# An Improved Voronoi-Diagram-Based Neural Net for Pattern Classification

Camillo Gentile and Mario Sznaier

***Abstract*—In this brief paper, we propose a novel two-layer neural network to answer a point query in $\mathcal{R}^n$ which is partitioned into polyhedral regions. Such a task solves among others nearest neighbor clustering. As in previous approaches to the problem, our design is based on the use of Voronoi diagrams. However our approach results in substantial reduction of the number of neurons, completely eliminating the second layer, at the price of requiring only two additional clock steps. In addition, the design process is also simplified while retaining the main advantage of the approach, namely its ability to furnish precise values for the number of neurons and the connection weights necessitating neither trial and error type iterations nor *ad hoc* parameters.**

***Index Terms*—Pattern classification, Voronoi diagrams.**

## I. Introduction

MANY problems in robust pattern classification can be reduced to nearest neighbor clustering in $R^n$. This problem has been given renewed attention lately in the context of machine vision applications such as object recognition and visual feedback [1], [6], [7], [10], [11]. 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 two-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 layers, and in the overall complexity of the design algorithm, with similar networks proposed in the past to solve the same problem.

Given a set of $S$ sample points in $R^n$, the proposed design is based upon constructing the Voronoi cell of each point [2] (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 previous work along these lines, this difficulty was solved by representing each cluster as the union of a finite number of convex regions. While this leads to a structure having at most three layers, it artificially inflates the number of neurons required. The key realization of the proposed method is the fact that if a query point lies within a nonconvex cluster, it lies within a convex polyhedron defined through the intersection of a *proper subset* of "*true*" *halfspaces* associated with the faces of the cluster. This fact allows for using a two-layer neural net *even when the clusters are nonconvex*, by constructing region-by-region a list of constraints rendered redundant ("*induced*") by each of its support halfspaces, and using five clock steps as follows: At the first step the input excites the true halfspaces, which in turn excite the induced halfspaces in latter clock steps. At the fifth step the network performs the intersection of the excited halfspaces to determine the polyhedral class that contains the query point.

This paper is organized as follows. In Section II, we introduce some mathematical preliminaries and precisely state the problem. In Section III, we present an algorithm that allows for reducing the number of constraints required to represent each class. In Section IV, we show how to implement these constraints using a two-layer neural network. These results are illustrated in Section V with a simple example, where the proposed network is compared against similar networks proposed in the past, and in Section VI with a practical application arising in the context of object recognition. Finally, in Section VII, we summarize our results and point out directions for further research.

Manuscript received May 23, 2000; revised March 14, 2001.

C. Gentile is with the Wireless Communications Technologies Group, National Institute of Standards and Technology, Gaithersburg, MD 20899 USA (e-mail: camillo@antd.nist.gov).

M. Sznaier is with the Department of Electrical Engineering, The Pennsylvania State University, University Park, PA 16802 USA (e-mail: msznaier@gandalf.ee.psu.edu).

Publisher Item Identifier S 1045-9227(01)07569-5.

## II. Preliminaries

### A. Notation and Geometrical Background

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

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

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 $t^+$: $\sum_{i=1}^{n+1} w_i p_i \geq 0$ and a negative halfspace $t^-$: $\sum_{i=1}^{n+1} w_i p_i < 0$. In the sequel we will use the notation $\mathrm{int}\{t^+\}$ ($\mathrm{int}\{t^-\}$) 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 $\mathcal{R}^n$ closer to it than to the other elements of the sample. This convex set is known as a Voronoi cell, and the collection of cells forms the Voronoi diagram of $S$ [8]. Each cell has $f$ *faces*, shared by an adjacent cell. 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.

Finally, $(M, T_d)$ denotes an interconnection weight $M$ between two neurons, with a time-delay of $T_d$ time steps.

### B. Statement of the Problem

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

*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 given point $P \notin S$.

Proceeding as in [2], this problem can be solved by considering the cells $C_k$ formed by the (generally nonconvex) union of the Voronoi cells of the points in the class $C_k$ and determining which cell contains the given point $P$. Thus the problem reduces to

*Problem 2:* Given $m$ polyhedral, not necessarily convex classes $C_i$, $C_i \cap C_j = \emptyset$, $i \neq j$, $\cup C_i = \mathcal{R}^n$, and a test point $P \in \mathcal{R}^n$, determine which class contains $P$.

