Distance-Discriminator Neural Networks for Classification and Pattern Recognition

LUCIANO DA FONTOURA COSTA,
JAVIER MONTENEGRO JOO, AND
ROLAND KÖBERLE

IFQSC – Instituto de Física e Química de São Carlos
Universidade de São Paulo
Caixa Postal 369
13560-970 São Carlos, SP, Brasil
FAX: +55 (162) 71-3616
e-mail: luciano@uspfsc.ifqsc.usp.br

Abstract. Distance-discriminator neurons – DDNs – and their combination in Distance-discriminator Neural Networks – DDNNs are proposed and discussed. DDNs, based on distance metric concepts, are able to discriminate whether a given point (x, y) belongs to a closed region such as diamond–, rectangle– and ellipse–bound regions, which are tasks traditionally performed by perceptrons. DDNs can also be straightforwardly modified in order to discriminate hollow regions having as outer boundaries the above mentioned geometrical figures or even combinations of them. The principal advantage of DDNNs over perceptrons is a substantial reduction of execution time and/or the amount of required hardware operators: many polygonal classification regions which would otherwise demand large perceptron structures can be discriminated with only a few DDNNs. DDNNs can also be easily programmed by design or automatically with the help of the Hough transform. Such issues as well as the relative advantages of DDNNs and perceptrons and a complete application example are presented and discussed in the present paper.

'Entia non sunt multiplicanda praeter necessitatem' – attributed to William of Occam, 1300–1349.

1 Introduction

Perceptrons and similar neural networks [Beale and Jackson (1990), Lippman (1987), Hush and Horne (1993)] are traditional alternatives for classification and pattern recognition tasks thanks to the following interesting features: (a) their operation can be trained (supervised or auto-organized); (b) their structures are inherently suitable for parallelization and (c) certain levels of generalization can be achieved [Lippman (1987), Beale and Jackson (1990)]. The model of neuron typically used in perceptrons is the *adaptive linear neuron*, or ADALINE, though higher order neurons are also possible [Widrow (1991), Beale and Jackson (1990)].

This paper presents distance-discriminator neurons – DDNs, as well as their combination into distance-discriminator neural networks – DDNNs, as means of improving the performance of perceptron-like feedforward artificial neural networks for the tasks of classification and pattern recognition. DDNs are inspired on distance metric operators that, though known for a long time, e.g. [Beale and Jackson (1990)], have been largely neglected as a means of speeding-up neural networks. Compared to ADALINEs, DDNs can be understood as more specialized types of neurons which require approximately the same amount of operations (or hardware operators, for the case of dedicated implementations) as ADALINEs, but that often can perform the same job as many ADALINEs.

The present paper starts by presenting the DDNs, their combination into DDNNs as well as their properties and training strategies and follows by comparing the performance of traditional ADALINE-based perceptrons and DDNNs. An actual application example is presented for which the DDNN resulted about 40% faster than the respective perceptron structure.

2 Distance Discriminator Neurons

A distance metric operator is a function which calculates some kind of distance between a given point (x, y) and another point $(\overline{x}, \overline{y})$ [Beale and Jackson (1990)]. Three types of distance metric operators are considered in this paper: diamond, rectangle (or

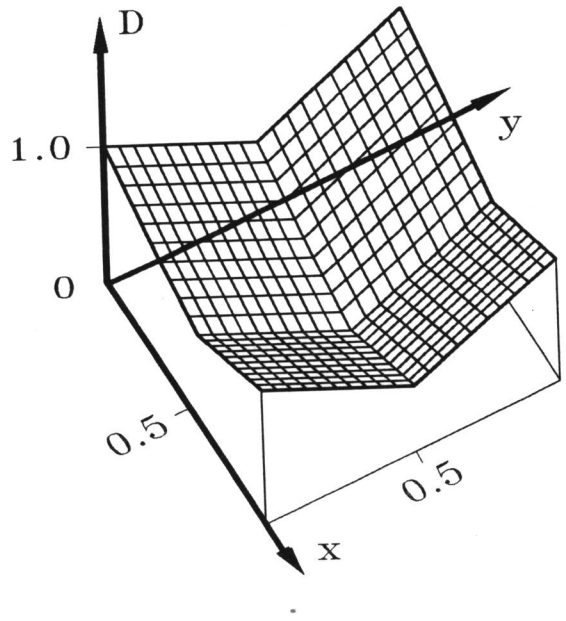


Figure 1: Mapping of D for $\overline{x} = \overline{y} = 0.5$ and $w_1 = w_2 = 1$.

