Proceedings of the 7th WSEAS International Conference on Neural Networks, Cavtat, Croatia, June 12-14, 2006 (pp96-101)

On Implementation of Nested Rectangular Decision Regions by Multi-layer Perceptrons I: Algorithm

CHE-CHERN LIN
Department of Industrial Technology Education
National Kaohsiung Normal University
116 Ho Ping First Road, Kaohsiung, 802
TAIWAN, ROC

Abstract: - There are three papers in a series of discussions related to the partitioning capabilities on nested rectangular decision regions using multi-layer perceptrons. We propose a constructive algorithm, called the up-down algorithm, to realize the nested rectangular decision regions. The algorithm determines the weights easily and can be generalized to solve other decision regions with the properties of similarity and dissimilarity. The first article gives preliminaries and describes the algorithm. The second one presents the properties of the algorithm and proves the feasibility. The last one discusses the applications of the algorithm using the properties of similarity and dissimilarity. As the first part of the series, this paper first discusses the partition capability of multi-layer perceptrons and then explains how two-layer perceptrons form the decision regions. The paper presents the formulas of determining the weights of the second layer and threshold of the output node for a two-layer percptron and demonstrates examples. Finally the paper discusses the generalization issues related to the proposed algorithm.

Key-Words: - Classification; Multi-layer perceptron; Nested decision region.

1. Introduction [1]

Necessary condition and sufficient condition, described in a mathematical viewpoint, of the implementation feasibility of two-layer perceptrons (TLPs) have been indicated in [2]. The first layer of a multi-perceptron produces decision boundaries and the rest of layers map the inputs to get the desired outputs [3]. Single-layer perceptrons 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 shape of decision regions [1,4]. Furthermore, any arbitrary decision regions can be approximated by TLPs [5]. It has been shown that convex recursive deletion regions can be implemented by TLPs [6]. Some particular nested decision regions implemented by TLPs have been presented in [7]. Partitioning properties of TLPs have been discussed in [8]. A general study on the partitioning capabilities of TLPs can be found in [1] where the authors presented the Weight Deletion/Selection Algorithm to examine the feasibility of implementation of decision regions. A constructive algorithm to implement convex recursive deletion regions has been presented in [9]. The space partitioning multi-layer perceptron model has been proposed to solve some intractable classification problems [10]. A constraint based decomposition training architecture for a multi-layer perceptron has been proposed where the second layer and third layer of a three-layer perceptron function as logic “AND” and “OR”, respectively [11].

In the series of the articles related to the partitioning capability on nested rectangular decision regions, we present an algorithm called the ***“Up-Down Algorithm”*** to implement nested rectangular decision regions. The up-down algorithm is so named because the values of weights are selected up (high value) and down (low value) alternatively. We explain how two-layer perceptrons form the decision regions and discuss the properties of the nested rectangular decision regions implemented by the proposed algorithm. We focus on how to apply the algorithm to implement some special decision regions according to the similarity and dissimilarity. To generalize the algorithm to solve more complex decision regions, we modify the network structure by adding a logic layer in the original neural network and discuss the partitioning capability of the modified network structure.

The three papers are organized as follows: the first article gives preliminaries and describes the algorithm; the second one presents the properties of the algorithm and proves the feasibility; the last one discusses the applications of the algorithm using the properties of similarity and dissimilarity.

As the first part of the series, this paper first discusses the partition capability of multi-layer

perceptrons and then explains how two-layer perceptrons form the decision regions. The paper presents the formulas of determining the weights of the second layer and threshold of the output node for a two-layer percptron and demonstrates examples. Finally the paper discusses the generalization issues related to the proposed algorithm.

2. Preliminaries [1, 3]

To explain how the first layer of a multi-layer perceptron forms decision boundaries, we present a two-class classification example implemented by a TLP with two inputs, four nodes in the first layer (the hidden layer) and one node in the second layer (the output layer), as shown in Figure 1(a) [1,3]. Figure 1(b) is the corresponding decision region where the two inputs generate a two-dimensional input space which is linearly divided into 11 sub-regions (numbered from 1 to 11) by the four partitioning lines [1,3]. In the figure, the shaded and blank sub-regions belong to class A and B, respectively.

