Hyperborders in the Voronoi-Diagram-Based Neural Net for Pattern Classification

Camillo Gentile
Wireless Communications Technologies Group
National Institute of Standards and Technology
Gaithersburg, MD 20899
cgentile@antd.nist.gov

Mario Sznaier
Department of Electrical Engineering
The Pennsylvania State University
University Park, PA 16802
msznaier@frodo.ee.psu.edu

Abstract - We propose a neural network to answer a point query in $\mathcal{R}^n$ partitioned based on the Voronoi diagram. Our novel design offers the potential to reduce both the number of neurons and connection weights of previous designs, employing a cost function which enables a tradeoff between the two to suit a specific implementation. Our simplified structure requires neither delay weights nor complex neurons, while retaining the main advantage of previous designs to furnish precise values for the neurons and connection weights, as opposed to trial and error iterations or ad-hoc parameters.

Keywords - Neural networks, Voronoi diagrams, Pattern classification

I. Introduction

Many problems in robust pattern classification can be reduced to nearest neighbor clustering in $\mathcal{R}^n$. This problem has been given renewed attention lately in the context of machine vision applications such as object recognition and visual feedback [1]-[3]. A critical constraint in these applications is that the problem must be solved in real time and the solution must be robust against perturbations arising for instance from noise or blurring. In this paper we propose a novel three-layer neural network with intralayer feedback to achieve nearest neighbor clustering. The number of neurons and the interconnection weights can be exactly determined a priori from the problem data, eliminating the need for trial and error type iterations or ad-hoc parameters. Moreover, the proposed net compares favorably, both in terms of the total number of neurons and connection weights, with similar networks proposed in the past to solve the same problem.

Given a set of S sample points in $\mathcal{R}^n$, the proposed design is based upon constructing the Voronoi cell of each point [4], [5] (i.e. the region of space containing those points that are closer, based on some norm, to the given point than to the other elements in the sample) and grouping these cells into clusters representing each class of patterns. Note that although each individual cell is convex, convexity may be lost when the cells are grouped, leading to clusters that are not linearly separable. In the original work along these lines [6], this difficulty was solved by representing each cluster as the union of a finite number of convex regions. While this leads to a feedforward structure having at most three layers, it artificially inflates the number of neurons required.

The key realization of a more recent design [7] exploits the fact that if a query point lies within a non-convex cluster, it also lies within a convex polyhedron defined through the intersection of a proper subset of "true" halfspaces associated with the faces of the cluster. This allows for using a two-layer neural network even when the clusters are non-convex, by constructing region-by-region a list of constraints rendered redundant ("induced") by each of its support halfspaces in the following manner: The input excites the true halfspaces of the first layer, which in turn excite the induced halfspaces through intralayer feedback. The second layer performs the intersection of all the excited halfspaces to determine the polyhedral class which contains the point. The network effectively constructs a non-convex version of a hyperplane ("hyperborder") for each class boundary to determine whether the query point lies in its positive or negative "halfregion".

The design proposed in this paper carries the two-layer structure a step further by disassociating a hyperborder from the boundary of a class polyhedron, driven by the fact that two neighboring polyhedra share a common Rather than form a class boundary independently for both classes, effectively accounting for these common boundaries twice, we form a single hyperborder for each common boundary of adjacent classes. Now a class polyhedron can be constructed as the intersection of the support halfregions of these hyperborders through a three-layer network with intralayer feedback. This enables a drastic reduction in the number of connection weights, as well as bypassing the need for multi-valued delay weights and complex neurons in the implementation.

The paper is organized as follows. In Section II we introduce some mathematical preliminaries and precisely state the problem. Section III reviews the designs of the two competing networks briefly mentioned above, and Section IV describes the proposed network in detail. In Section V we discuss the implementation issues of the three-layer network with intralayer feedback. These results are illustrated in Section VI with a simple example, where the proposed network is compared against the other two. Finally, in Section VII we summarize our results and point out directions for further research.