'city- block') and Euclidian, which are respectively given by expressions (1), (2) and (3). Figure 1 illustrates the mapping of D in terms of the Cartesian coordinates x and y; it can be easily verified that the produced surface corresponds to the boundaries (except for the basis) of an inverted pyramid.

$$D = |(x - \overline{x}).w_1| + |(y - \overline{y}).w_2| \quad (1)$$

$$R = Max(|x - \overline{x}|.w_1, |y - \overline{y}|.w_2) \quad (2)$$

$$E = \sqrt{(x - \overline{x})^2.w_1^2 + (y - \overline{y})^2.w_2^2} \quad (3)$$

where w_1 and w_2 are non-negative real values.

By taking the difference between the distance-metric operator output (ie. D, R or E) and a specific threshold (respectively $\overline{D}$, $\overline{R}$ or $\overline{E}$), distance-discriminator neurons – DDNs – can be obtained which produce the outputs d, r, and e, as given by expressions (4) to (6) [1]. The diamond, rectangle and Euclidian DDNs are henceforth abbreviated as D–, R–and E–DDNs. The following three situations are defined according to the DDN output value: (a) *negative* – when the input point (x, y) is inside the classification region; (b) *zero* –if the input point lies on the region contour and (c) *positive* – for input points outside the region (the zero-output contours for the three considered DDNs are shown in Figure 2). It is thus clear that DDNs are inherently capable of classification according to their respective regions. It is also possible to define hollow classification regions by adopting expressions (7) to (9) instead of (4) to (6); the thus obtained classification regions are illustrated in Figure 3(a), (b) and (c). It should be observed that infinite straight bands can also be similarly obtained from ADALINE classification regions, as given by expression 11 and exemplified in Figure 3(d). It should be observed that the contour width does not depend exclusively on δ, but also on the respective neuron weights.

$$d = D - \overline{D} \quad (4)$$

$$r = R - \overline{R} \quad (5)$$

$$e = E - \overline{E} \quad (6)$$

$$d_h = |D - \overline{D}| - \delta \quad (7)$$

$$r_h = |R - \overline{R}| - \delta \quad (8)$$

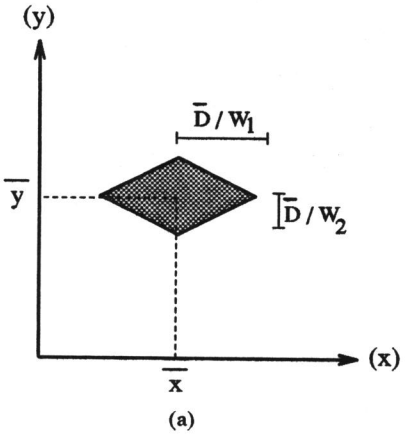
$$e_h = |E - \overline{E}| - \delta \quad (9)$$

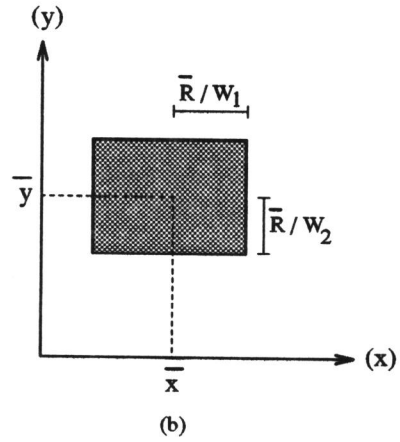
3 Distance-Discriminator Neural Networks

DDNs can be used as building blocks of three principal kinds of feedforward neural networks: those containing exclusively DDNs (hence pure–DDNNs); a mix of DDNs and Boolean-logic operators (i.e. those able to perform AND, OR and NOT, henceforth referred to as logic-DDNNs), and combinations of DDNs, ADALINEs and Boolean logic operators (henceforth abbreviated as hybrid– DDNNs). It should be observed that the translation of the sign of the DDN or ADALINE output into a logical value (true or '1' is henceforth adopted to indicate that the input point belongs to the classification region) is assumed to occur at the input of the logic operators.

While *pure-DDNNs* do not seem to yield any particularly interesting property, the other two kinds of DDNNs can be employed to substantially extend the repertoire of possible classification regions. For instance, by using the logical operations OR, AND and NOT, *logic-DDNNs* can be obtained which implement classification regions respectively given by the union, intersection and complement of the standard DDN regions: an asymmetric diamond can be defined through the union of two D-DDNs (see Figure 5(a)), and a nut-like region can be obtained by the intersection of a standard circular region and a complemented diamond region (see Figure 5(b)).

