$

whereθ_h is the threshold for the second layer node.

It has been known that to successfully implement two-class classification problems, the minimum θ value of sub-regions belonging to class A must be greater than the maximumθ value of sub-regions belonging to class B.

2.2 Convex Recursive Deletion Region

Consider an n-dimensional Euclidean space R^n. Let C_0 be the input space in R^n, and $C_1, C_2\ldots, C_p$ be a series of nested convex polyhedrons with the following relation:

$$C_0 \supset C_1 \supset C_2 \supset \cdots \supset C_p \qquad (3)$$

A convex recursive deletion region S is a set of the form [5]:

$$S = (C_0 \cap \overline{C}_1) \cup (C_2 \cap \overline{C}_3) \cup \cdots \cup (C_{p-1} \cap \overline{C}_p) \qquad (4)$$

where $\overline{C}_i$ denotes the complement of C_i.

Fig. 2 shows the example of a convex recursive deletion region consisting of three nested convex polyhedrons: C_1, C_2, and C_3. Each of the three convex polyhedrons is bounded by a group of hyper-planes. We call the hyper-planes bounding a convex polyhedron "bounding hyper-planes" of the convex polyhedron. For example, in Fig. 2 (b), C_1 is bounded by bounding hyper-planes z_1, z_2, z_3, z_4, and z_5. C_2 is bounded by bounding hyper-planes z_6, z_7, and z_8. C_3 is bounded by bounding hyper-planes z_9, z_{10}, z_{11}, and z_{12}. A bounding hyper-plane divides the space into two linearly separable hyper-planes. The '1' side of a bounding hyper-plane of a convex polyhedron is the separable hyper-plane toward the convex polyhedron, and the '0' side is the other one. The '1' sides and '0' sides of the bounding hyper-planes for C_1, C_2, and C_3 are shown in Fig. 2 (b).

A pattern is in a convex polyhedron if and only if it is on the '1' sides of all bounding hyper-planes of the convex polyhedron. We can set up a threshold in each node in the second layer associated with a particular convex polyhedron to be the number of the bounding hyper-planes of the convex polyhedron, and therefore determine whether the pattern is in the convex polyhedron or not.

3. The Proposed Network Structure

3.1 Neural Network Structure

The proposed neural network is a three-layer structured network. Fig. 3 shows the example of the proposed network. The first layer of the network serves to form a convex recursive deletion region. The second layer detects the pattern location in the convex recursive deletion region. The third layer determines whether the pattern belongs to class A or class B according to the pattern location detected by the second layer. All of the weights in the second and three layers are set to be 1's. The threshold for a node in the second layer (θ_h) associated with a particular convex polyhedron is equal to the number of the bounding hyper-planes of the convex polyhedron. The output layer consists of only one node.

Let $v = \sum_{t=1}^{q} C_t$

where C_t is a second layer node and q is the number of the second layer nodes. The activation function for the output node y is as follows:

$$y = \begin{cases} (v+1) \bmod 2, \text{if } C_0 \cap \overline{C}_1 \text{ belongs to class A} \\ v \bmod 2, \quad \text{if } C_0 \cap \overline{C}_1 \text{ belongs to to class B} \end{cases} \quad (5)$$

where notation 'mod' denotes a modulus (remainder) operation of two integer numbers.

3.2 Proof of the Network Structure

The proof is straightforward. We explain it by Fig. 2 and Fig. 3. In Fig. 2, the convex recursive deletion region consists of three nested convex polyhedrons: C_1, C_2, and C_3. In this case, $C_0 \cap \overline{C}_1$ belongs to class B. If a pattern is in $C_0 \cap \overline{C}_1$, none of the nested convex polyhedrons contains the pattern. v is therefore equal to 0. By Eq. (5), $y = 0$ (class B). If a pattern is in $C_1 \cap \overline{C}_2$, only C_1 contains the pattern. v is equal to 1, and $y = 1$ (class A). If a pattern is in $C_2 \cap \overline{C}_3$, both C_1 and C_2 contain the pattern. v is equal to 2, and $y = 0$ (class B). If a pattern is in C_3, all of the three convex polyhedrons (C_1, C_2, and C_3) contain the pattern. v is equal to 3, and $y = 1$ (class A).

One can use the similar procedure to get the sequentially alternative classification results (0 and 1) for any convex recursive deletion regions.