II. Preliminaries

A. Notation and Geometrical Background

A *hyperplane* t is a set of points $\mathbf{p} \in \mathcal{R}^n$ which satisfy the equation

$$\sum_{i=1}^{n+1} w_i \cdot p_i = 0 \tag{1}$$

0-7803-7278-6/02/$10.00 ©2002 IEEE

where $\mathbf{p}$ has coordinates $(p_1, p_2, \ldots, p_n, 1)$ and w_1 through w_{n+1} are constants, or *weights*. t divides $\mathcal{R}^n$ into a positive halfspace $t^+ : \sum_{i=1}^{n+1} w_i \cdot p_i \geq 0$ and a negative halfspace $t^- : \sum_{i=1}^{n+1} w_i \cdot p_i < 0$. In the sequel we will use the notation $\text{int}\{t^+\}$ ($\text{int}\{t^-\}$) to denote the interior of the halfspace t^+ (t^-).

Given S sample points in $\mathcal{R}^n$, to each point we associate the set of points in the space closer to it than to the other elements in the sample. This convex set is known as a *Voronoi cell* of dimension n, and the collection of cells forms the Voronoi diagram of S [4], [5]. In turn, each cell has *f faces* of dimension $n-1$ shared by adjacent cells. A face is the intersection of a halfspace which bounds the polyhedron and the boundary of the polyhedron. If the positive (negative) halfspace of the cell is associated with the face of the cell, the negative (positive) halfspace is associated with the face of the adjacent cell. Sample points in adjacent cells are said to be *neighboring*, thus a sample point has f neighboring points. In turn, each face has a number of *subfaces* of dimension $n-2$ shared by adjacent faces, and so forth.

B. Statement of the problem

The problem that we address in this paper can be precisely stated as:

Problem 1: Given a set S of sample points in $\mathcal{R}^n$, partitioned into m classes, each one represented by a subset S_k, find the class that contains the nearest neighbor to a query point $\mathbf{p} \notin S$.

Proceeding as in [6], this problem can be solved by considering the cells C_k formed by the the (generally non-convex) union of the Voronoi cells of the points in the class C_k and determining which cell contains $\mathbf{p}$. Thus the problem reduces to:

Problem 2: Given m polyhedra, not necessarily convex classes C_i, $C_i \bigcap C_j = \emptyset$, $i \neq j$, $\bigcup C_i = \mathcal{R}^n$, and a query point $\mathbf{p} \in \mathcal{R}^n$, determine which class contains $\mathbf{p}$.

III. Previous Designs

A. Three-layer feedforward network

A convex polyhedron can be defined through the intersection of a finite number of halfspaces. Thus, in the case of convex cells, Problem 2 can be solved using a two-layer neural network as in [6]. The first layer determines whether a query point lies in the positive or negative halfspace of all the hyperplanes which support the convex region. The second layer performs the intersection or AND operation of all the determined halfspaces. If TRUE, the point lies in the polyhedron. If the cells are non-convex, the same approach can be used, by decomposing each non-convex cell C_i into the union or OR operation of convex polyhedrons C_{ij} in the third layer (see [8] for details). However, as pointed out earlier, this approach artificially inflates the number of neurons required, since potentially many of the constraints defined by the halfspaces supporting the regions C_{ij} are redundant.

B. Two-layer network with intralayer feedback

The two-layer feedback network in [7] has a similar structure to the one above, but eliminates the need for redundant constraints by avoiding decomposing the cells C_i. The key idea reveals that if a query point lies within a non-convex polyhedron C_i, it lies within a convex polyhedron C_{ip} defined by the intersection of a proper subset of *true halfspaces* associated with the faces of C_i. Therefore a TRUE intersection can be realized by having this subset *induce* all the remaining halfspaces associated with the cell C_i. In the sequel we present a systematic procedure to accomplish this.

