A Constructive algorithm to solve “convex recursive deletion” (CoRD) classification problems via two-layer perceptron networks

by
C. Cabrelli[1], U. Molter[1]
Departamento de Matemática, Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires, Ciudad Universitaria, Pabellón I
1428 Buenos Aires, Argentina
and
CONICET, Argentina
email: ccabrell@dm.uba.ar, umolter@dm.uba.ar

and

R. Shonkwiler
School of Mathematics
Georgia Institute of Technology
email: shonkwiler@math.gatech.edu, *web:* www.math.gatech.edu/~shenk

Abstract

A sufficient condition that a region be classifiable by a 2-layer feed-forward neural net (a two-layer Perceptron) using threshold activation functions is that either it be a convex polytope, or *that* intersected with the complement of a convex polytope in its interior, or *that* intersected with the complement of a convex polytope in its interior, or . . . recursively. These have been called Convex Recursive Deletion (CoRD) regions. We give a simple algorithm for finding the weights and thresholds, in both layers, for a feed forward net which implements such a region.

The results of this work help in understanding the relationship between the decision region of a perceptron and its corresponding geometry in input space. Our construction extends in a simple way to the case that the decision region is the disjoint union of CoRD regions (requiring three layers). Therefore this work also helps in understanding how many neurons are needed in the second layer of a general, three layer network. In the event that the decision region of a network is known and is the union of CoRD regions, our results enable the calculation of the weights and thresholds of the implementing network directly and rapidly without the need for thousands of backpropagation iterations.

§1 Introduction

We define an (n-input) *neuron* N as a device capable of

[1] This research was carried out during a six month visit of Cabrelli and Molter at the School of Mathematics, Georgia Institute of technology. These authors acknowledge the support received during their visit. They are alsso partially supported by grants: UBACyT TW084 and CONICET, PID 456/98.

(1) forming the weighted sum $\sigma = \Sigma x_i w^i$ of its inputs, $x_1, \ldots, x_n$, with weights w^1, w^2, …, w^n, and

(2) thresholding the resultant sum with a given value θ, a real number, to produce an output: $y = 0$ if $\sigma < \theta$ or $y = 1$ if $\sigma \geq \theta$.

Mathematically such a neuron evaluates the function $y = u_\theta(\mathbf{x}^t\mathbf{w})$ where $u_\theta(\cdot)$ is the step function (or *hard limiter*) with threshold θ, and $\sigma = \mathbf{x}^t\mathbf{w}$ is the dot product between the input vector $\mathbf{x} = (x_1, \ldots, x_n)^t$ and weight vector $\mathbf{w} = (w^1, \ldots, w^n)^t$.

With each n-input neuron there is associated a unique oriented hyperplane $H_{\mathbf{w},\theta}$ of n-dimensional Euclidean space $\mathbf{R}^n$ given by

$$H_{\mathbf{w},\theta} = \{\mathbf{x} : \mathbf{x}^t\mathbf{w} = \theta\} \tag{1.1}$$

whose *positive* side is in the direction $\mathbf{w}$ and whose distance from the origin is $d = |\theta|/\|\mathbf{w}\|$. $H_{\mathbf{w},\theta}$ decomposes $\mathbf{R}^n$ into two half-spaces, $H^+_{\mathbf{w},\theta} = \{\mathbf{x} : \mathbf{x}^t\mathbf{w} \geq \theta\}$ and $H^-_{\mathbf{w},\theta} = \{\mathbf{x} : \mathbf{x}^t\mathbf{w} < \theta\}$, and is the boundary of both. The hyperplane itself belongs fully to the positive half space. The output of the neuron is 1 for an input $\mathbf{x}$ if and only if $\mathbf{x} \in H^+_{\mathbf{w},\theta}$. Since the solutions $\mathbf{x}$ to the inequality $\mathbf{x}^t\mathbf{w} \geq \theta$ are invariant under its multiplication by a positive constant, that is, for $c > 0$, also $\mathbf{x}^t(c\mathbf{w}) \geq c\theta$, distinct neurons (their weights and thresholds differing by a positive multiple) may be associated with the same half-space. But this fact may be used to advantage by allowing scaling of the weights or threshold as necessary to meet implementation requirements. For example, a threshold might correspond to a voltage in an electric circuit which may not be arbitrarily large. An artificial Neural Network consisting of a single such neuron is known as a *Perceptron*.