[1] E–DDNs can be implemented without square roots by adopting $E^2 = (x - \overline{x})^2.w_1^2 + (y - \overline{y})^2.w_2^2$ instead of expression (3).





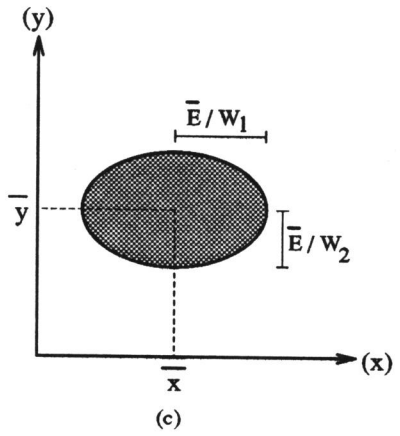
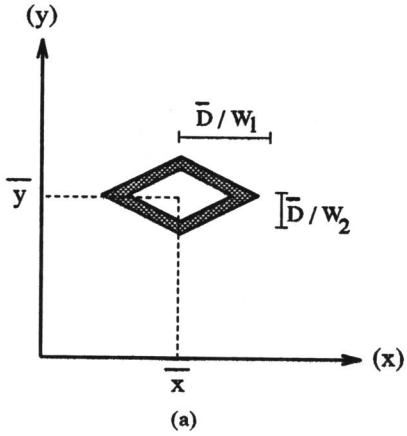
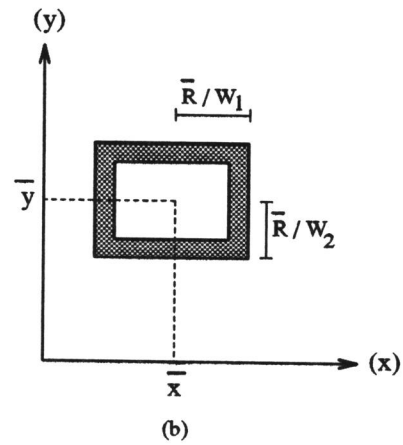


Figure 2: Output values for diamond- (a), rectangular- and Euclidian- (c) DDNs.





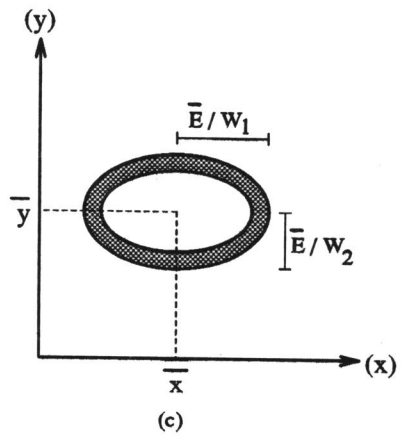


Figure 3: Output values for hollow diamond- (a), rectangular- and Euclidian- (c) DDNs.

Hybrid-DDNNs can be used as means of extending still further the possible classification regions through the combination of ADALINEs and DDNs. The basic two-dimensional ADALINE, defined by expression (10) [Widrow (1991)], defines the semi-plane classification region illustrated in Figure 4(a); applying a strategy similar to that used to define hollow DDNs, it is possible to modify the basic ADALINE in order to define the *band* [Costa and Sandler (1989), Costa and Sandler (1993), Costa (1992-a)], given by expression (11) and depicted in Figure 4(b). ADALINEs which implement trivial semi-planes, i.e. those with vertical or horizontal boundaries, are henceforth referred to as *trivial ADALINEs*; such neurons can be straightforwardly obtained by comparing respectively the x and y coordinates of each input point to be classified. Examples of hybrid-DDNNs yielding triangle, trapezium and section-of-circle classification regions are respectively exemplified in Figures 5(c), (d) and (e).

$$a = x.w_1 + y.w_2 + \overline{A} \tag{10}$$

$$a_h = |a| - \delta \tag{11}$$

Before proceeding to describe how perceptrons and DDNNs can be programmed to perform classification and pattern recognition, it is opportune to review the meaning of these two operations. The operation which determines whether a given point (x, y) lies inside, outside or on the border of a specific region is henceforth referred to as *classification*. Pattern recognition consists of classifying every input point with respect to the region: the input is understood to contain an instance of the sought pattern whenever the amount of input points found to be within the region is higher than a fixed threshold T. Pattern recognition can be performed in two principal ways: by scanning the whole set of input points serially over the recognizing neural net as it would be typical in sequential software executions, or by associating every input point to a copy of that neural network in such a way as to achieve massively parallel execution. It is interesting to notice that DDNNs and perceptrons when applied to pattern recognition perform exactly the same operation as template matching; the advantages of the neural aproaches consist basically in the fact that these latter are usually more compact (just some weights, rather than the complete template, have to be stored) and that the associated classificaiton regions, being mathematically defined, can be straightforwardly transformed (e.g. dilation, rotation, translation).