Given a point lying in a halfspace t^+ of a polyhedron P, faces that lie completely in the complement of t^+ are redundant, in the sense that they do not contribute information about whether or not the point lies in the polyhedron. Thus, a true halfspace t^+ induces another halfspace i^+ (induced halfspace) of the same polyhedron if *the corresponding face f of i^+ lies completely in the complement halfspace of t^+ and the complement halfspace of i^+ intersects the polyhedron and t^+, i.e., if* $f \cap \text{int}\{t^+\} = \emptyset$ *and* $P \cap t^+ \cap i^- \neq \emptyset$, where $f \doteq \partial P \cap i^+$ and ∂P denotes the boundary of P. Note that if the first condition holds true but the second condition does not, i^+ is indeed redundant but it need not be induced. This is due to the fact that any point $\mathbf{p} \in P \cap t^+$ renders i^+ TRUE. These simple observations, which in practice amount to cutting off a non-convex part of the polyhedron, form the basis of the proposed procedure.

To illustrate this approach, consider for example the non-convex polyhedron in $\mathcal{R}^3$ shown in Figure 1. (Only six faces of the polyhedron labeled a through b are shown.) Assume that the negative halfspaces of the faces support the polyhedron. A point in b^- (the negative halfspace of b) induces c^- since face c lies completely in b^+. Faces a, e, and f lie completely in b^- and face d lies in both b^- and b^+, therefore a^-, d^-, e^-, and f^- are not induced. A point in c^- induces b^- since face b lies completely in c^+. Note that a^- does not need to be induced, since a point in $P \cap c^-$ automatically renders a^- TRUE, and also that a point in the halfspace of a polyhedron can induce only the halfspaces of that polyhedron.

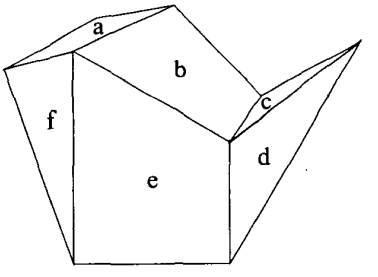


Figure 1. A non-convex polyhedron in $\mathcal{R}^3$ to illustrate true and induced hyperplanes

The design described in this section actually provides a redundant number of induced halfspaces for each true halfspace of a halfregion. A procedure to eliminate this redundancy

is outlined in [7], thereby reducing the number of induction weights presented here to the minimum necessary.

IV. Proposed Design

A. Three-layer network with intralayer feedback

The design in Section III-B, while minimizing the number of neurons in the network, still inflates the number of connection weights. In forming the non-convex boundaries of the class polyhedra through induction, it employs both the positive and negative halfspaces of each boundary hyperplane, necessitating additional weights to excite both halfspaces at different time steps. Moreover this implementation requires multivalued delay weights and complex neurons with positive and negative terminals.

This design can be rendered more efficient by realizing that two neighboring class polyhedra share a non-convex boundary. Rather than form the entire boundary of a class polyhedron as the union of the faces of its support hyperplanes, we form only the common boundary of the neighboring polyhedra in the same manner[1]. We denote these boundaries in the general case as *hyperborders*, since they divide $\mathcal{R}^n$ into positive and negative *halfregions*[2]. Now this common boundary can be accounted in both polyhedra by forming only the positive halfregion which supports one of them, and whose complement (negative) halfregion supports the other. This enables a class polyhedron to be represented as the intersection of its support halfregions. The positive halfregion is defined as the one of the two which requires less induction weights to realize, and the weight reduction follows from the absence of induction weights required to form the negative halfregion.