We define an (m-neuron) *layer* $\mathcal{L}_n$ of n-input neurons as a list $N_1, \ldots, N_m$ of m neurons defined over the same set of n inputs.

Let neuron N_j have threshold θ_j and weight vector $\mathbf{w}_j = (w^1_j, \ldots, w^n_j)^t$, $j = 1, \ldots, m$, that is w^i_j is the weight connecting the ith input to the jth neuron. Let W be the $n \times m$ matrix whose m columns are the n-vectors $\mathbf{w}_j$. Then the m-dimensional vector of weighted sums $\bar{\sigma}$ is given by the matrix product

$$\bar{\sigma}^t = \mathbf{x}^t W \tag{1.2}$$

and the m-dimensional vector *output* of the layer is given by

$$\mathbf{y} = U_\theta(\mathbf{x}^t W) = (u_{\theta_1}(\mathbf{x}^t\mathbf{w}_1), \ldots, u_{\theta_m}(\mathbf{x}^t\mathbf{w}_m))^t. \tag{1.3}$$

(The last member defines $U_\theta(\cdot)$.) Each component y_j of $\mathbf{y}$ is either 0 or 1 depending on the output of the j^{th} neuron and so the possible outputs are vertices of the m-dimensional unit cube Q_m,

$$Q_m = \{(y_1, \ldots, y_m) : y_j \in \{0,1\},\ 1 \leq j \leq m\}. \tag{1.4}$$

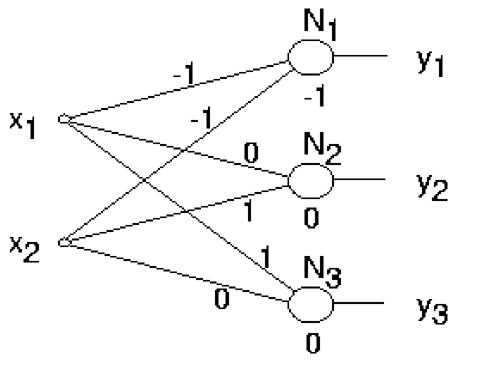


Fig. 1a. Network, $\mathbf{w}_1 = [-1 \quad -1]^t$, $\mathbf{w}_2 = [0 \quad 1]^t$, $\mathbf{w}_3 = [1 \quad 0]^t$, $\theta = [-1 \quad 0 \quad 0]^t$

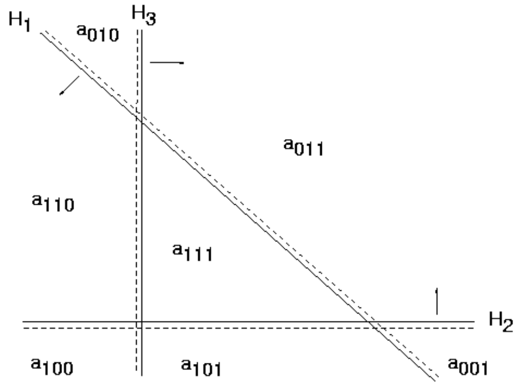


Fig. 1b. Input space and atoms e.g., a_{011} is the intersection of the neg. side of H_1 and pos. sides of H_2, H_3

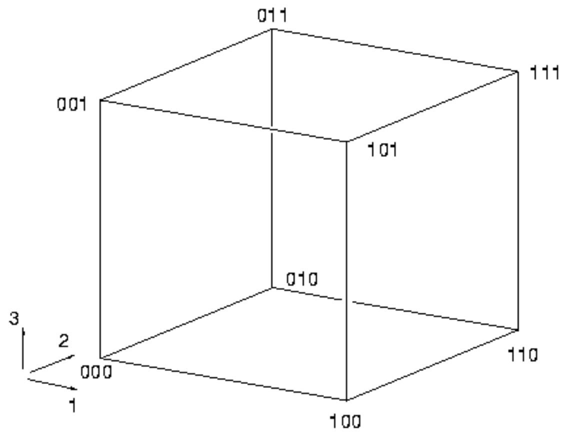


Fig. 1c. 3-cube representation of the output e.g. $y_1 = 1$, $y_2 = y_3 = 0$ is the vertex (100)