Similarly, one can easily prove the feasibility of the proposed structure when $C_0 \cap \overline{C}_1$ belongs to class A.

Fig. 3 is the neural network to implement the convex recursive deletion region shown in Fig. 2. In Fig. 3, all of the weights of the second and third layers are 1's. The thresholds for the nodes of the second layer and the activation function for the output node are also demonstrated in the figure.

4. Conclusions

We proposed a simple three-layer network structure to implement the convex recursive deletion regions in which all parameters (weights and activation functions) are pre-determined according to the structures of the convex recursive deletion regions. No constructive algorithm is needed for solving the convex recursive deletion region problems. We used an illustrative example to explain how the first layer of a multi-layer perceptron forms the decision boundaries. We presented the neural network structure and finally proved the implementation feasibility of the network structure.

As for the directions of further studies, we suggest to add more layers to the proposed structure to improve the partitioning capabilities.

References

[1] J. Makhoul, A. El-Jaroudi, R. Schwartz, Partitioning Capabilities of Two-layer Neural Networks, *IEEE Trans. on Signal Processing*, Vol. 39, No. 6, 1991, pp.1436-1440.

[2] R. P. Lippmann, An Introduction to Computing with Neural Nets, *IEEE ASSP Mag.*, Vol.4, 1987,

pp. 4-22.

[3] R. Shonkwiler, Separating the Vertices of N-cubes by Hyperplanes and its Application to Artificial Neural Networks, *IEEE Trans. on Neural Networks*, Vol. 4, No. 2, 1993, pp. 343-347.

[4] C. Lin and A. El-Jaroudi, An Algorithm to Determine the Feasibilities and Weights of Two-Layer Perceptrons for Partitioning and Classification, *Pattern Recognition*, Vol. 31, No. 11, 1998, pp. 1613-1625.

[5] C. Cabrelli, U. Molter, and R. Shonkwiler, A Constructive Algorithm to Solve Convex recursive deletion (CoRD) Classification Problems via Two-layer Perceptron Networks, *IEEE Trans. On Neural Networks*, Vol. 11, No. 3, 2000, pp. 811-816.

[6] W. Fan and L Zhang, Applying SP-MLP to Complex Classification Problems, *Pattern Recognition Letters*, Vol. 21, 2000, pp. 9-19.

[7] S. Draghici, The Constraint Based Decomposition (CBD) Training Architecture, *Neural Networks*, Vol. 14, 2001, pp. 527-550.

[8] V. Deolalikar, A Two-layer Paradigm Capable of Forming Arbitrary Decision Regions in Input Space, *IEEE Trans. On Neural Networks*, Vol. 13, No. 1, 2002, pp. 15-21.

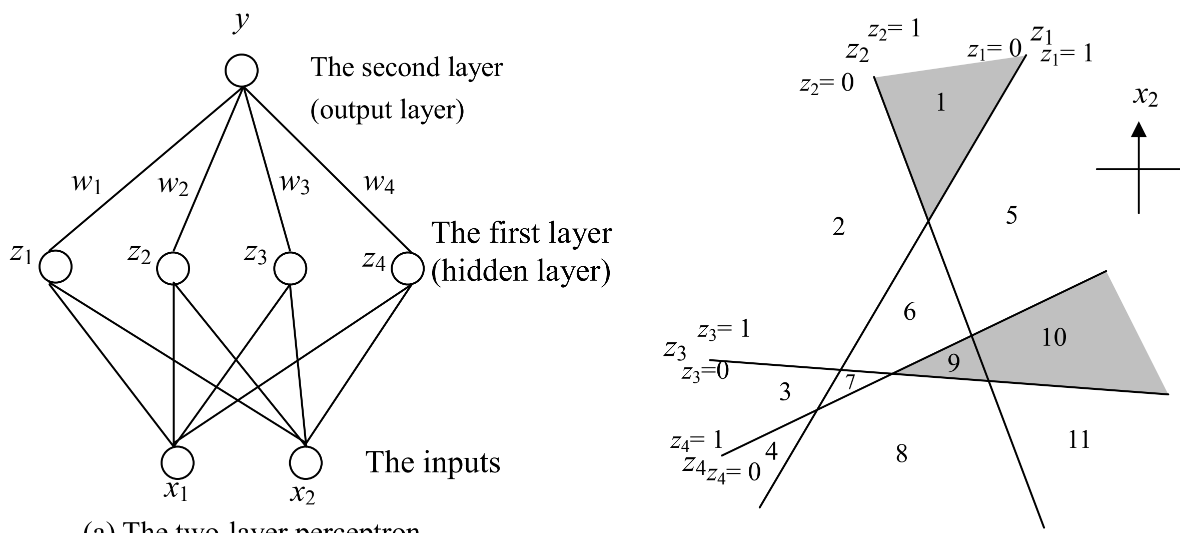
[9] G. Huang, Y Chen and H. A. Babri, Classification Ability of Single Hidden Layer Feedforward Neural Networks, *IEEE Trans. on Neural Networks*, Vol. 11, No. 3, 2000, pp. 799-801.