It is important to know that each node in the first layer will form a two-dimensional partitioning line in the corresponding decision region because it is a linear combination of inputs x_1 and x_2. To implement the four partitioning lines, one therefore needs four nodes in the first layer with one-to-one corresponding relationship to the four partitioning lines. For convenience, we use the same labels (z_1 to z_4) to represent the four nodes and their associated partitioning lines. Each node in the first layer generates a "1" output if the input is on one side of its corresponding partitioning line, and a '0' output if on the other side [1,3]. The second layer performs mappings from the first layer to the second layer and makes a final decision [1,3]. It is important to note that the weights in the first layer are pre-determined if a decision region is established [1,3]. To implement the decision regions using TLPs, one only needs to train the weights of the second layer of the TLPs. The outputs of the first layer are the training examples for the second layer. Table 1 shows the training examples of implementation of Figure 1. In Figure 1(b), sub-region 1 is on the '0' side of line z_1, the '1' side of line z_2, the '1' side of line z_3, and the '1' side of line z_4. Therefore the training example for sub-region 1 is $z_1 = 0$, $z_2 = 1$, $z_3 = 1$, $z_4 = 1$.

The second layer serves to classify these sub-regions into the two classes (class A and class B). The θ value for a sub-region is defined as follows [1]

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

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

The two-class classification problem is implemented using the hard limiter [4], which is

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

where θ_h is the threshold for the second layer node.

It has been known that the necessary and sufficient condition for implementing two-class classification problems in a decision region is the minimum θ value of sub-regions belonging to class A must be greater than the maximum θ value of sub-regions belonging to class B [1,3,7].

A ***"nested rectangular decision region"*** is a decision region partitioned by vertical and horizontal lines and therefore structured by a series of concentric rectangular rings from the innermost ring to outermost ring [7]. It is important to mention that the nested rectangular decision regions are the special case of convex recursive deletion regions solved by the previous study [9]. To be brief, in the rest of the paper, we use ***"nested decision region"*** to represent ***"nested rectangular decision region"***. These rings in the nested decision region change their classes alternatively. A nested decision is called an ***"iA-jB decision region"*** if there are ***i*** rectangular rings belonging to class A and ***j*** rectangular rings belonging to class B in the decision region [7]. Figure 2(a) is a 2A-1B decision region where the entire decision region can be divided into 25 sub-regions, of which 17 sub-regions belong to class A (labeled A1 to A17), and 8 sub-regions belong to class B (labeled B1 to B8) [7]. These sub-regions are grouped into three nested rectangular rings. Two of them belong to class A (A1 and A2 to A 17) and one of them belongs to class B (B1 to B8). One therefore calls this decision region a 2A-1B nested decision region. In this decision region, sub-regions A2 to A17 form the outermost rectangular ring and sub-region A1 forms the innermost rectangular ring. Figure 2(b) is an example of a 2A-2B nested decision region [7]. Note that in a nested decision region a sub-region obtains '0' if it is on the left of a vertical partitioning line, and '1' if on the right of the partitioning line, as shown in Figure 2. Similarly, a sub-region obtains '0' if it is below a horizontal partitioning line, and '1' if above the partitioning line. Again, in a nested decision region the shaded areas represent class A, while blank ones represent class B.

It is important to note that for an *i*A-*j*B decision region, if the outermost ring belongs to class A, the number of partitioning lines in the region is $8 \cdot (i\text{-}1)$, which is dividable by 8 (note that $i > 1$). If the

outermost ring belongs to class A, the number of partitioning lines in the region is 8*i*-4 (dividable by 4 but not by 8). For example, there are 8 partitioning lines in the 2A-1B decision region (Figure 2(a)) and 12 partitioning lines in the 2A-2B decision region (Figure 2(b)).

It is also important to known that the innermost ring in a nested decision region consists of only one sub-region. For example, the innermost ring in Figure 2(a) consists of sub-region A1 only.

A sub-region is said to be a ***"corner sub-region"*** of a rectangular ring if it is located at one of the four corners of the rectangular ring. For example, in Figure 2(a), sub-regions A2, A6, A13, and A17 are the corner sub-regions of the outermost ring of the 2A-1B nested decision region. A sub-region is said to be a ***"sub-corner sub-region"*** of a rectangular ring if it is adjacent to one of the four corner sub-regions of a rectangular ring with a Hamming distance of "1" [1,4]. For example, in Figure 2(a) sub-regions A3 and A7 are sub-corner sub-regions of the outermost rectangular ring because they are adjacent to corner sub-region A2 with a Hamming distance of 1. However, sub-region B1 is not a sub-corner sub-region of the outermost ring because its Hamming distance to corner sub-region A2 is 2. Similarly, in Figure 2(a), sub-regions A5, A8, A11, A14, A12 and A16 are sub-corner sub-regions of the outermost rectangular ring.