For a layer of n-input neurons, define the function $q : \mathbf{R}^n \to Q_m$ by

$$q(\mathbf{x}) = U_\theta(\mathbf{x}^t W). \tag{1.5}$$

Since the m-cube has exactly 2^m vertices, many inputs $\mathbf{x}$ will have the same q value. Given a vertex $\mathbf{y} \in Q_m$, the $\mathbf{y}^{th}$ *atom* or *cell* $a_\mathbf{y} \subset \mathbf{R}^n$ is the inverse image

$$a_\mathbf{y} = \{\mathbf{x} \in \mathbf{R}^n : q(\mathbf{x}) = \mathbf{y}\}. \tag{1.6}$$

Each such atom is the intersection of half-spaces,

$$a_\mathbf{y} = \bigcap_{j=1}^{m} H^{\epsilon_j}_{\mathbf{w}_j,\theta_j}, \tag{1.7}$$

where $\epsilon_j = +$, if $y_j = 1$, and $\epsilon_j = -$, if $y_j = 0$. Therefore each atom is a convex polytope or the empty set. The set of all (non-empty) atoms $\{a_\mathbf{y} : \mathbf{y} \in Q_m\}$ forms a partition of input space $\mathbf{R}^n$ into mutually exclusive, exhaustive convex polytopes. (A convex set is one for which, given any two points contained in it, then their joining line segment is also in it. A convex polytope is a convex set whose boundary consists of a finite number of hyperplanes. The entire space qualifies since it is convex and its boundary is empty. A set which is the intersection of a finite number of half-spaces is convex because whenever two points lie on one side of a half-space, then their joining line seqment does also.) Atoms may be empty, bounded, unbounded, open, closed, or contain only part of their boundary. In general there must necessarily be empty atoms. Fig. 1 illustrates these concepts for a 2-input 3-neuron example.

Now let the output $\mathbf{y}$ of the layer $\mathcal{L}^1_n$ be taken as the input to a second layer $\mathcal{L}^2_m$ consisting of a single m-input neuron O with weight matrix (vector) V and threshold η. We will refer to such a two layer feed forward net as a *Two-layer Perceptron.* As above, O corresponds to an m-dimensional hyperplane $K_{V,\eta}$,

$$K_{V,\eta} = \{\mathbf{y} : \mathbf{y}^t V = \eta\}, \tag{1.8}$$

$$K^+_{V,\eta} = \{\mathbf{y} : \mathbf{y}^t V > \eta\}, \qquad K^-_{V,\eta} = \{\mathbf{y} : \mathbf{y}^t V < \eta\}, \tag{1.9}$$

which exists in cube-space along with Q_m. This *cube-plane*, as we shall call it, may intersect Q_m. In such an event, the vertices of the cube are partitioned into two disjoint sets,

$$F = K^+_{V,\eta} \cap Q_m, \qquad \text{and} \qquad G = K^-_{V,\eta} \cap Q_m. \tag{1.10}$$

In turn, the set of vertices F correspond to a set of atomic convex regions of input space; let

$$\mathcal{F} = \bigcup_{\mathbf{y} \in F} a_{\mathbf{y}} \subset \mathbf{R}^n \tag{1.11}$$

be their union, see Fig. 2. Note that in general there are vertices that correspond to no actual atom, these are "don't care" vertices. For example, only 7 atoms, at most, result from the intersection of 3 lines in the Euclidean plane while there are 8 vertices of the 3-cube. Using the same labeling as in fig. 1, vertex E in the figure is a "don't care" vertex.