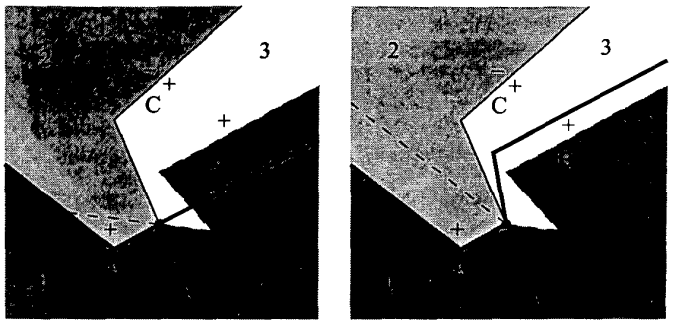
The three-layer network requires four clock steps to answer a point query: At the first step, the input excites the true halfspaces of the network, which in turn excite the induced halfspaces at the following step. The network excites the halfregions at the third step (i.e. performs the intersection of the excited halfspaces to determine whether the point lies in the positive or negative halfregions of the network), and on the fourth step performs the intersection of the excited halfregions to determine the polyhedral class which contains the query point.

B. Artificial Hyperplanes

A *fundamental hyperborder* is one composed solely of the faces whose support hyperplanes separate two class polyhedra. We define the subfaces where fundamental hyperborders meet as the *fundamental subfaces* of the Voronoi diagram. Figure 2(a) displays the fundamental hyperborders A, B, and C (and associated fundamental halfregions) which separate the shaded regions 1, 2, and 3 of a partitioning in $\mathcal{R}^2$. The partitioning contains a single fundamental subface.

[1] In both cases, faces which share only one subface with other faces in the union terminate at infinity.

[2] If the boundaries are convex (i.e. contain just one face each), the hyperborders reduces to hyperplanes and the halfregions to halfspaces.



(a) Fundamental hyperborders (b) Extended fundamental hyperborders

Figure 2. Artificial hyperplanes

Definition 1: Feasibility condition: A hyperborder which separates neighboring class polyhedra cannot intersect any of them.

A class polyhedron can be formed through the intersection of the fundamental halfregions which support it, provided that they meet the feasibility condition. Hyperborder A, which separates regions 1 and 2, violates this condition by its intersection with region 1. Likewise B, which separates regions 1 and 3, results infeasible since it intersects region 1. C, which separates regions 2 and 3, does not violate the feasibility condition.

Joining additional faces to those faces of an infeasible hyperborder which share only one subface with the others in the union, can render the hyperborder feasible. We denote the corresponding hyperplanes of these additional faces as *artificial*. The two artificial hyperplanes which appear in region 3 of figure 2(b) render the extended fundamental hyperborder A feasible; B requires solely the single artificial hyperplane which appears in region 2 to render it feasible. Computing the minimum number of artificial hyperplanes required to establish the feasibilty of a hyperborder falls into the realm of visibility problems [9]. The artificial hyperplanes are not unique, and are chosen here not to coincide with the boundary hyperplanes. The cited technique, however, does furnish unique artificial hyperplanes with minimum cardinality. Now that the three hyperborders in the network are feasible, region 1 is constructed as $A^- \bigcap B^-$, region 2 as $A^+ \bigcap C^-$, and region 3 as $B^+ \bigcap C^+$.

C. Optimal Network Design

While the class polyhedra can be formed through the intersection of the extended fundamental halfregions which support them, this may not lead to the most efficient design in terms of neurons. The design lies in the selection of network hyperborders which minimize a cost amongst a list of *candidate hyperborders*, each one bearing an associated cost in the number of neurons required to form it. A candidate hyperborder is a feasible hyperborder composed of both boundary and artificial hyperplanes, each hyperplane contributing one neuron to the network. Each of these hyperborders, independent of the hyperplanes which form it, also contributes one neuron to the network. Section V provides complete details of the network

structure.

Definition 2: Coverage condition: The union of the hyperborders in the network must cover all the fundamental hyperborders.

The coverage condition ensures that all the regions of the pattern classification problem can be uniquely determined through the intersection of the network hyperborders. We seek the design which minimizes the number of neurons required in its implementation, provided that it meets the coverage condition.