Again, to brief the discussion, in the following sections when we mention a ***"ring"***, it means a ***"rectangular ring"***.

3 The Up-Down Algorithm

The partitioning lines in a nested decision region can be grouped into two groups: the group of vertical partitioning lines and the group of horizontal partitioning lines. For convenience, we sequentially number the vertical partitioning lines from the left to the right and the horizontal partitioning lines from the bottom to the top. For example, in Figure 2(a) the vertical partitioning lines are w_1, w_2, w_3, and w_4, and the horizontal partitioning lines are w_5, w_6, w_7, and w_8.

3.1 Determining the Weights

Consider a nested decision region with p partitioning lines. By letting $q = p/4$ (p is dividable by 4), the weights of the second layer of a TLP are determined by the following criterion:

$$w_k = \begin{cases} (-1)^{q+k}(q-k+1) & \text{for } k = 1, 2, .., q \\ (-1)^{q+k}(k-q) & \text{for } k = q+1, q+2, .., 2q \\ (-1)^{q+k}(3q-k+1) & \text{for } k = 2q+1, 2q+2, .., 3q \\ (-1)^{q+k}(k-3q) & \text{for } k = 3q+1, 3q+2, .., 4q \end{cases} \quad (3)$$

For example, the weights of the 2A-1B decision region (Figure 2(a)) are selected as follows: w_1 = -2, w_2 = 1, w_3 = -1, w_4 = 2, w_5 = -2, w_6 = 1, w_7 = -1, and w_8 = 2.

Similarly, the weights of the 2A-2B decision region (Figure 2(b)) are determined as follows: w_1 = 3, w_2 = -2, w_3 = 1, w_4 = -1, w_5 = 2, w_6 = -3, w_7 = 3, w_8 = -2, w_9 = 1, w_{10} = -1, w_{11} = 2, and w_{12} = -3.

It is important to note that the weights determined by the up down algorithm are horizontally and vertically anti-symmetrical. For Example, in Figure 2(b), $w_1 = -w_6$, $w_2 = -w_5$ (horizontally anti-symmetrical), $w_7 = -w_{12}$, $w_8 = -w_{11}$ (vertically anti-symmetrical).

3.2 Determining the Threshold (θ_h)

Let θ_{sc} be the θvalue of a sub-corner sub-region of the outermost ring of a nested decision region . The threshold of the output node (θ_h) of the TLP to implement the nested decision region is determined as follows:

$$\theta_h = \begin{cases} \theta_{sc} + 0.5 & \text{if the outermost ring is class B,} \\ \theta_{sc} - 0.5 & \text{if the outermost ring is class A.} \end{cases} \quad (4)$$

4 Generalizations

4.1 Generalization to Multi-Dimensional Cases

This algorithm is easy to generalize to multi-dimensional decision regions by adding two partitioning hyper-planes in each dimension when one adds a multi-dimensional ring to the original decision region. Consider an m-dimensional nested decision region with p partitioning hyper-planes (note that p is dividable by $2m$). Let $q = p/(2m)$. The weights of the second layer of a TLP are determined by the following criterion:

$$w_k = \begin{cases} (-1)^{q+k}(q-k+1) & \text{for } k = 1, 2, .., q \\ (-1)^{q+k}(k-q) & \text{for } k = q+1, q+2, .., 2q \\ (-1)^{q+k}(3q-k+1) & \text{for } k = 2q+1, 2q+2, .., 3q \\ (-1)^{q+k}(k-3q) & \text{for } k = 3q+1, 3q+2, .., 4q \\ (-1)^{q+k}(5q-k+1) & \text{for } k = 4q+1, 4q+2, .., 5q \\ (-1)^{q+k}(k-5q) & \text{for } k = 5q+1, 5q+2, .., 6q \\ \quad \text{M} & \quad \text{M} \quad \text{M} \\ (-1)^{q+k}[(2m-1)q-k+1] & \\ \quad \text{for } k = (2m-2)q+1, (2m-2)q+2, \Lambda, (2m-1)q & \\ (-1)^{q+k}[k-(2m-1)q] & \\ \quad \text{for } k = (2m-1)q+1, (2m-1)q+2, \Lambda, 2mq & \end{cases} \quad (5)$$

4.2 Generalization to Partially Nested Regions

A ***"partially nested decision region"*** is a particular nested decision region implemented by adding only part of the outermost ring to the original decision region. Partially nested decision regions result from adding any combination of the four partitioning lines (w_{left}, w_{right}, w_{bottom}, and w_{top}) to the original decision region. Figure 3 shows some examples of partially nested decision regions implemented by adding some combinations of the four partition lines to the original 2A-1B nested decision region. We will discuss the implementation feasibility of the partially nested decision regions in the next paper of the series of the studies (Part II: Properties and Feasibility).