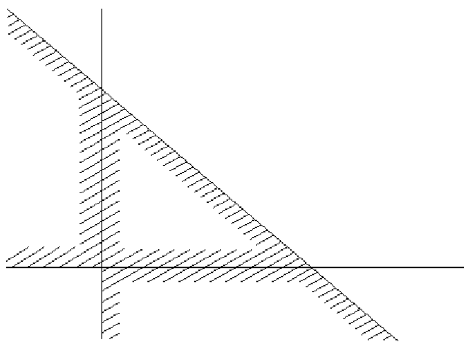


Fig. 2a. Union of atomic regions $\mathcal{F}$.

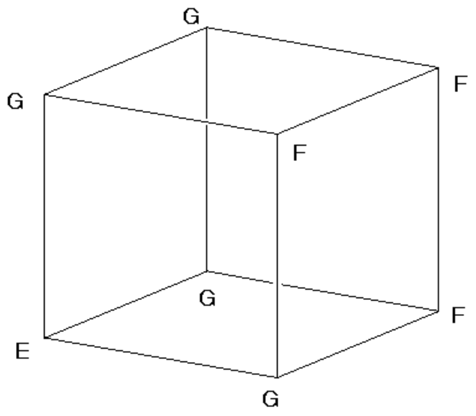
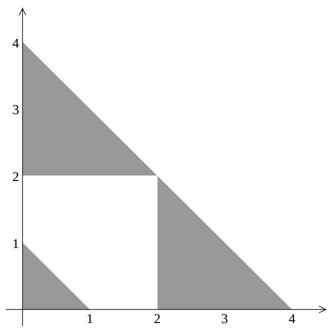


Fig. 2b. Their corresponding cube vertices.

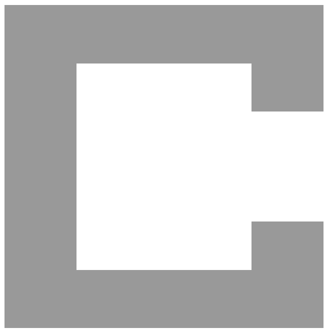
We say the region $\mathcal{F}$ is *implemented* by the two layer net $\mathcal{L}_n^1$ and $\mathcal{L}_m^2$ because the output of neuron O is 1 if and only if $x \in \mathcal{F}$. The Two-layer Perceptron classification problem is that of finding a characterization of those regions of n-dimensional space which can be implemented by a two layer neural net. As we've seen, a collection $\mathcal{F}$ of convex polytopes arising from the decomposition of the input space $\mathbf{R}^n$ by hyperplanes will be two-layer classifiable if and only if their corresponding set of vertices in cube-space can be separated by a cube-plane from the vertices corresponding to the region complementary to $\mathcal{F}$. Evidently, the complement of a classifiable region is itself classifiable because the output neuron will be 1 on the complement if and only if it is 0 on the region.

Although an intrinsic characterization of two-layer classifiable regions is not known, several sets of sufficient conditions have been given, see [1],[2],[3],[5]. One general type of classifiable region is a nest of convex polytopes which alternate in black and white monochromatic colors, that is between $\mathcal{F}$ and its complement; these are called CoRD regions, see [4] and below.

Continuing the color analogy, we shall refer to the cube vertices designated by F as *Black* and those designated by G as *White*. Further, in the remainder we will assume the output of O is to be 1 in the Black regions and 0 on the White ones.



(a) A CoRD region


(b) A CoRD region

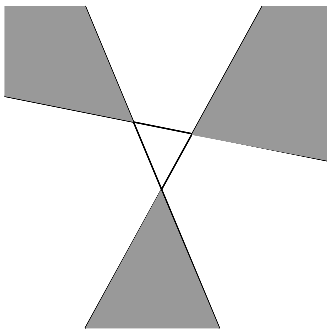

(c) Not a CoRD region

Fig. 3. Regions implementable by a 2-layer net

CoRD Regions

Let $C_1, C_2, \ldots, C_p$ be a nest of convex polytopes

$$C_1 \supset C_2 \supset \ldots \supset C_p. \tag{2.1}$$

We assume p is even, otherwise put $C_{p+1} = \emptyset$. A *convex recursive deletion*, or CoRD, region is a set S of the form

$$S = (C_1 \cap C_2') \cup (C_3 \cap C_4') \cup \ldots \cup (C_{p-1} \cap C_p'). \tag{2.2}$$

where C' denotes the complement of the region C. We allow the possibility that $C_1 = \mathbf{R}^n$. (Then the complement of a CoRD region is such a region also.) Some examples of CoRD regions in $\mathbf{R}^2$ are illustrated in fig. 3a,b. Note that region c of that figure is two-layer implementable but is not a CoRD region.