## III. Proposed Approach

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 [2]. 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 nonconvex, the same approach can be used, by decomposing each nonconvex cell $C_i$ into the union of convex polyhedrons $C_{ij}$ (see [2] 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.

The proposed two-layer neural network has a similar structure, but eliminates the need for redundant constraints, by avoiding decomposing the cells $C_i$. The key idea is the fact that, if a query point lies within a nonconvex polyhedron $C_i$, it lies within a convex polyhedron $C_{ip}$ defined by the intersection of a proper *subset* of 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.

### A. Reducing the Number of Hyperplanes

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 $\partial P$ 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 $x \in P \cap t^+$ renders $i^+$ TRUE. These simple observations, which in practice amount to cutting off a nonconvex part of the polyhedron, form the basis of the proposed procedure.

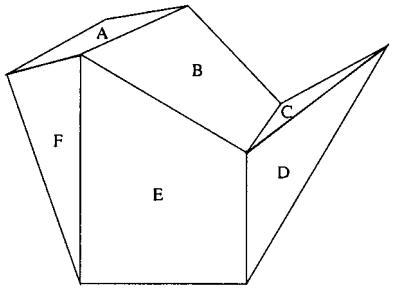


Fig. 1. A nonconvex polyhedron in $R^3$ to illustrate the algorithm for reducing the number of hyperplanes.

To illustrate this approach, consider for example the nonconvex polyhedron in $R^3$ shown in Fig. 1. (Only six faces of the polyhedron labeled $A$ through $F$ are shown.) Assume that the negative halfspaces of the faces form 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.

Note that a point in the halfspace of a polyhedron can induce only the halfspaces of that polyhedron.

### B. Reduction of Induction Weights by Pruning

The number of induction weights found in Section III-A can be reduced by eliminating redundant constraints as follows. Let $I_k$ denote the set of induced halfspaces of class $C_k$. For each element $i_j \in I_k$ we generate a list $T_{kj}$ of the true halfspaces, i.e., the halfspaces of $C_k$ which induce $i_j$. In the pruning step redundant constraints are eliminated from $T_{kj}$.

Take as an example the infinite nonconvex region in $R^2$ to the left of the boundary in Fig. 2. This region is formed by the positive halfspaces of faces $A$, $B$, $C$, $D$, and $E$. In the pruning step we find that $A^+$ is induced by both $B^+$ and $D^+$. Thus if the query point lies in the intersection of $A^-$ and the polyhedron, $A^+$ must be induced by $B^+$ or $D^+$. If we examine however the region common to the polyhedron and $A^-$ we find that it is completely covered by $B^+$. Thus $B^+$ is sufficient to induce $A^+$ and the intralayer weight from $D^+$ to $A^+$ is redundant and can be discarded.

In summary, in the pruning step, for each induced halfspace $i_j \in I_k$ we group in $T_{kj}$ all the halfspaces that induce it. Among this group of halfspaces we retain only those whose union completely covers the intersection of the polyhedron and the complement halfspace of the induced halfspace.

## IV. The Neural Network

In this section, we show how to exploit the ideas of Section III to construct a neural network that solves Problem 2. Several implementations are possible. Later, we discuss two in detail and briefly comment on a third.

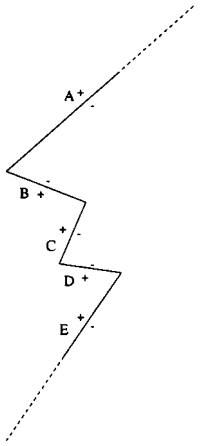


Fig. 2. A nonconvex polyhedron in $R^2$ to illustrate the pruning step.

### A. The Bounded Input Case

Consider first the case, common in many pattern recognition applications, where is it known *a priori* that the distance from any admissible query point to any one of the boundary hyperplanes is less than a positive constant $M$. In this case Problem 2 can be solved using a synchronous network that requires five clock steps to generate the desired output. At the first step the input excites the true halfspaces, which in turn excite the induced halfspaces in latter clock steps. At the fifth step, the network performs the intersection of the excited halfspaces to determine the class of the polyhedron which bounds the query point.

*1) Inter- and Intra-Layer Weight Selection:* Upon construction of the Voronoi diagram for the $S$ sample points we need only compile the following lists:

- the $(n+1)$ parameters for each boundary hyperplane;
- the boundary halfspaces which support the polyhedron for each class;
- the induced halfspaces and the corresponding inducing halfspaces.