[10] C. Lin, Partitioning Capabilities of Multi-layer Perceptrons on Nested Rectangular Decision Regions Part I: Algorithm, *WSEAS Transactions on Information Science and Applications*, Issue 9, Volume 3, September, 2006, pp. 1674-1680.

[11] C. Lin, Partitioning Capabilities of Multi-layer Perceptrons on Nested Rectangular Decision Regions Part II: Properties and Feasibility, *WSEAS Transactions on Information Science and Applications*, Issue 9, Volume 3, September, 2006, pp. 1681-1687.

[12] C. Lin, Partitioning Capabilities of Multi-layer Perceptrons on Nested Rectangular Decision Regions Part III: Applications, *WSEAS Transactions on Information Science and Applications*, Issue 9, Volume 3, September, 2006, pp. 1688-1694.

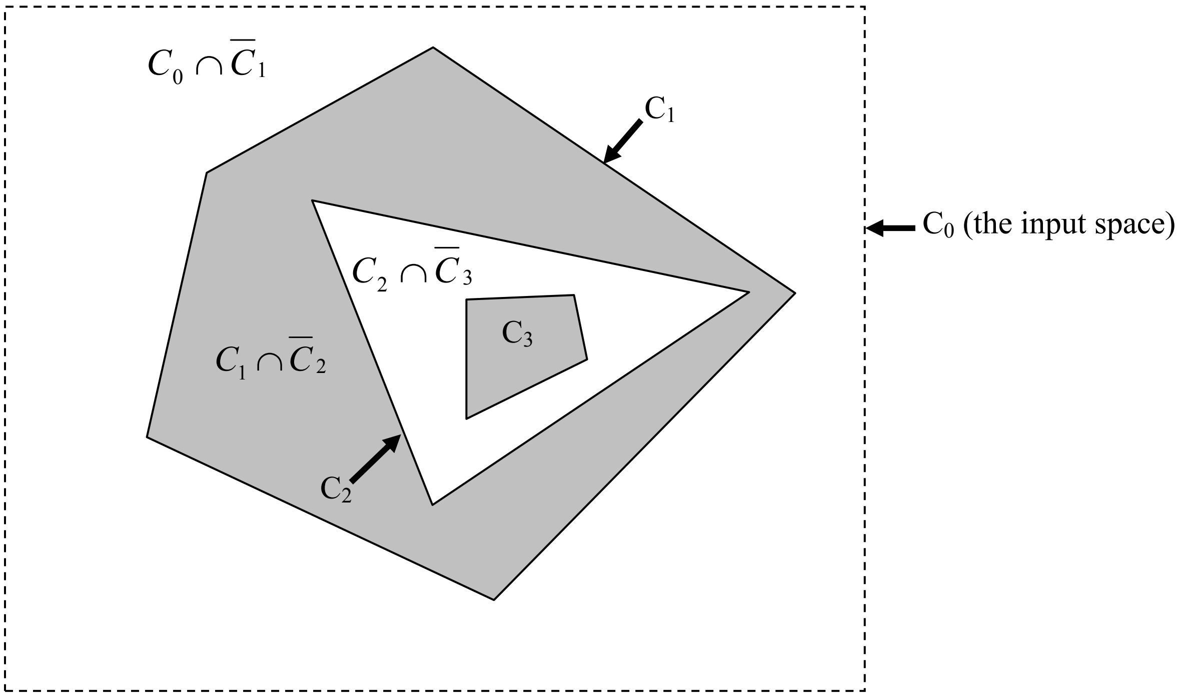
[13] C. Lin, A Constructive Algorithm to Implement Celled Decision Regions Using Two-Layer Perceptrons, *WSEAS Transactions on Information Science and Applications*, Issue 9, Volume 3, September, 2006, pp. 1654-1660.