5 Conclusions

We presented three papers to discuss the implementations of nested rectangular decision regions using multi-layer perceptrons where a constructive algorithm is used to realize these regions. The algorithm determines the weights easily and can be generalized to solve other decision regions with the properties of similarity and dissimilarity. The first article gave preliminaries and described the algorithm. The second one will present the properties of the algorithm and prove the feasibility. The last one will discuss the applications of the algorithm using the properties of similarity and dissimilarity.

In this article, we first discussed the partition capabilities of multi-layer perceptrons and explained how two-layer perceptrons form the decision regions. We also presented the formulas of determining the weights of the second layer and threshold of the output node for a two-layer perceptron. Finally we discussed the generalization issues related to the proposed algorithm.

.

Reference

[1] C. Lin, 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.

[2] P. J. Zwietering, E. H. L. Arts, J. Wessels, Exact Classification with Two-Layered Perceptrons, *Int. Journal of Neural Systems*, Vol. 3, No. 2, 1992, pp. 143-156.

[3] 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.

[4] R. P. Lippmann, An Introduction to Computing with Neural Nets, *IEEE ASSP Mag.*, Vol. 4, 1987, pp. 4-22.

[5] G. Cybenko, Approximation by Superpositions of a Sigmoidal Function, *Math. Contr., Signals, Syst.*, 1989, pp. 303-314.

[6] 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.

[7] C. Lin, A. El-Jaroudi, A Study on the Partitioning Capabilities of Two-Layer Neural Networks, *IEEE Int. Conf. on Neural Network*, Orlando, Florida, July 1994, pp. I-360-365.

[8] G. J. Gibson, C. F. Cowan, On the Decision Regions of Multilayer Perceptrons, *Proceeding of the IEEE*, Vol. 78, No. 10, 1990, pp. 1590-1594.

[9] C. Cabrelli, U. Molter, 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.

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

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

[12] 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.

[13] G. Huang, Y Chen, 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.

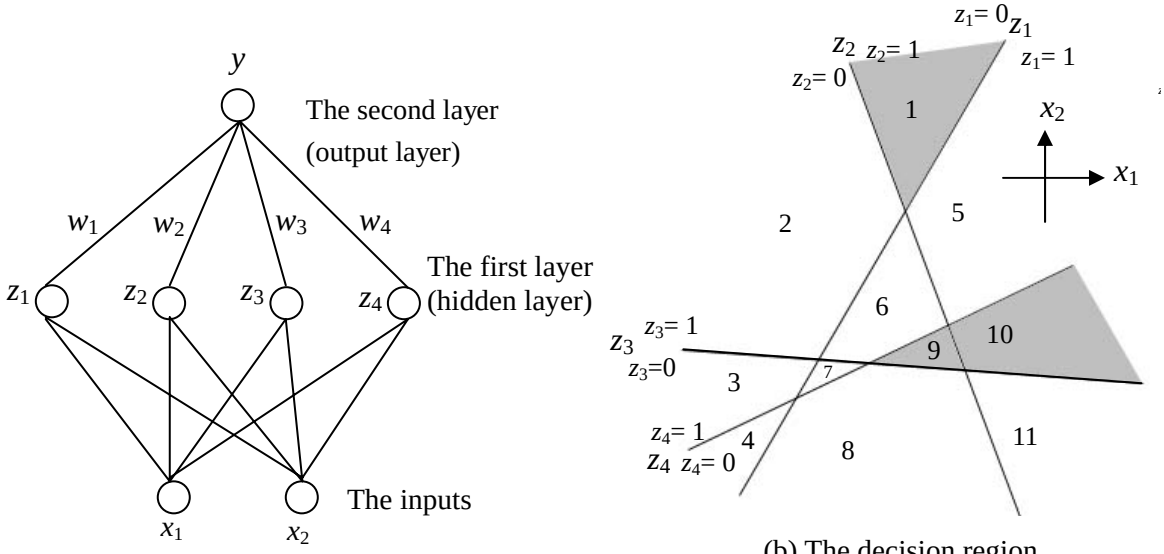


Figure 1: The TLP and the corresponding decision region (taken from [1, 3]).