The network has $(n+1)$ inputs (including the fixed biases): the coordinates $(p_1, p_2, \ldots, p_n, 1)$ of the query point $\mathbf{p}$. The first layer of the network has one neuron for each boundary hyperplane. We determine the interlayer connection weights between the network inputs and the first-layer neurons as in [2]. These weights establish the positive and negative halfspaces of the hyperplanes.

The intralayer weight in the first layer between the positive (negative) terminal of neuron $j$ and the input of neuron $k$ is $(M, 2)$ (i.e., scalar value of $M$ and a two-unit delay) if the positive (negative) halfspace of neuron $j$ induces the *positive* halfspace of neuron $k$, $(M, 3)$ if the positive (negative) halfspace of neuron $j$ induces the *negative* halfspace of neuron $k$, and zero otherwise. Fig. 3 illustrates the design for a simple example. Here a solid weight indicates a one-unit delay, a dashed weight a two-unit delay, and a boldfaced weight a three-unit delay.

At the first clock step, we present the query point $\mathbf{p}$ as the input to the network. The neurons in the first layer fire to excite the true halfspaces at this step, which in turn excite the positive and negative induced halfspaces. A potential problem with this approach is that a neuron in the first layer can receive both positive and negative inputs. For instance, the point $x_1$ in Fig. 3(a) renders the halfspaces $A^-$ and $B^+$ TRUE. Since $B^+$ induces $A^+$, the neuron $A$ receives both positive and negative inductions. To avoid this situation, positive and negative induction must occur at separate clock steps. To this end the network employs two and three-unit delays to excite the positive and negative induced halfspaces at the third and fourth clock steps, respectively.[1]

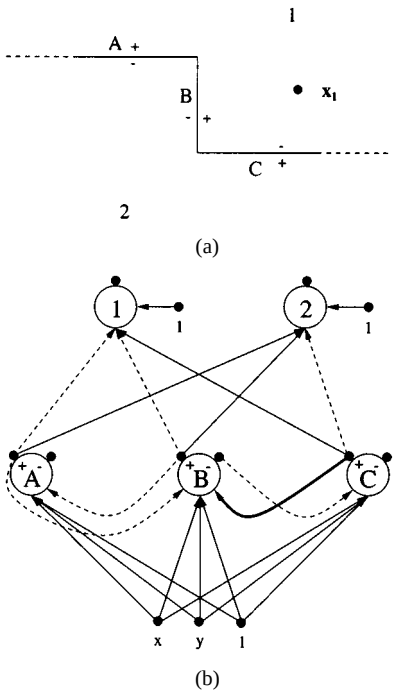


Fig. 3. (a) A simple example (b) The corresponding two-layer network.

At the third clock step, we again present $\mathbf{p}$ as the input to the network to excite the true halfspaces. When the neurons in the first layer fire at the third clock step, there are four possibilities at the input of a neuron in the first layer:

1) The input excites the true positive halfspace *and* the intralayer feedback excites the induced positive halfspace.
2) The input excites the true positive halfspace *and* the intralayer feedback does not excite the induced positive halfspace.
3) The input excites the true negative halfspace *and* the intralayer feedback excites the induced positive halfspace.
4) The input excites the true negative halfspace *and* the intralayer feedback does not excite the induced positive halfspace.

Note that since the intralayer connection has a weight of $M$ and the input is always less than $M$, the first three cases results in the positive terminal of the neuron being excited, with its output reaching the second layer at the fifth time step. In the fourth case, the negative terminal of the neurons is excited but has no connection to the second layer.

[1]The unit delays for induction must be relatively prime numbers so that the inductions do not interfere with each other. Thus, we have selected these delays to be two and three units, respectively. As a result all the neurons assume a quiet state at the second clock step since they receive no input from that of the network and they receive no induction from the intralayer feedback.

TABLE I
STATUS OF THE NEURONS AS A FUNCTION OF TIME FOR THE SIMPLE EXAMPLE

| | neuron terminals | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| step | $A^+$ | $A^-$ | $B^+$ | $B^-$ | $C^+$ | $C^-$ | 1 | 2 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

At the fourth clock step, we present $-\mathbf{p}$ as the input to the network, effectively exciting the true negative halfspace through the true positive halfspace, thus allowing the same reasoning as in the third clock step above. In the first three cases, the positive terminal of the neuron is excited and reaches the second layer at the fifth time step. In the fourth case, the negative terminal of the neurons is excited but has no connection to the second layer. Table I shows the status of the different neurons at each clock step for the simple example used above, when $\mathbf{p} = \mathbf{x_1}$.