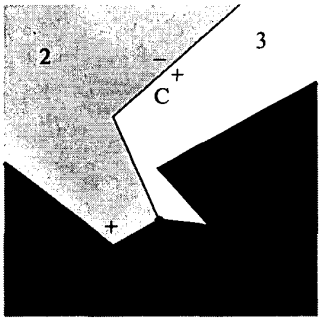


Figure 3. Optimal network design

The network design in figure 2(b) requires a total of 13 neurons for implementation: Hyperborder A require 5 neurons (2 boundary, 2 artificial, plus 1), B requires 5 neurons (3 boundary, 1 artificial, plus 1), and hyperborder C requires 3 neurons (2 boundary, 0 artificial, plus 1). Consider the alternative design in figure 3, where the fundamental hyperborders A and B are joined into a single hyperborder T. Hyperborder T requires 6 neurons (5 boundary, 0 artificial, plus 1) and hyperborder C again requires 3 neurons. This optimal design requires only 9 neurons, where region 1 is formed as T^-, region 2 as $T^+ \bigcap C^-$, and region 3 as $T^+ \bigcap C^+$.

The optimal network design translates to solving equation (2) as an *integer* program, where hyperborder T_j of N candidates has associated cost b_j in number of neurons. Candidate hyperborders consist of a number of simply-connected fundamental hyperborders of the network. The hyperborders can form one loop (i.e. a polyhedron with finite space) originating at a fundamental subface and terminating at that same subface. In this case, the positive halfregion is the interior of the polyhedron and the negative halfregion is its exterior. Details of extracting the list of candidate hyperborders for a specific pattern classification problem appear in [10]. The complexity of the algorithm grows as the cube of the number of fundamental subfaces in the Voronoi diagram, a number which varies according to the number of classes in the problem, not as a function of the individual boundary hyperplanes. Therefore the complexity is essentially independent of the number of sample points. The objective function of (2) reflects the number of neurons in the network, and the constraints, equal in number to the fundamental hyperborders $\widehat{T}_i$, impose the coverage condition. The variable a_{ij} assumes value 1 if $\widehat{T}_i$ makes up T_j, and 0 otherwise.

$$\begin{aligned} &minimize \quad \sum_{j=1}^{N} b_j \cdot T_j \\ &subject\ to \quad \sum_{j=1}^{N} a_{ij} \cdot T_j \geq 1, \ \forall \widehat{T}_i \\ &\qquad\qquad T_j \geq 0 \end{aligned} \tag{2}$$

Rather than engaging the primary problem of (2), we can efficiently solve its dual problem as a *linear* program. Details are provided in [11]. The dual linear program yields the hyperborders of the network which incur the least cost in terms of the neurons required to implement it. Up to this point we have considered only reducing the number of neurons for network realization independent of the required weights, however the objective function of (2) could be easily altered to minimize the number of weights instead, or in the more general case a weighted sum of the two parameters.

Note that we need only consider the fundamental hyperborders as the building blocks of the candidate hyperborders in the network design, as opposed to the individual boundary hyperplanes. By contradiction, if we select a single boundary hyperplane as a hyperborder, feasibilty can be guaranteed only by joining to it the remaining hyperplanes of the common boundary between the two class polyhedra as artificial hyperplanes, such that the hyperborder does not terminate in either of the two classes, in essence creating a fundamental hyperborder anyway.

The total complexity of the optimal design, both in generating the candidate hyperborders and solving the linear program, may be prohibitive for applications which require discrimination amongst a very large number of classes. Here we outline briefly a hierarchical approach to reduce this complexity, however yielding a sub-optimal solution: The first step groups the classes of Problem 2 into a number of disjoint *superclasses*, and determines the optimal set of inter-superclass hyperborders by solving the integer program (2) with only the hyperborders which separate the superclasses as candidates. The second step determines the optimal set of intra-superclass hyperborders for each superclass by solving the integer program with only the hyperborders which separate the classes within the superclass as candidates.