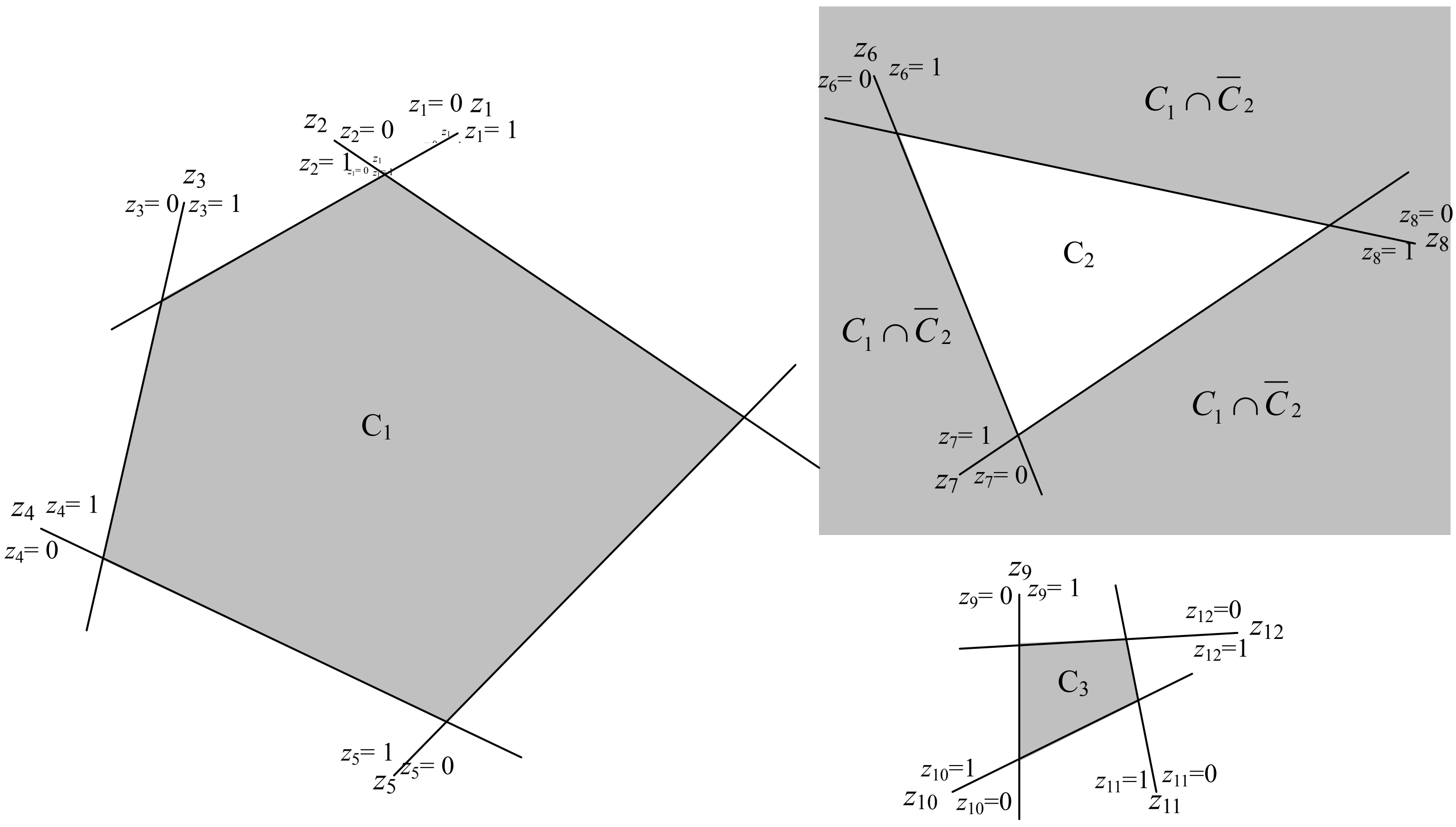
(a) The two-layer perceptron

(b) The decision region

Fig. 1: The two-layer perceptron and the corresponding decision region (taken from [1, 4]).