The second layer of the network contains $m$ neurons: one neuron per class. The interlayer connection weight between the positive terminal of neuron $j$ in the first layer and the input to neuron $k$ in the second layer is $(1, 2)$ if the positive halfspace of neuron $j$ supports the polyhedron of class $k$, $(1, 1)$ if the negative halfspace of neuron $j$ supports the polyhedron of class $j$, and zero otherwise. Since the positive and negative halfspaces are excited at separate clock steps, the positive halfspaces have an extra delay of one unit with respect to the negative halfspaces in order to synchronize the two such that they reach the second layer at the same fifth step. Each neuron in the second layer also has a negative bias weight equal to the number of halfspaces that support its class. Coupled with a unipolar hardlimiter transfer function, each neuron in the second layer realizes the AND operation of the halfspaces which support its class. We emphasize that there are no connections between the negative terminals of the neurons in the first layer and the inputs to the neurons in the second layer.

### B. The Unbounded Input Case

When the input cannot be bounded *a priori*, a similar effect can be accomplished by using feedback weights (rather than continuous excitation) to excite the true positive (negative) halfspaces in the third (fourth) clock step. As a result we present the query point $\mathbf{p}$ as the input to the network only at the first clock step. Each neuron in the first layer has a feedback weight $(1, 2)$ connected to its input from its positive terminal and a feedback weight $(1, 3)$ connected to its input from its negative terminal. The intralayer weight in the first layer between the positive (negative) terminal of neuron $j$ and the input of neuron $k$ is $(1, 2)$ if the positive (negative) halfspace of neuron $j$ induces the *positive* halfspace of neuron $k$, $(1, 3)$ if the positive (negative) halfspace of neuron $j$ induces the *negative* halfspace of neuron $k$, and zero otherwise.

### C. Implementation Considerations

In neural modeling a weight simulates the axon which interconnects neurons. The speed with which a pulse propagates along an axon varies greatly [3]. A simplistic hardware realization models the axon as a transmission line with a resistor and a capacitor, the resistance adjusting the weight value and the capacitance adjusting the propagation time. Following the work by Bose and Garga, we implement our system through a McCulloch–Pitts network model [4] assuming 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. The additional delays of two and three units are obtained by simply adding two extra capacitors. Thus these delays do not increase the complexity of the system.

The neurons of the network proposed in [2] employ the bipolar hardlimiter as a transfer function. The neurons in the first and second layers necessitate two terminals: a positive terminal (the output of the bipolar hardlimiter) connected to positive weights and a negative terminal (the negative value of the positive terminal) connected to negative weights, thus simulating the negative weights in the physical implementation.

The neurons of the proposed network employ the unipolar hardlimiter as a transfer function. The neurons in the first layer also necessitate two terminals: a positive terminal (the output of the unipolar hardlimiter) to excite the halfspaces induced by the true positive halfspaces and a negative terminal (the logic complement of the positive terminal) to excite the halfspaces induced by the true negative halfspaces. We have only positive weights connected from the output terminals of the neurons.

Finally, it is worth emphasizing the fact that since the implementation of the neural net is not unique, a more suitable implementation may be tailored to the a specific application. For example yet another implementation uses only one terminal for the neurons in the first layer (thus eliminating the logic inverters), however necessitating eight clock steps.

## V. A SIMPLE EXAMPLE

Consider the Voronoi diagram used as an example in [5], shown in Fig. 4. This diagram was generated from 15 random sample points, clustered into three classes. The solid lines show the 16 boundary faces labeled $A$ through $P$ and the plus and minus signs indicate their corresponding positive and negative halfspaces.

Using the rules provided in Section III-A, we compile a list of true halfspaces and corresponding induced halfspaces for a each class in Table II, sorted according to the true halfspaces.

We resort Table II in Table III according to the induced halfspaces. The boldface indicates the pruned induced halfspaces using the rules provided in Section III-B.

Finally, we again resort Table III in Table IV according to the true halfspaces, however without the pruned induced halfspaces, thus providing the minimum set of induced halfspaces for each class.