Table 1 The corresponding patterns for Figure 1 (taken from [1, 3]).

SR	z_1	z_2	z_3	z_4	Class	SR	z_1	z_2	z_3	z_4	Class
1	0	1	1	1	A	7	1	0	0	1	B
2	0	0	1	1	B	8	1	0	0	0	B
3	0	0	0	1	B	9	1	0	1	0	A
4	0	0	0	0	B	10	1	1	1	0	A
5	1	1	1	1	B	11	1	1	0	0	B
6	1	0	1	1	B						

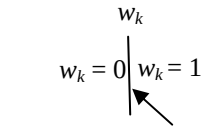
Remark: SR = sub-region

w_1	w_2	w_3	w_4	
A2	A3	A4	A5	A6	w_8
A7	B1	B2	B3	A8	w_7
A9	B4	A1	B5	A10	w_6
A11	B6	B7	B8	A12	w_5
A13	A14	A15	A16	A17	

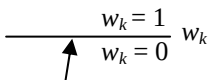
(a) The 2A-1B decision region

w_1	w_2	w_3	w_4	w_5	w_6		
B9	B10	B11	B12	B13	B14	B15	w_{12}
B16	A2	A3	A4	A5	A6	B22	w_{11}
B17	A7	B1	B2	B3	A8	B23	w_{10}
B18	A9	B4	A1	B5	A10	B24	w_9
B19	A11	B6	B7	B8	A12	B25	w_8
B20	A13	A14	A15	A16	A17	B26	w_7
B21	B27	B28	B29	B30	B31	B32	

(b) The 2A-2B decision region



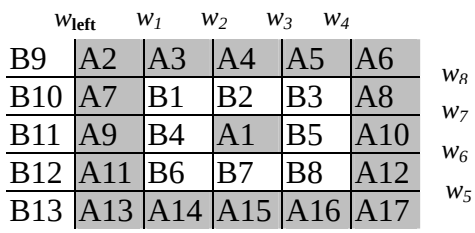
Vertical partitioning line



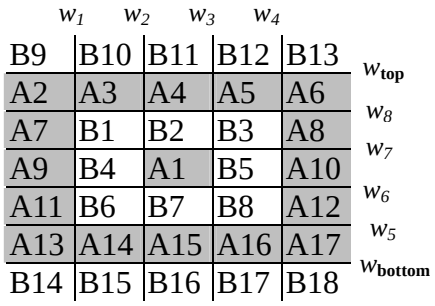
Horizontal partitioning line

Remarks:
1. Shaded areas indicate class A, while blank ones indicate class B.
2. A sub-region obtains '0' if it is on the left hand side of a vertical partitioning line, and '1' if on the right hand side of the partitioning line.
3. A sub-region obtains '1' if it is above a horizontal partitioning line, and '0' if below the partitioning line.

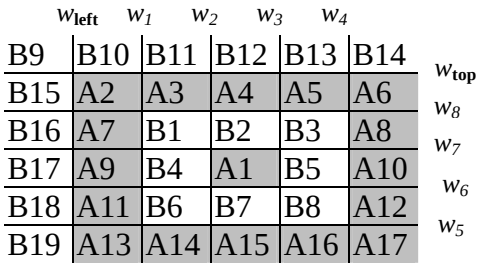
Figure 2: The examples of nested decision regions (taken from [7]).



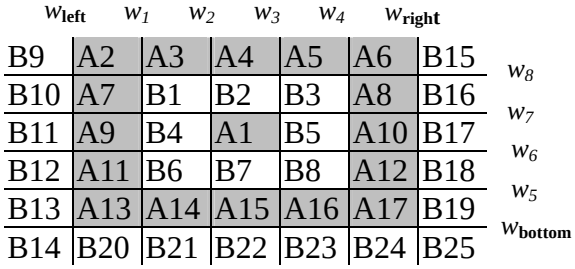
(a) Implemented by adding w_{left}



(b) Implemented by adding w_{top} and w_{bottom}



(c) Implemented by adding w_{left} and w_{top}



(d) Implemented by adding w_{left}, w_{right}, and w_{bottom}.

Figure 3: The examples of partially nested decision regions.