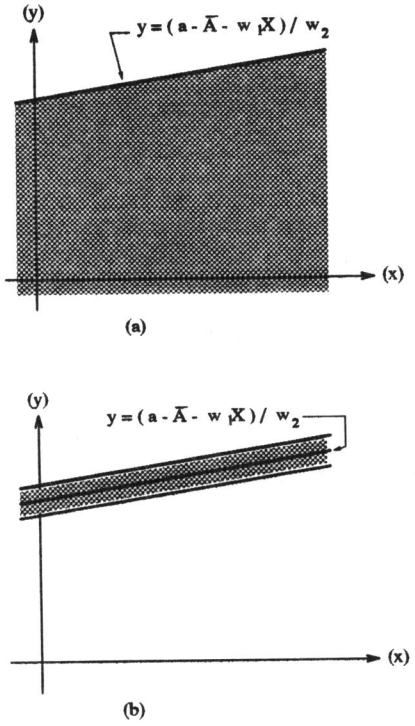


Figure 4: Semi-plane (a) and band (b) classification regions defined by ADALINEs.

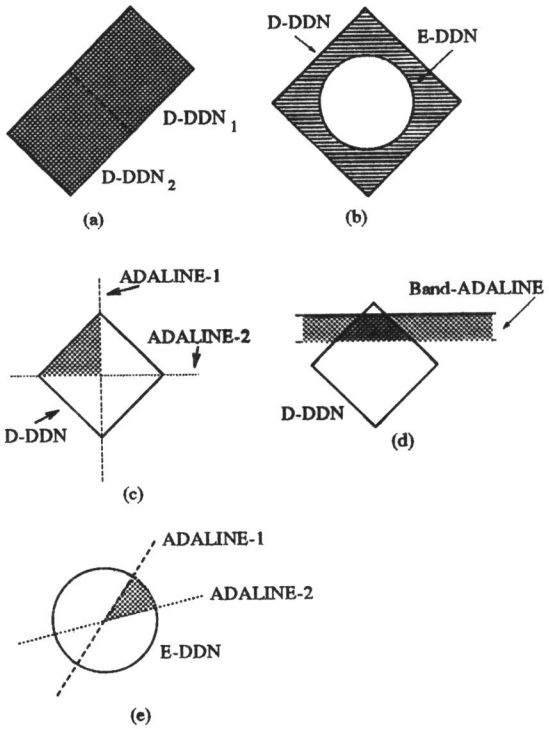


Figure 5: Asymmetric diamond (a), nut-like boundary (b), straight-triangle (c), trapezium (d) and a section of a circular region (e).

4 Programming DDNNs

Assuming that the intended classification regions are composed of parametric curves, it is in principle possible to devise algorithms which program the desired operation of DDNNs according to parametric and endpoint information about the classification region constituent curves produced by some curve detection algorithm. For simplicity's sake, the following discussions are restricted to polygonal regions – the extension to other curves is straightforward.

Since polygons are composed exclusively of straight line segments, an algorithm suitable for such kind of curve detection has to be firstly obtained. An effective Hough-transform-based alternative for digital bar segment detection (digital bar segments are henceforth understood as extensions of straight line segments with non-infinitesimal width) has been presented in [Costa and Sandler (1993), Costa (1992-b), Costa (1992-a)]. In that approach, an improved Hough transform variation, namely the binary Hough transform, is adopted as a means of detecting possible digital bars in the image, which are confirmed through a post-Hough transform connectedness analysis also capable of determining the endpoints of the actual digital bar segments. In order to reduce the amount of replicated and broken segments usually produced by Hough transform-based techniques, a merging stage is applied after the connectedness analysis. The final product of this framework is a list L of the parameters and endpoints of most of the significant straight segments in the image. The performance of such a framework has been quantitatively assessed in terms of execution speed, amount of false and true detected lines and accuracy for the segment parameter determination in [Costa (1992-a)].