For this example, the proposed network requires 19 neurons, 123 weights, and two layers as opposed to the network in [5]

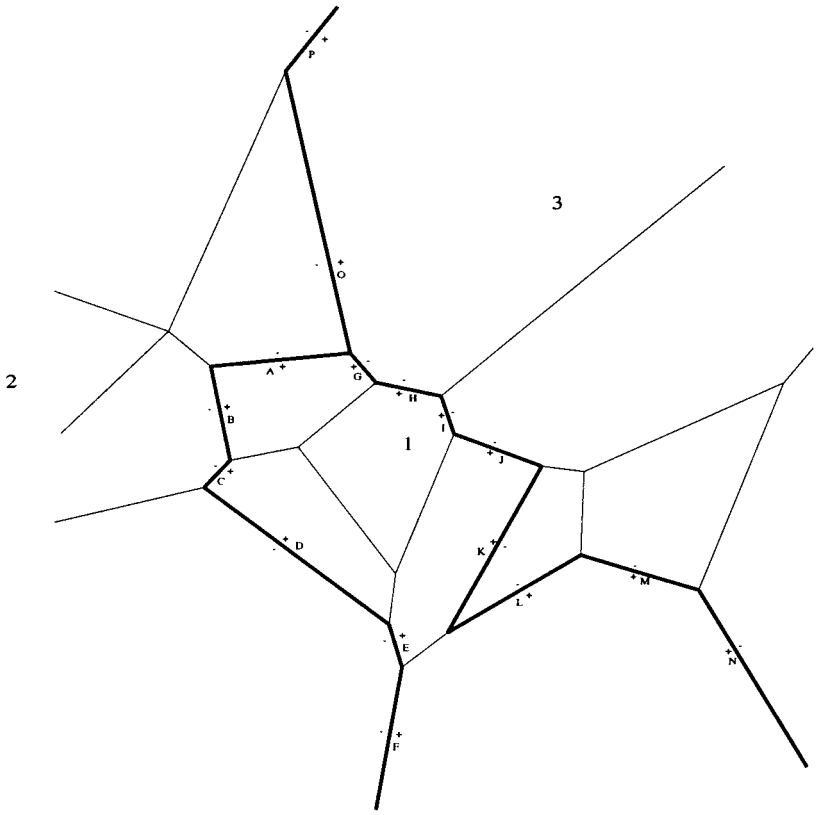


Fig. 4. Voronoi diagram for the simple example.

TABLE II
ORIGINAL INDUCED HALFSPACES VERSUS TRUE HALFSPACES

| Class 1 | Class 2 | Class 3 |
|---|---|---|
| $A^+ \Rightarrow$ | $A^- \Rightarrow B^-, C^-, D^-, E^-, F^-$ | $G^- \Rightarrow$ |
| $B^+ \Rightarrow C^+$ | $B^- \Rightarrow A^-, E^-, O^-, P^-$ | $H^- \Rightarrow I^-, K^-, L^-, N^-$ |
| $C^+ \Rightarrow B^+$ | $C^- \Rightarrow D^-, E^-, F^-$ | $I^- \Rightarrow G^-, H^-$ |
| $D^+ \Rightarrow E^+, F^+$ | $D^- \Rightarrow A^-, B^-, C^-, P^-$ | $J^- \Rightarrow K^-, L^-, M^-, N^-$ |
| $E^+ \Rightarrow B^+, C^+, D^+, F^+$ | $E^- \Rightarrow P^-$ | $K^- \Rightarrow G^-, H^-, I^-, J^-$ |
| $F^+ \Rightarrow B^+, D^+, E^+, G^+$ | $F^- \Rightarrow$ | $L^- \Rightarrow M^-, N^-$ |
| $G^+ \Rightarrow H^+, I^+, J^+, M^+$ | $O^- \Rightarrow P^-$ | $M^- \Rightarrow L^-, N^-$ |
| $H^+ \Rightarrow G^+$ | $P^- \Rightarrow A^-, B^-, C^-, D^-, E^-, F^-, O^-$ | $N^- \Rightarrow G^-, H^-, J^-, K^-, L^-, M^-$ |
| $I^+ \Rightarrow J^+, M^+$ | | $O^+ \Rightarrow$ |
| $J^+ \Rightarrow G^+, I^+$ | | $P^+ \Rightarrow$ |
| $K^+ \Rightarrow L^+, M^+$ | | |
| $L^+ \Rightarrow B^+, D^+, E^+, G^+, I^+, K^+$ | | |
| $M^+ \Rightarrow G^+, I^+$ | | |
| $N^+ \Rightarrow$ | | |