Note that the representation of a CoRD region S is not unique. For example, the shaded region in Fig. 3a is either the outer triangle with the inner square removed and the inner triangle put back, a three stage construction, or the outer triangle with the truncated square removed, a two stage construction.

It is shown in [4], via a non-constructive proof, that CoRD regions are two-layer classifiable. In this work we give a simple method for calculating a separating cube-plane $K_{V,\eta}$, i.e. for finding weights and thresholds in a two-layer perceptron network which implements a CoRD decision region. This method is presented in the next section. We follow that with our conclusions.

The results of this work help in understanding the relationship between the decision region of a perceptron and its corresponding geometry in input space. Further our constructions extend in a simple way to the case that the decision region is the disjoint union of CoRD regions (requiring three layers). Therefore this work also helps in understanding how many neurons are needed in the second layer of a general, three layer network. Finally, in the event that the decision region of a network is known and is the union of CoRD regions, our results enable the calculation of the weights and thresholds of the implementing network directly and rapidly without the need for thousands of backpropagation iterations.

§2 Separating Cube-plane Construction

The proof that a CoRD region is 2-layer implementable proceeds step-wise by hyperplanes starting from the inside and working out. As each new hyperplane is added, the associated cube increases by one dimension. The new cube consists of the old cube as one face, a duplicate of the old cube as the parallel face in the new dimension, and edges that adjoin like vertices between the old cube and its duplicate. Thus, a 3-dimensional cube consists of two parallel 2-dimensional faces with their four corresponding vertices adjoined. There is a separating cube-plane at each step because the duplicate of the old cube in the new dimension consists of verticies of only one color (or don't care vertices). The construction below follows this pattern; as the hyperplanes are added one at a time, the previous cube-plane solution is extended into the new dimension and then slightly rotated, by adding a new term, in such a way that the duplicated face lies entirely on one side of the new cube-plane.

We begin by ordering the input space hyperplanes, $H_1, H_2, \ldots, H_m$, starting with the inner-most convex region and working out. More explicitly, let S be a CoRD region defined by a nest of convex sets as in equations (2.1) and (2.2). The inner-most convex set, C_p, is the intersection of half-spaces, say $j_p \leq m$ in number, which we label (in arbitrary order) $H_1^{\epsilon_1}$, $H_2^{\epsilon_2}$, …, $H_{j_p}^{\epsilon_{j_p}}$; thus

$$C_p = \bigcap_{k=1}^{j_p} H_k^{\epsilon_k}.$$

As above, ϵ_k is either $+$ or $-$ depending on the orientation of H_k. Next, index the j_{p-1} hyperplanes giving C_{p-1} starting with $k = j_p + 1$ up to $k = j_p + j_{p-1} \leq m$. Continue for C_{p-2} to, lastly, C_1. Then $j_p + j_{p-1} + \ldots + j_1 = m$. Note that, as a consequence, the coordinates y_k will likewise be in this order.

We now define the orientation of the hyperplanes H_k.

Rule 1:

Orient each hyperplane outward (from the convex set it forms), see fig. 4.

Thus for the inner-most convex set, $\epsilon_1 = \ldots = \epsilon_{j_p} = -$ and it corresponds to the vertex at the origin of cube-space.

Without loss of generality, we may write the cube-plane equation in the form

$$s_0 + s_1 a_1 y_1 + s_2 a_2 y_2 + \ldots + s_m a_m y_m = 0, \tag{2.3}$$

where the s_i are $+1$ or -1 and the a_i are non-negative. These parameters will be chosen so the resulting cube-plane separates the Black from the White vertices. Recall we are assuming the Black vertices should evaluate to a positive result.