$C_0 \cap \overline{C}_1$
C_1
C_0 (the input space)
$C_2 \cap \overline{C}_3$
C_3
$C_1 \cap \overline{C}_2$
C_2

(a) The example of a convex recursive deletion region where C_0 is the input space in R^n.



(b) The bounding hyper-planes.

Fig. 2: The convex recursive deletion region and the bounding hyper-planes.

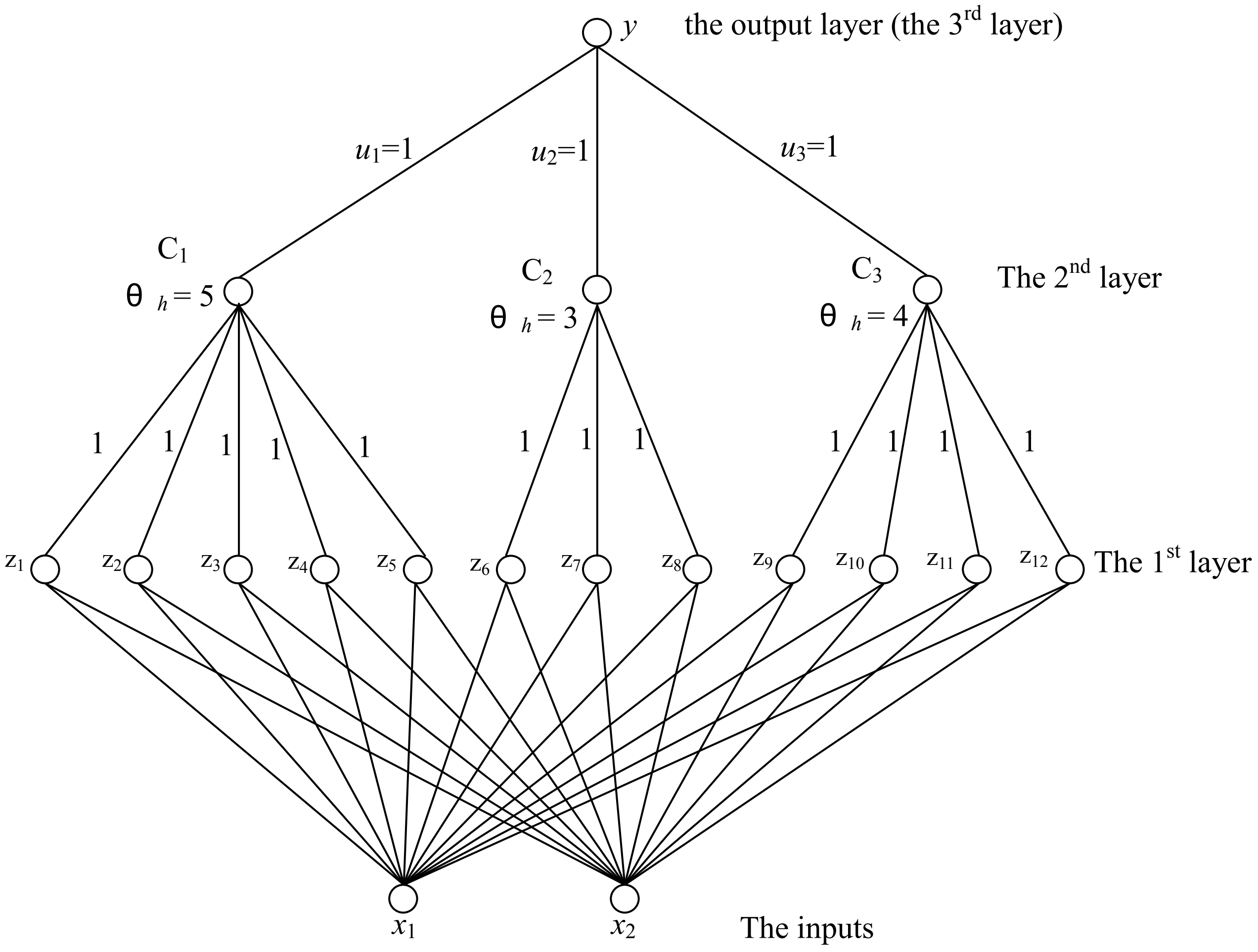


Fig. 3: The proposed network structure.