Now, an algorithm must be devised which processes the above mentioned list L of straight segments in order to derive the suitable DDN weights. One possible algorithm, which treats separately D– and R–DDNs and which is not guaranteed to produce optimal results, is presented in the following. The algorithm begins by looking for parallel straight segments in L which indicate the presence of rectangular regions or sub-regions in the image. The shortest of the parallel lines must be chosen, the other parallel will have to be cut in such a way as both parallels have the same length, this length will give the value of the weights w_1 and w_2. The point where the diagonals joining the end points of these parallel lines meet will yield the point $(\overline{x}, \overline{y})$ where the R–DDN is centered. This process is then repeated with other pairs of parallel segments. From the list of straight line segment endpoints it is also possible to find the corners defined by a pair of straight segments. Once a pair of these segments has been identified, they can be used to program a D–DDN. In this case, it will have to be determined if both lines are symmetric with respect to either a vertical or a horizontal/vertical lines passing through the vertices. Here also both lines will have to be of the same length, otherwise the longest one will have to be cut to the length of the shortest. As usual, the length of the lines will yield the value of the weights w_1 and w_2. The D–DDN center can be determined similarly to the R–DDN case.

The above described programming strategy can be extended to the situation where many distinct patterns are present in the input image. This can be achieved by applying a second conectedness analysis which assigns pointers expressing the connections between distinct straigth segments. In this way, the closed polygonal region will be reconstructed. Each of the thus obtained polygons will then be treated by the above presented algorithm.

5 Comparing DDNNs and Perceptrons

The relative advantages and disadvantages of DDNNs and perceptrons for classification and pattern recognition respectively to a series of performance parameters such as amount of required hardware resources, execution time, programming and generalization capability are discussed in the following subsections.

5.1 Execution Time and Required Hardware

The basic operations in the two-dimensional ADALINE and the D–, R– and E– DDNs are respectively illustrated in Figures 7 and 6(a) to (d) It can be easily verified from this figure that an ADALINE and a D–DDN respectively require four (two products and two additions) and eight (two products, four additions and two simple absolute value) basic operations. Assuming that all these operations can be executed in the same time period T_c, it follows that a total of respectively $4.T_c$ and $8.T_c$ basic cycles will be required by sequential execution of each of these neurons. For dedicated parallel hardware implementations where each neuron is implemented as an individual hardware structure, this will mean that less hardware will have to be used to achieve the same operation. It thus becomes clear that DDNNs will only allow execution time reductions for those regions which can be conveniently partitioned into diamonds, rectangles or pieces of these in such a way that each DDN implement at least two line segments of the region contour. Given the large percentage of such types of features in natural and man-made ob-

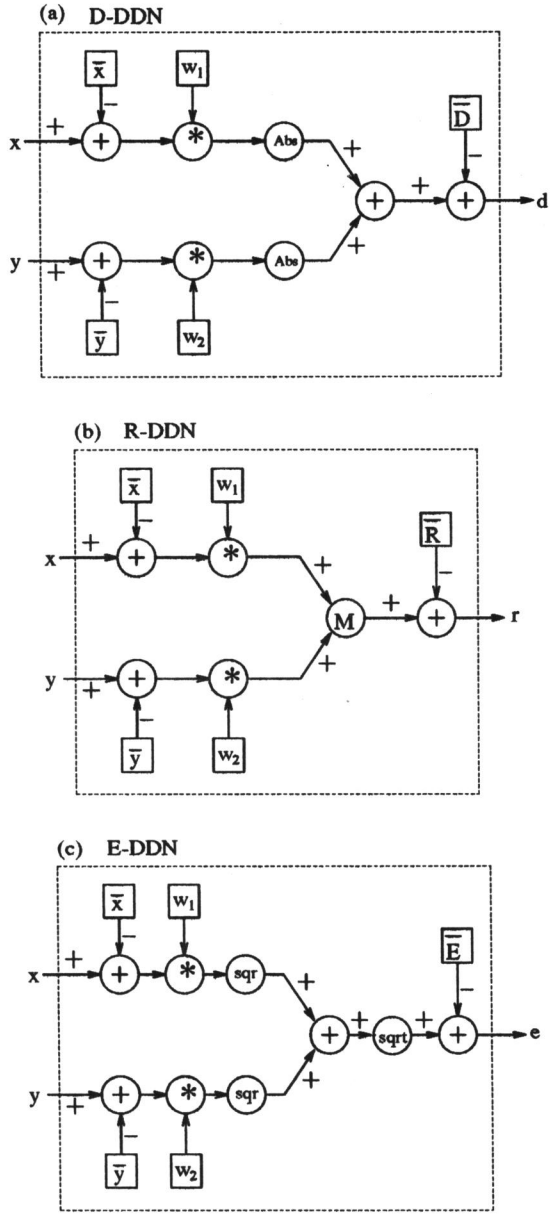


Figure 6: The constituent operations of the D– (b), R– (c) and E–DDNs (d).