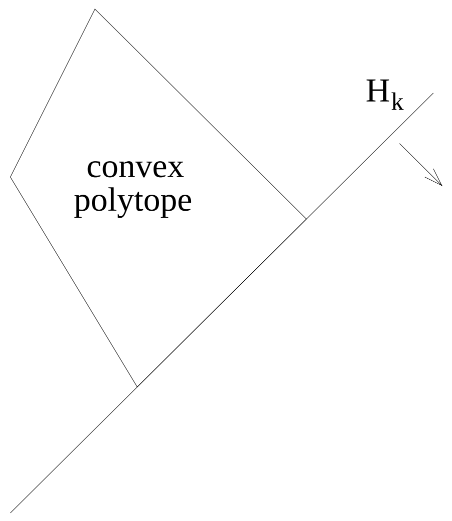


Fig. 4, Hyperplane orientation by Rule 1

Rule 2:

For each new term, $s_k a_k y_k$, if the outward side of H_k is Black (at this stage of the construction), then $s_k = +1$, otherwise $s_k = -1$.

Said differently, for those convex sets used directly to form S, the signs corresponding to their hyperplanes are $+1$, for those whose complement is used to form S, the signs will be -1.

2.1 *CoRD construction algorithm*

The construction starts, step 0, with the entire input space taken to be the same color as the inner-most region. If this region is Black, then the initial sign, $s_0 = +1$ (since we have taken positive to correspond to Black); if the inner-most convex set is White, then $s_0 = -1$.

Now we iterate over the input space hyperplanes working in the order of H_1 to H_m. Rule 2 determines the sign of the new term; it remains to determine the magnitude a_k.

The simplest way to assign the factors in (2.3) is to take

$$a_k = 2^k. \tag{2.4}$$

2.2 *Correctness of the construction algorithm*

The correctness of this simple solution is proved in a similar way as that given below for the more delicately selected magnitudes which we discuss later. For a region as in Fig. 3a, a 5 line White inner convex set removed from a 3 line outer one, this method gives

$$-1 + 2y_1 + 4y_2 + 8y_3 + 16y_4 + 32y_5 - 64y_6 - 128y_7 - 256y_8 = 0.$$

This power of 2 solution yields a cube-plane that is not best possible in the sense of being at the maximal distance from the vertex sets F and G. A maximal distance solution is desirable from the stand point of numerical stability in that it minimizes the possibility that numerical error will result in a faulty assignment. The best possible solution is constructed as follows (see Theorem 1 in Section 2.5 for the demonstration). Put

$$A_k = \text{sum of the coefficients, excluding } s_0, \text{ in sign opposite to } s_k; \tag{2.5}$$

so $-s_k A_k = |A_k|$. Mathematically $A_k = \sum_{j=1}^{k-1} [s_j a_j]^{\epsilon'_k}$ where ϵ'_k is $+$ if $s_k = -1$ and $-$ if $s_k = +1$, and $[x]^\epsilon$ is defined by

$$[x]^+ = \frac{x + |x|}{2} \qquad [x]^- = \frac{x - |x|}{2}.$$

(That is, $[x]^+$ equals x if $x > 0$ and equals 0 otherwise, while $[x]^-$ equals x if $x < 0$ and equals 0 otherwise.) We take a_k to be the solution of

$$s_0 + A_k + s_k a_k = s_k \tag{2.6}$$

which is motivated as follows. On the outward side of H_k, $y_k = 1$. On this side, the expression must equal $+1$ if $s_k = +1$ and -1 if $s_k = -1$. Finally, in the worst case, somewhere in the outward side of H_k, all the $y's$ with opposite sign could be $+1$ while those of the same sign could be 0. Multiply (2.6) by s_k and transpose to get

$$a_k = 1 - s_k s_0 + |A_k|. \tag{2.7}$$

2.3 *Examples*

To illustrate this method, consider the region of Fig. 3a which may be regarded as a three level nest of convex sets. The inner-most convex set, the triangle, is Black and is defined by 3 lines. Following the orientation imposed by Rule 1, and using the step function u_θ as in the Introduction, they are:

$$y_1 = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 0 \\ -1 \end{bmatrix}), \qquad y_2 = u_1(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 1 \\ 1 \end{bmatrix})$$
$$y_3 = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} -1 \\ 0 \end{bmatrix}).$$