which requires 28 neurons, 98 weights (taking into account the constant bias weights), and three layers. Thus, in this case our approach yields a 47% decrease in the number of neurons, at the price of a 20% increase in the number of weights. In addition, the proposed design process requires solely the computation of the Voronoi diagram, as opposed to the approach in [5] that also requires computing the Delaunay tessellation and the convex hull of each polyhedron. When the nonconvex boundary of the polyhedron becomes complicated, the number of neurons in the second layer of the network proposed in [2] may become increasingly large. Our approach avoids this problem completely by eliminating this second layer, thus allowing for a more efficient implementation of the network.

## VI. A PRACTICAL APPLICATION: OBJECT RECOGNITION

The computational complexity entailed in recognizing objects from a cluttered scene can be substantially reduced by using eigenspace decomposition techniques [7], [11]. The main idea of the method is to use principal component analysis to compress images of objects contained in a precomputed database to just a few ($n$) components which exhibit the largest

TABLE III
TRUE HALFSPACES VERSUS ORIGINAL INDUCED HALFSPACES

| Class 1 | Class 2 | Class 3 |
|---|---|---|
| $A^+ \Leftarrow$ | $A^- \Leftarrow B^-, D^-, \mathbf{P}^-$ | $G^- \Leftarrow \mathbf{I}^-, K^-, \mathbf{N}^-$ |
| $B^+ \Leftarrow C^+, \mathbf{E}^+, \mathbf{F}^+, \mathbf{L}^+$ | $B^- \Leftarrow A^-, D^-, \mathbf{P}^-$ | $H^- \Leftarrow I^-, K^-, \mathbf{N}^-$ |
| $C^+ \Leftarrow B^+, \mathbf{E}^+$ | $C^- \Leftarrow A^-, D^-, \mathbf{P}^-$ | $I^- \Leftarrow H^-, K^-$ |
| $D^+ \Leftarrow E^+, F^+, \mathbf{L}^+$ | $D^- \Leftarrow A^-, C^-, \mathbf{P}^-$ | $J^- \Leftarrow K^-, \mathbf{N}^-$ |
| $E^+ \Leftarrow D^+, F^+, \mathbf{L}^+$ | $E^- \Leftarrow A^-, \mathbf{B}^-, \mathbf{C}^-, \mathbf{P}^-$ | $K^- \Leftarrow \mathbf{H}^-, J^-, \mathbf{N}^-$ |
| $F^+ \Leftarrow D^+, E^+$ | $F^- \Leftarrow \mathbf{A}^-, \mathbf{C}^-, P^-$ | $L^- \Leftarrow \mathbf{H}^-, \mathbf{J}^-, M^-, N^-$ |
| $G^+ \Leftarrow \mathbf{F}^+, H^+, \mathbf{J}^+, \mathbf{L}^+, \mathbf{M}^+$ | $O^- \Leftarrow \mathbf{B}^-, P^-$ | $M^- \Leftarrow \mathbf{J}^-, L^-, N^-$ |
| $H^+ \Leftarrow G^+$ | $P^- \Leftarrow \mathbf{B}^-, \mathbf{D}^-, \mathbf{E}^-, O^-$ | $N^- \Leftarrow \mathbf{H}^-, \mathbf{J}^-, L^-, M^-$ |
| $I^+ \Leftarrow \mathbf{G}^+, J^+, \mathbf{L}^+, \mathbf{M}^+$ | | $O^+ \Leftarrow$ |
| $J^+ \Leftarrow \mathbf{G}^+, I^+$ | | $P^+ \Leftarrow$ |
| $K^+ \Leftarrow L^+$ | | |
| $L^+ \Leftarrow K^+$ | | |
| $M^+ \Leftarrow \mathbf{G}^+, \mathbf{I}^+, K^+$ | | |
| $N^+ \Leftarrow$ | | |