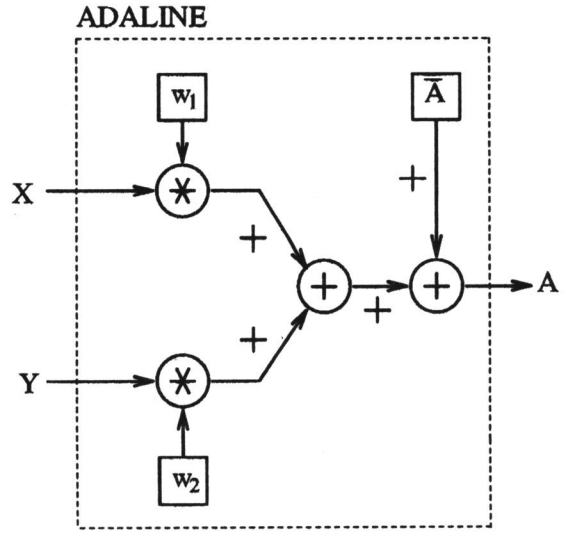


Figure 7: The ADALINE constituent operations.

jects and it is reasonable to expect that DDNNs will often lead to performance benefits (see Section 6).

5.2 Programming

Back propagation (or delta-rule) has traditionally been used for perceptron programming. Such a strategy relies on the successive modification of the ADALINE weights in the several layers in such a way as to minimize the difference between the current and desired outputs, which characterizes the programming strategy commonly referred to as supervised-learning. Many interactions are typically required until the perceptron, which usually starts with random weights, be programmed in the desired way. It is clear that a perceptron intended for recogniton of polygonal classification region needs N_c ADALINES, one for each straight line segment in the sought pattern contour. Regions which are not concave [2] have to be partitioned into concave pieces, at the cost of an additional ADALINE per partition. Direct programming of the weights $\overline{A}$, w_1 and w_2 of each of these ADALINEs can be done by taking as basis the orientation and position of each of the straight line segments defining the sought pattern contour, which can be obtained by using the Hough transform [Costa and Sandler (1989)]. The evidences from these ADALINES in the first perceptron layer have to be combined by additional structures in the following perceptron layers. In this paper we assume that this combination is done by Boolean logical neurons [Widrow (1991)]. Similarly, in a DDN-based

[2] A region R is said to be concave *iff*, given any two internal points p_1 and p_2, all the points on the line extending from p_1 to p_2 are also points of R.

approach, the position and horizontal and vertical extents of each of the regions can be used as basis to respectively determine the suitable DDN structure reference point and weights.

When discriminating a polygonal region with ADALINES, the geometrical distribution of these is unique and it is fixed by the shape of the polygonal region itself, whereas when discriminating the same polygonal region with DDNs there is freedom for the programer to choose which combination of polygon DDN structures presents potential for allowing better convergence and programming strategies . This means that a smarter programer will be able to choose a better combination of polygons to discriminate a given region.

5.3 Generalization Capability

The generalization properties of pattern recognition with perceptrons arise from two principal reasons: the tolerance commonly allowed for the threshold T and the re-utilization of ADALINES.

It can be easily verified from their underlying expressions that D–DDNs and their respective ADALINE counterparts exhibit basically the same recognition properties, except for differences implied by the finite numeric representation usually required for computational implementations.

Let us illustrate the re-usability concept through a simple example. Given in Figure 8(a) containing two regions, a square and a triangle with a collinear side. The minimal perceptron structure capable of recognizing such a pattern requires six ADALINES: four to recognize the square and only two to recognize the triangle, since the collinear sides can be implemented with the same ADALINE, see Figure 8(b). It can be straightforwardly verified that DDNNs do not allow such a re-usability capability, but, would this imply a severe disadvantage? Before answering this question, it should be firstly observed that it is relevant only for situations in which the collinear sides belong to regions which can be implemented with DDNs, since all the other cases should be anyway implemented with ADALINES. Such a situation is illustrated by the nine diamonds in the 3 X 3 arrangement shown in Figure 9. Recognizing these diamonds with perceptrons will invariably demand programming twelve perceptrons, whereas if DDNs are used instead, only nine D–DDNs are required. It is thus clear that the reduced execution time and/or amount of hardware resources allowed by DDNs for such situations will usually compensate for their lack of re-usability.