Moving to the second layer, the signs of the coefficients of y_1, y_2, and y_3 are, by Rule 2, -1. Using (2.7) for the coefficient magnitudes, the sum of the non-constant terms of opposite sign is 0, so $a_1 = 1 - (-1)(1) = 2$ and the same for a_2 and a_3. Their neural net connection in the second layer will be

$$1 - 2y_1 - 2y_2 - 2y_3 = 0.$$

Next comes the square,

$$y_4 = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 0 \\ -1 \end{bmatrix}), \qquad y_5 = u_1(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 1 \\ 0 \end{bmatrix}),$$
$$y_6 = u_1(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 0 \\ 1 \end{bmatrix}), \qquad y_7 = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} -1 \\ 0 \end{bmatrix}).$$

By Rule 2 the sign for these terms is +1 and the sum of the non-constant coefficients of opposite sign is -6. Hence from (2.7), $a_k = 1 - (+1)(1 - 6) = 6$, $k = 4, 5, 6, 7$. Adding them to the second layer gives

$$1 - 2y_1 - 2y_2 - 2y_3 + 6y_4 + 6y_5 + 6y_6 + 6y_7 = 0.$$

Finally, the outer convex set is the triangle,

$$y_8 = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 0 \\ -1 \end{bmatrix}), \qquad y_9 = u_4(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 1 \\ 1 \end{bmatrix}),$$
$$y_{10} = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} -1 \\ 0 \end{bmatrix}).$$

Doing the coefficient calculation and adding these terms to the second layer gives

$$1 - 2y_1 - 2y_2 - 2y_3 + 6y_4 + 6y_5 + 6y_6 + 6y_7 - 26y_8 - 26y_9 - 26y_{10} = 0.$$

Notice that several of the outputs of the first layer are the same, for example $y_1 = y_4 = y_8$ and $y_3 = y_7 = y_{10}$. These may be combined to give

$$1 - 22y_1 - 2y_2 - 22y_3 + 6y_5 + 6y_6 - 26y_9 = 0.$$

The second interpretation of this CoRD region has an inner-most convex set consisting of the truncated square bounded by 5 outward oriented lines,

$$z_1 = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 0 \\ -1 \end{bmatrix}), \qquad z_2 = u_2(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 1 \\ 0 \end{bmatrix}),$$
$$z_3 = u_2(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 0 \\ 1 \end{bmatrix}). \qquad z_4 = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} -1 \\ 0 \end{bmatrix}),$$
$$z_5 = u_{-1}(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} -1 \\ -1 \end{bmatrix}).$$

Hence

$$-1 + 2z_1 + 2z_2 + 2z_3 + 2z_4 + 2z_5 = 0.$$

This sits within the outer trianglar convex set, with

$$z_6 = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 0 \\ -1 \end{bmatrix}), \qquad z_7 = u_4(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} 1 \\ 1 \end{bmatrix}),$$
$$z_8 = u_0(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^t \begin{bmatrix} -1 \\ 0 \end{bmatrix}).$$

And so

$$-1 + 2z_1 + 2z_2 + 2z_3 + 2z_4 + 2z_5 - 10z_6 - 10z_7 - 10z_8 = 0.$$

As above, this can be simplified.

2.4 *Proof of the construction algorithm*

The proof is by induction on the list of input space hyperplanes. The "0th" hyperplane (none at all) corresponds to the constant term which was chosen to give the right answer inside the inner-most convex set. Now assume the cube-plane equation is correct for hyperplanes H_1, ..., H_{k-1}. By choosing the orientation outward, all regions in the negative half-space of H_k still give the correct response since the new term, $s_k a_k y_k$, is zero there. But in the postive half-space of H_k, the equation is positive if that half-space is Black or negative if it is White. By Rule 1 the choice of the sign, s_k, matches the requirement. Further, the construction of A_k assures that, for $y_k = 1$, the new term by itself exceeds the rest of the equation for all possible values of y_1, ..., y_{k-1}. Hence induction is complete.