TABLE IV
PRUNED INDUCED HALFSPACES VERSUS TRUE HALFSPACES

| Class 1 | Class 2 | Class 3 |
|---|---|---|
| $A^+ \Rightarrow$ | $A^- \Rightarrow B^-, C^-, D^-, E^-$ | $G^- \Rightarrow$ |
| $B^+ \Rightarrow C^+$ | $B^- \Rightarrow A^-$ | $H^- \Rightarrow I^-$ |
| $C^+ \Rightarrow B^+$ | $C^- \Rightarrow D^-$ | $I^- \Rightarrow H^-$ |
| $D^+ \Rightarrow E^+, F^+$ | $D^- \Rightarrow A^-, B^-, C^-$ | $J^- \Rightarrow K^-$ |
| $E^+ \Rightarrow D^+, F^+$ | $E^- \Rightarrow$ | $K^- \Rightarrow G^-, H^-, I^-, J^-$ |
| $F^+ \Rightarrow D^+, E^+$ | $F^- \Rightarrow$ | $L^- \Rightarrow M^-, N^-$ |
| $G^+ \Rightarrow H^+$ | $O^- \Rightarrow P^-$ | $M^- \Rightarrow L^-, N^-$ |
| $H^+ \Rightarrow G^+$ | $P^- \Rightarrow F^-, O^-$ | $N^- \Rightarrow L^-, M^-$ |
| $I^+ \Rightarrow J^+$ | | $O^+ \Rightarrow$ |
| $J^+ \Rightarrow I^+$ | | $P^+ \Rightarrow$ |
| $K^+ \Rightarrow L^+, M^+$ | | |
| $L^+ \Rightarrow K^+$ | | |
| $M^+ \Rightarrow$ | | |
| $N^+ \Rightarrow$ | | |

eigenvalues. Thus, it eliminates redundancy among the images while preserving their essential features.

Fig. 5 displays five example objects from the database COIL-20 [9]. Since each of these objects may appear at any angle in the test scene, as shown for instance in Fig. 6, the database consists of a representative number of appearances for each object, rather than just the object at a single angle, and each of these appearances maps to a single point in the $n$th-dimensional search space. Object recognition is accomplished by performing, *in real time*, a nearest neighbor search, to find the closest match among the objects in the database. Since this search can be performed by the proposed neural network in five clock steps, it has the potential to perform object recognition at a frame rate (30 Hz or higher), a task that usually cannot be accomplished using serial point query.

For this application, the proposed network can be constructed by considering the Voronoi diagram of all the points in the database and grouping the cells corresponding to each appearance of the same object to create a (generically nonconvex) object *class*. Our example database contains 16 objects and 36 appearances (every 10°) for each object. In this particular case, keeping the eigenvectors corresponding to just the $n = 4$ largest eigenvalues proved sufficient for reliable discrimination between the objects in the database in the presence of noise.

The corresponding neural network contains $X = 1824$ neurons in the first layer (one for each of the bounding hyperplanes) and 16 neurons in the second layer (one for each of the object classes). The number of feedforward weights to the first layer is 9120 ($X \cdot (n+1)$), the number of feedforward weights to the second layer is 3664 ($X \cdot 2 + 16$), and the number of intralayer weights is 5467. The number of neurons and weights required by the VONNET proposed in [2], [5] depends on the number of subclasses required to cover the nonconvex classes. Proposition 1 in [5] yields 36 subclasses per class, for a total of 576 neurons in the second layer, and assuming an average of four faces/subclass (since we are working in $R^4$, compact polytopes have at least five faces), 2880 weights, including the 576 bias weights required by the neurons. Thus, in this example the VONNET requires 2416 neurons and on the order of 13 170 weights (9120, 2880, and 1170 for the first, second, and third layers, respectively). Hence our approach leads to a reduction of approximately 30% in the number of neurons at the price of a 38% increase in the number of weights.

Finally, note in passing that the same approach can be used to deal with occlusion by using *parts* of the object rather than the object itself [6]. The main idea is that, even if a large portion of the object displays occlusion in the scene, it can still be recovered through its visible parts. *Adjacency relations* enhance the reliability of the system, thus taking into account the detection of neighboring parts originating from the same object. Now the database consists of the appearances of parts rather than the appearances of objects. In the implementation in [6], a segmentation algorithm based on homogeneous texture seeks to partition an object into its physical parts. Such a partitioning maps the same parts at different appearances close to each other in the eigenspace, allowing compact nearest neighbor clustering. However, since each part contains homogeneous texture, they tend to be less distinctive than the original objects. This characteristic coupled with the fact that the part database contains


Fig. 5. Five example objects in the database.