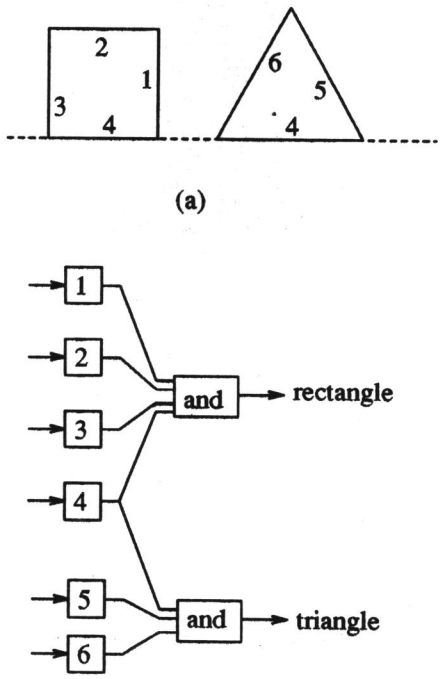


Figure 8: A triangle and a square with collinear sides (a) and the minimal logic-perceptron for its recognition (b).

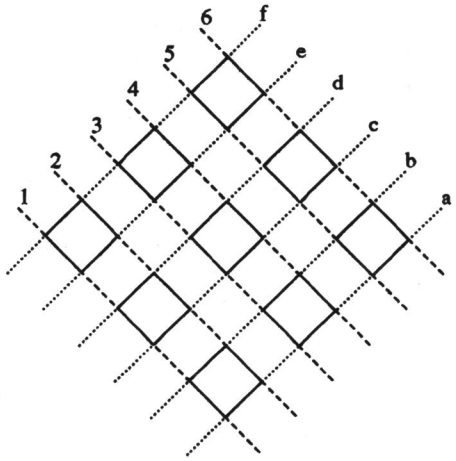


Figure 9: Nine diamonds.

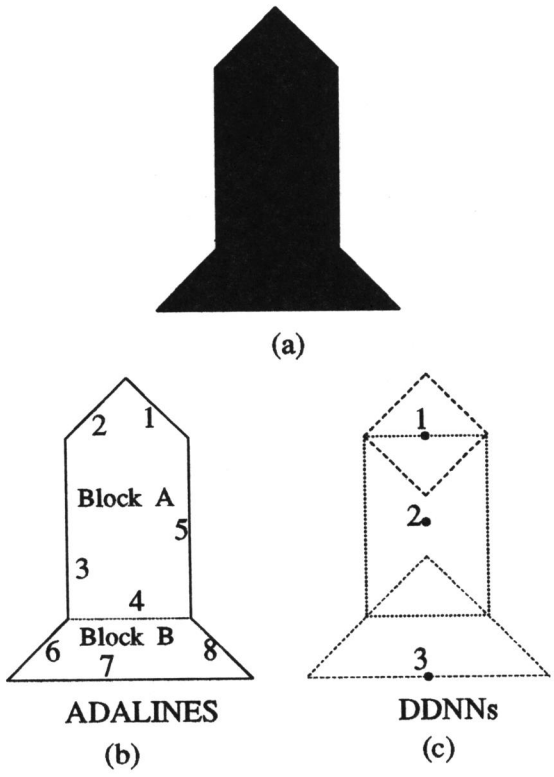


Figure 10: The sought pattern (a) and its possible decomposition for perceptron-based (b) and DDNN-based (c) pattern recognition.

6 Application Example and Experimental Results

The relative performance of perceptrons and DDNNs is illustrated by the following actual application example for pattern recognition. Let us assume that a specific instance of the 511-element-pattern in figure 10(a) is to be verified in an image.

It is clear from figure 10(b) that a minimum of 7 ADALINES have to be implemented only in the first perceptron layer; an additional ADALINE is required in order to partition the pattern into two concave regions and more structures (assumed to be Boolean logic neurons) are required in the following layers in order to combine the first layer outputs. A possible pseudo-code algorithm for the recognition of the rocket–like object in figure 10(a), where $ADALINE()$ is a function which returns the ADALINE output for the input point (x, y), is given below (it has been assumed that $w_{1,i}$, $w_{2,i}$ are non-negative real values and $w_{0,i}$ is a negative real value):

Function inblockA:
inblockA:=false
If $ADALINE(x, y, w_{0,1}, w_{1,1}, w_{2,1}) \leq 0$
and $ADALINE(x, y, w_{0,2}, w_{1,2}, w_{2,2}) \geq 0$
and $ADALINE(x, y, w_{0,3}, w_{1,3}, w_{2,3}) \geq 0$
and $ADALINE(x, y, w_{0,4}, w_{1,4}, w_{2,4}) \geq 0$
and $ADALINE(x, y, w_{0,5}, w_{1,5}, w_{2,5}) \leq 0$
then inblockA:=true