2.5 *Optimality of the constructive solution*

Theorem 1. *The solution above is the mini-max solution in the sense that it maximizes the minimal distance from the cube-plane to each of the vertex sets F and G.*

Proof. The most general cube-plane may be put into the form

$$1 + s_1 c_1 y_1 + s_2 c_2 y_2 + \ldots + s_m c_m y_m = 0$$

where the s_i are the signs as above and the c_i are non-negative and yet to be determined. Since this plane must separate vertices, it follows that the signs s_i must be the same as those determined by the construction algorithm above. By elementary methods, one calculates that the directed distance from a point $(b_1, b_2, \ldots, b_m)$ in cube-space and the plane is given by

$$dd = \frac{s_0 + s_1 c_1 b_1 + s_2 c_2 b_2 + \ldots + s_m c_m b_m}{\sqrt{c_1^2 + c_2^2 + \ldots + c_m^2}}.$$

For a given set of coefficients c_i, the nearest points to the plane are those that numerically minimize the numerator. Also note that this directed distance changes sign in concert with the signs s_i. It follows that the candidates for the nearest points can be reduced to those whose coordinates b_k are 1 when the sign sequence s_k switches and the other coordinates for the same sign as s_k are zero. The list can further be reduced by symmetries. When two coordinates can be interchanged and still remain in the same color group, those coordinates are symmetric.

As a consequence, the minimizing vertices give rise to these directed distances (with denominator surpressed)

$$
\begin{aligned}
&s_0 \\
&s_0 + s_1 c_1 \\
&s_0 + n_1 s_1 c_1 + s_{n_1+1} c_{n_1+1} \\
&\dots .
\end{aligned}
$$

The first sign change is between s_0 and s_1. Then after n_1 instances of the sign s_1, the next change is s_{n_1+1} and so on.

Now we seek to maximize the minimum of these numbers. Since $1 = |s_0|$ is among them, it will not be possible to chose the c_i so the minimum exceeds 1. But on the other hand, neither can the maximum of the minimum exceed 1. In fact, by equating all these expressions, there results a solvable system which must be the unique mini-max solution. But this is equivalent to the way the coefficients are chosen in the construction algorithm. ▮

§3 Conclusions

We have presented a simple algorithm capable of computing all free parameters (weights and thresholds) of a two-layer perceptron whose separating hyperplane (cube-plane in the output cube-space) implements a CoRD decision region in input space. We have also proved that this algorithm yields the best possible hyperplane, in the sense that it computes the cube-plane that maximizes its minimal distance from vertex subsets F and G in the cube-space.

References

[1] J. Makhoul, A. El-Jaroudi, R. Schwartz, "Formation of disconnected decision regions with a single hidden layer," *Proceedings of the International Joint Conference on Neural Networks* **I**, 455—460, IEEE TAB Neural Network Committee (1989)

[2] R. McCurley, K. Miller, R. Shonkwiler, "Classification Power of Multiple-layer Artifical Neural Networks," *SPIE 1990, Technical Symposium on Optical Engineering and Photonics in Aerospace Sensing; Program on Optical/Neural Image and Information Processing* Vol. 1294, 577-587 (May 1990)

[3] P. Rujan, M. Marchand, "A Geometric Approach to Learning in Neural Networks," *Proceedings of the International Joint Conference on Nerual Networks* **II**, 105—109, IEEE TAB Neural Network Committee (1989)

[4] R. Shonkwiler, "Separating the Vertices of N-Cubes by Hyperplanes and its Application to Artificial Neural Networks." *Transactions on Neural Networks*, Vol. 4, No. 2, 343-347 (1993).

[5] A. Wieland, R. Leighton, "Geometric Analysis of Neural Network Capabilities," *Proceedings of the Second IEEE International Conference on Nerual Networks* **III**, 385—392, IEEE Computer Society (1988)