V. The Neural Network

This section details the implementation of the proposed three-layer network with intralayer feedback. As in the previous designs, we employ the McCulloch-Pitts network model [12], [13] which assumes a delay of one unit for a signal to propagate along a weight from its originating terminal to the input of the neuron to which it is connected, and fire. We consider here the bounded input case, common in many pattern classification problems, where it is known a priori that the distance from any admissible query point to any one of the boundary hyperplanes is less than a positive constant M. Details of

the unbounded input case can be found in [7].

The synchronous network accepts the coordinates of the query point $\mathbf{p}$ as the input with $n+1$ terminals. The first layer contains one neuron for each hyperplane (both boundary and artificial) which forms the hyperborders of the network[3]. The interlayer connection weights from the input to the first layer establish the halfspaces of the hyperplanes which support the positive halfregions of the network. These weights are determined as in [6] up to one degree of freedom, and we choose the support halfspaces as positive[4]. The input excites the true halfspaces of the network at the first clock step, and they in turn excite their associated halfspaces at the second step through *intralayer* weights. If the input $\mathbf{p}$ is again applied at the second step, exciting the same true halfspaces as at the previous step, all the halfspaces which support the positive halfregions in which the point lies will be excited at the second step, in conjunction with the induced halfspaces. The neurons in the first layer employ the unipolar hardlimiter as a transfer function, yielding the value 1 if the point lies in its positive halfspace (i.e. excited) and 0 otherwise. The intralayer weight from a true halfspace to the induced halfspace is M: a value large enough to override that from the input through the interconnection weights in the case where the induced halfspace is not also a true halfspace.

The second layer contains one neuron for each hyperborder in the network, to determine whether the query point lies in its positive or negative halfregion by performing the intersection of the excited halfspaces of the first layer which support it. If the intersection is TRUE, the neuron yields the value 1 indicating the positive halfregion, and -1 otherwise. The connection weight from the first layer is 1 if the halfspace supports the halfregion and no connection otherwise. The neurons in the second layer employ the bipolar hardlimiter as a transfer function and their bias weights reflect the number of halfspaces which support the halfregion.

The third layer determines the class polyhedron in which the query point lies, and so contains one neuron for each class. The connection weight from the second layer is 1 if the positive halfregion supports the class polyhedra, -1 if the negative halfspace supports it, and no connection otherwise. The neurons in the third layer employ the unipolar hardlimiter as a transfer function. Their bias weight reflects the number of halfregions which support the polyhedra, effectively performing their intersection, yielding a 1 if the point lies in it and 0 otherwise.

Figure 4 illustrates the network realization of the optimal design of Figure 3: t_1 through t_5 indicate the hyperplanes of T from left to right, and c_1 through c_2 indicate those of C from top to bottom. The intralayer weights of the first layer identify the mutual inductions between t_2^+ and t_3^+ and between t_4^+ and t_5^+. Note that since the halfregion T^- supports the entire boundary of class 1, the corresponding neuron in the second layer serves as the class discriminator as well, requiring one less neuron in the third layer and as a result two less weights: the connection weight from the second layer to the neuron in the third layer, and its associated bias weight.

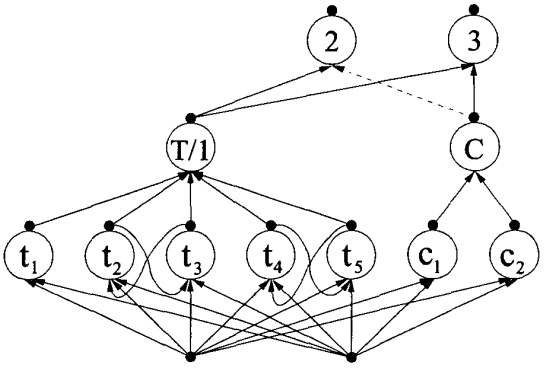


Figure 4. Network realization of the optimal design in Figure 3

VI. A Simple Example

Consider the Voronoi diagram shown in Figure 5 used as an example in both Garga & Bose 1994 [8] and Gentile & Sznaier 2001 [7], constructed from 15 random sample points and clustered into three classes. The figure shows the three fundamental hyperborders A, B, and C terminating at the single fundamental subface to avoid clutter. Note that none of them require artificial hyperplanes.