Function inblockB:
inblockB:=false
If $ADALINE(x, y, w_{0,6}, w_{1,6}, w_{2,6}) \leq 0$
and $ADALINE(x, y, w_{0,7}, w_{1,7}, w_{2,7}) \geq 0$
and $ADALINE(x, y, w_{0,8}, w_{1,8}, w_{2,8}) \geq 0$
and $ADALINE(x, y, w_{0,3}, w_{1,3}, w_{2,3}) < 0$
then inblockB:=true

Main:
count:=0
For every image feature element do
If (inblockA=true) or (inblockB=true)
then count:=count+1
If count $\geq$ T then recognize pattern

In a DDNN implementation, the rocket–like object in figure 10(a) has to be partitioned into diamonds, rectangles or parts of these. A minimal partition requiring a total of just three DDNs (a whole diamond, a whole rectangle and the superior half of another diamond) is illustrated in Figure 10(c) and the respective pseudo-code detection procedure, where $D()$ and $R()$ are functions which respectively implements the D- and R- DDNs and $\overline{Y}$ is the y-coordinate used for partitioning the lower diamond according to the horizontal axis, is given below:

Function inside:
inside:=false
If $D(x, y, \overline{x_1}, \overline{y_1}, w_{1,1}, w_{2,1}) \leq 0$
or $R(x, y, \overline{x_2}, \overline{y_2}, w_{1,2}, w_{2,2}) \leq 0$
or $((y > \overline{Y})$ and $(D(x, y, \overline{x_3}, \overline{y_3}, w_{1,3}, w_{2,3})) \leq 0)$
then inside:=true

Main:
count:=0
For every image feature point do
If inside=true then count:=count+1
If count $\geq$ T then recognize pattern

One should notice that the simpler test required to implement the trivial ADALINE region (i.e. '$y > \overline{Y}$' in the fourth line of the function 'inside') should always precede the DDN function calls in order to statistically improve the overall execution time.

Both procedures above were translated into Pascal and executed in a VAX- 6420 computer. When applied to a 92x92 binary image containing a rocket–like object in a specific position, with $T = 511 -$

511/4 (a 25% tolerance is allowed), both programs allowed similar accuracy for the determination of the position of the sought object. The DDNN structure resulted about 40% faster than the perceptron alternative.

7 Concluding Remarks

The principal objective of this paper has been to present three distance discriminator neurons based on distance metrics and to point out their potential as effective building elements in homogenuous and hybrid neural networks for classification and pattern recognition.

DDNNs are useful because they can often outperform the equivalent perceptron networks in respect to both execution time and/or amount of required hardware elements. The fact that more than one DDNN can often be found which performs the same classification task indicates that these networks have potential for allowing higher programming flexibility and faster convergence than perceptrons. Although the three basic DDNs implement symmetric classification regions, it has been shown that many other, symmetric or asymmetric, hollow or compact, classification regions can be obtained by using logic and hybrid-DDNNs. DDNNs can also be easily extended to accept N-dimensional inputs.

8 References

R. Beale and T. Jackson, *Neural Computing – An Introduction*, Adam Hilger, 1990.

L. da F. Costa and M. B. Sandler, *Neural networks and Hough transform for pattern recognition* Proc. IEE International Conference on Artificial Neural Networks, 81–86, London, UK, Oct. 1989.

L. da F. Costa, *Effective detection of line segments with Hough transform*, PhD Thesis, King's College London, University of London, London, UK, May 1992-a.

L. da F. Costa, *Towards a versatile framework for intermediate-level computer vision*, Proc. IAPR Workshop on Machine Vision Applications 1992, 261–264, Tokyo, Japan, Dec. 1992-b.

L. da F. Costa and M. B. Sandler, *Effective detection of bar segments with Hough transform*, Computer Vision, Graphics and Image Processing (in press), 1993.

D. R. Hush and B. G. Horne, *Progress in supervised neural networks*, IEEE ASSP Magazine, 8–39, Jan. 1993.

R. P. Lippmann, *An introduction to computing with neural nets*, IEEE ASSP Magazine, 4–22, Apr. 1987.

B. Widrow, R. G. Winter and R. A. Baxter, *Layered neural nets for pattern recognition*, IEEE Transactions on Acoustics, Speech and Signal Processing, 36(7):1109–1118, Jul. 1991.