Fig. 6. *Duck* appearances.

many more items than the object database[2] requires a significantly higher number of eigenvalues for reliable discrimination, on the order of 100. Thus a part database for 100 objects and 36 appearances for each part in $n = 100$ may contain 18 000 points for a total of 18 000 000 coordinates in a nearest neighbor search. The number of generated boundary hyperplanes in this case proves quite high, so reducing the number of neurons in the hardware implementation becomes an important issue.

## VII. Conclusions and Further Work

The problem of designing a classifier capable of fast, robust pattern classification has received renewed interest lately in the context of active vision applications. In this paper, we proposed to solve this problem using a novel neural network motivated by computational geometry methods. This net shares all the advantages of similar nets proposed in the past (namely its structure, number of neurons, and interconnection weights can be determined *a priori* from the problem data), while using substantially less neurons, at the price of potentially increasing the number of weights. However, as illustrated with both the simple example and the practical application, the number of intralayer weights and feedforward weights to the second layer dwindles with respect to that of the feedforward weights to the first layer. Specifically, in a problem with inputs in $R^n$, $C$ classes and an average number $f$ of faces/class, the total number of weights required by our approach is given by

$$N = \underbrace{C * f}_{\text{output weights}} + \overbrace{p(f) * f * C}^{\text{intralayer weights}} + \underbrace{(1+n) * f * \frac{C}{2}}_{\text{input weights}} \tag{2}$$

where $p(f)$ is the average number of hyperplanes induced per face. While this number is application dependent,[3] it typically increases with $f$ at a much lower rate than linear. Note that the last term in (2) also appears when computing the number of weights in [2]. The latter necessitates additional weights from/to the second and third layers as well (absent in our approach), leading to the following estimate for the number of weights:

$$N_{\text{VONNET}} = \underbrace{C * s(C)}_{\text{output weights}} + \overbrace{s(C) * (n+1) * C}^{\text{second to third layer weights}} + \underbrace{(1+n) * f * \frac{C}{2}}_{\text{input weights}} \tag{3}$$

where $s(C)$ denotes the average number of subclasses per class, and where we have assumed an average of $n$ faces per subclass. Note that the first and second terms in (2) and (3) have the same order of magnitude, and that in higher dimensional applications these terms are usually dominated by the third. It follows that in these cases both networks require roughly the same number of weights.

We are currently working on a network which exhibits the same induction principle of boundary halfspaces to represent nonconvex polyhedra, but requires no delay weights and drastically reduces the number of interconnection weights involved.

## Acknowledgment

The authors are indebted to Prof. N. K. Bose and Dr. A. Garga for discussions on using computational geometry and graph theory to design neural nets for pattern classification, and to N. Pande for providing the data for the practical application.

## References

[1] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," in *Proc. Europ. Conf. Comput. Vision*, vol. 1064, 1996, pp. 329–358.
[2] N. K. Bose and A. K. Garga, "Neural network design using Voronoi diagrams," *IEEE Trans. Neural Networks*, vol. 4, pp. 778–787, Sept. 1993.
[3] N. K. Bose and P. Liang, *Neural Network Fundamentals*. New York: McGraw-Hill, 1996.
[4] W. S. McCulloch and W. A. Pitts, "A logical calculus of the ideas immanent in neural nets," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.
[5] A. K. Garga and N. K. Bose, "Structure training of neural networks," in *IEEE Int. Conf. Neural Networks*, 1994, pp. 239–244.

[2]The segmentation generates on average five parts from each object appearance.

[3]Cases where the cells are highly nonconvex yield larger values of $p$.

[6] C. Y. Huang, O. I. Camps, and T. Kanungo, "Object recognition using appearance-based parts and relations," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 1997, pp. 877–883.
[7] A. Leonardis and H. Bischof, "Dealing with occlusion in the eigenspace approach," in *Proc. IEEE Conf. Comput. Vision. Pattern Recognition*, 1996, pp. 453–458.
[8] K. Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
[9] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (COIL-20),", Tech. Rep. CUCS-005-96, Feb. 1996.
[10] F. de la Torre, S. Gang, and S. McKenna, "View-based adaptive affine tracking," *Lecture Notes Comput. Sci.*, vol. 1406, pp. 828–842, 1998.
[11] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.