Table I contains the costs, both in terms of neurons (b_j^n) and weights (b_j^w), that each hyperborder adds to the network. The initial simplex table for the example using the neuron costs appears in Table II.

TABLE I

CANDIDATE HYPERBORDERS AND ASSOCIATED COSTS

T_j	A	B	C	AB	AC	BC
b_j^n	7	9	3	14	8	10
b_j^w	35	42	11	73	44	55

TABLE II

INITIAL SIMPLEX TABLE

	T_1	T_2	T_3	T_4	T_5	T_6	
A	1	0	0	1	1	0	1
B	0	1	0	1	0	1	1
C	0	0	1	0	1	1	1
	7	9	3	14	8	10	

Since the three solutions $[T_1\ T_2\ T_3\ T_4\ T_5\ T_6]$= [1 0 0 0 0 1], [0 1 0 0 1 0], and [0 0 1 1 0 0] all yield the minimum neuron cost of 17, we use the weight cost as a discriminant. The final solution [0 0 1 1 0 0] gives the structure with the minimum weight cost of 84[5]. See [7] for details on the induction weights

[3]Certain hyperplanes may contribute to more than one hyperborder, and each should have one neuron in the first layer.

[4]The negative halfspaces have no application in the network.

[5]These costs do not account for the additional three neurons and three associated bias weights in the third layer.

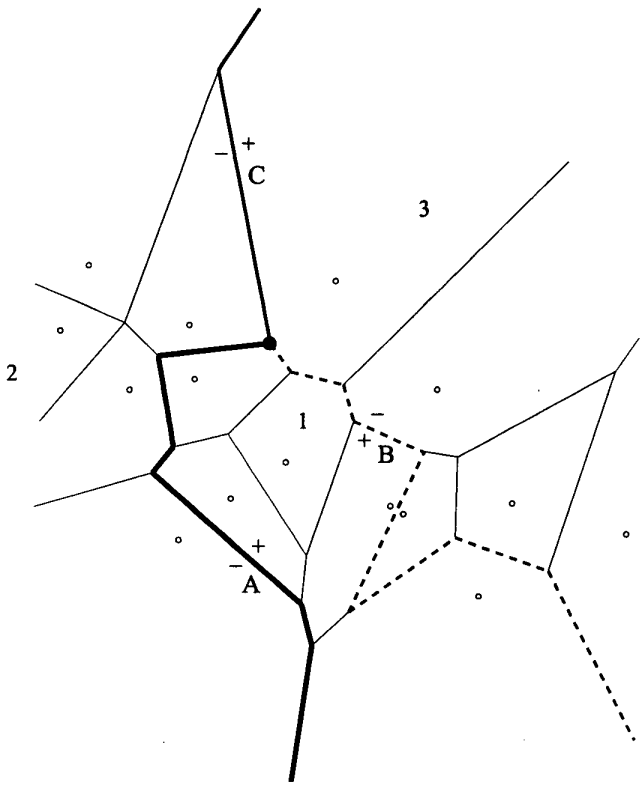


Figure 5. Voronoi diagram for the simple example

for this example.

Table III compares the proposed network to the ones described in Section III both in terms of neurons and weights required for implementation. Our network incurs a 40% decrease in neurons and a 13% decrease in weights over Garga and Bose 1994, and a 42% decrease in weights at the expense of just one neuron over Gentile and Sznaier 2001. In addition our network necessitates neither delay weights nor complex neurons, and answers a point query in one less clock step.

TABLE III

COMPARING NETWORK DESIGNS

	neurons	weights
Garga & Bose 1994	28	98
Gentile & Sznaier 2001	19	123
Gentile & Sznaier 2002	20	87

VII. CONCLUSION

The problem of designing a classifier capable of fast, robust pattern classification has received renewed interest lately in the context of active vision applications. Reducing the size of a network proves critical in real-world applications, where the number of sample points in an image database can easily reach into the millions [7].

In this paper, we propose to solve this problem by using a three-layer neural network with intralayer feedback motivated by computational methods, to answer a point query in $\mathcal{R}^n$ partitioned based on the Voronoi diagram. The paper outlines the design of our novel network which offers the potential to reduce both the number of neurons and connection weights of previous designs, as demonstrated in Section VI, employing a cost function which enables a tradeoff between the two parameters to suit a specific implementation. Our simplified structure requires neither delay weights nor complex neurons, while retaining the main advantage of the previous designs to furnish precise values for the neurons and connection weights, as opposed to trial and error iterations or ad-hoc parameters.

The design in Section III-B generates a structure with the minimum number of neurons required for implementation, while the designs in Sections III-A and IV-A generate networks with less connection weights at the price of increased neurons. We are presently studying methods to compute the minimum number of weights required for implementation. This study takes a closer look at generating the artificial hyperplanes in Section IV-B that currently only guarantees minimum cardinality of the additional neurons, but not of the additional induction weights, which proves a much more challenging problem. In addition we would like to perform a study of the hierarchical approach outlined in section IV-C to quantify the loss in optimality when applied in terms of the number of superclasses, and their structure.

REFERENCES

[1] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, Vol. 3:1, pp. 71-86, 1991.

[2] C.Y. Huang, O.I. Camps, T. Kanungo, "Object Recognition Using Appearance-Based Parts and Relations", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 877-883, 1997.

[3] M.J. Black and A.D. Jepson. "Eigentracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," European Conference on Computer Vision, Vol. 1064, pp. 329-358, 1996.

[4] K. Mulmuley, "Computational Geometry: An Introduction Through Randomized Algorithms," Prentice Hall, 1994.

[5] F.P. Preparata and M.I. Shamos. "Computational Geometry," Springer-Verlag New York, Inc., 1985.

[6] N.K. Bose and A.K. Garga, "Neural Network Design Using Voronoi Diagrams," *IEEE Transactions on Neural Networks*, Vol. 4:5, pp. 778-787, 1993.

[7] C. Gentile and M. Sznaier, "An Improved Voronoi-Diagram-Based Neural Net for Pattern Classification," *IEEE Transactions on Neural Networks*, Vol. 12:5, pp. 1227-1234, 2001.

[8] A.K. Garga and N.K. Bose, "Structure Training of Neural Networks," *IEEE International Conference on Neural Networks*, pp. 239-244, 1994.

[9] C.L. Shih, T.T. Lee, and W.A. Gruver, "Motion Planning with Time-Varying Polyhedral Obstacles Based on Graph Search and Mathematical Programming," *Robotics and Automation*, Vol. 1, pp. 331-337, 1990.

[10] M. A. Samad, "An Efficient Method for Terminal and Multiterminal Pathset Enumeration," *Microelectronics and Reliability*, Vol. 27:3, pp. 443-446, 1987.

[11] D. DeVleeschauwer, "An Intensity-Based, Coarse-to-Fine Approach to Reliably Measure Binocular Disparity," *CVGIP: Image Understanding*, Vol. 57:2, pp. 204-218, 1992.

[12] W.S. McCulloch and W.A. Pitts, "A Logical Calculus of The Ideas Immanent in Neural Nets," *Bull. Math. Biophysics*, Vol. 5, pp. 115-133, 1943.

[13] N.K. Bose and P. Liang, "Neural Network Fundamentals," McGraw-Hill, Inc., 1996.