PATTERN RECOGNITION WITH UNTRAINED PERCEPTRONS

(Beta version)

Daniel Crespin
Facultad de Ciencias
Universidad Central de Venezuela

Abstract

It is shown that pattern recognition problems can be solved by constructing a polyhedron P. Algorithms for effective construction of P in terms of the problem data are given. In a previous paper it was explained how to build a linear perceptron network equal to the characteristic function of any polyhedron. The perceptron network corresponding to P solves the pattern recognition problem. No network training is necessary.

1.- **Introduction.** Several pattern recognition problems have the following abstract formulation. Finite disjoint sets of data $A = \{a_1, \ldots, a_s\}$, $B = \{b_1, \ldots, b_t\}$ are given in $\mathbf{R}^m$. Elements a_i are *examples* and elements b_j are *counterexamples*. A subset R of $\mathbf{R}^m$ *recognizes* data A *against* data B or simply *recognizes*, if $A \subseteq \text{int}R$ and $B \subseteq \text{int}(\mathbf{R}^m - B)$. See next section for notation. In this definition the interiors are preferred instead of the sets themselves in order to avoid numerical unstability. It will also be said that R *fits the data*. The pattern recognition problem consists in specifying a set R that fits the data. The specification could be an explicit formula for the characteristic function χ_R of R. Without further requirements the problem has many answers, for example the trivial one R =union of open balls with small enough radii centered at points of A, but in general this is not a useful region. A *good* solution is a region R that fits and additionally predicts data, that is, if new examples $a'_1, \ldots, a'_{s'}$ or counterexamples $b'_1, \ldots, b'_{t'}$ are added then R still fits the data sets $A \cup \{a'_1, \ldots, a'_{s'}\}$ and $B \cup \{b'_1, \ldots, b'_{t'}\}$. If R fits but is too small, a situation known as *overfitting*, new examples may not belong to R. If on the contrary R is too large, referred to as *underfitting*,

new counterexamples may belong to R. Therefore the criteria used to select examples and counterexamples have to be analyzed in order to extract the formal rules they contain, an often difficult task. When the formal rules become explicitly known the construction of a good R should be facilitated. But in the last analysis pattern recognition problems can be considered not well posed because the addition of examples and counterexamples, in principle an arbitrary procedure and part of the definition of a good set, may modify the requirements on R in unforeseen ways. So, rather than attempting an unsolvable problem, the algorithms proposed provide a practical tool whose actual performance has to be carefully checked in each particular application. In spite of these practical aims, this paper is essentially theoretical and offers no numerical results.

A procedure much employed in pattern recognition is to select a linear perceptron neural network architecture and initial weights, replace discontinuous thresholds by sigmoids and train by backpropagation using the given data until certain new weights are obtained. The hope is that when the new weights and old discontinuous thresholds are used the output of the network is 1 for $a \in A$ and 0 for $b \in B$. In [3] it is proved that the functions defined by linear perceptron networks are always characteristic functions of linear polyhedra. Therefore perceptron approaches to pattern recognition consist in the construction of the characteristic function of a polyhedron expecting that it fits the data and is a good region. The polyhedron is initially chosen more or less at random (architecture and initial weights) and then it is moved around trying to make it fit the data (training). Metaphorically, a nervous system is randomly initialized and then interaction with the enviroment modifies the setup until certain optimum configuration is reached. The difficulty with this method is that, additionally to unwanted and unpredictable over and underfitting, the chosen architecture could be inadequate and the training may take too long or may even result in a polyhedron that does not fit the data.

It would be useful to have an efficient method that tells how to choose a perceptron architecture and initial weights. If training has to be applied to these initial elements, the method should anticipate the number of necessary iterations. After training the region should fit the data and have reasonable prediction properties for the recognition task.

Such method is explained in this paper. It is shown here how to construct a polyhedron that fits given data. Knowing the polyhedron, the theorems in [7] and [4] indicate how to choose an architecture and weights so that the linear perceptron equals the characteristic map of the polyhedron. According to this method the number of training iterations is zero: the polyhedron always fits. The effect of posterior use of backpropagation or other training procedures to fine tune the weights and modify the fitting of the polyhedron to data has yet to be be methodically explored.

Mathematically the pattern recognition method proposed is simple and natural. Is there an analogous neurophysiological learning mechanism? The author does not know the answer.

2.- **Terminology and notation.** The interior of a set X, in the sense of elementary topology will be denoted int$X$. A *characteristic indicator* or just *indicator* is a cartesian product of characteristic functions. A, B are finite, non-empty disjoint sets in $\mathbf{R}^m$. Linear forms are non-homogeneous linear maps $f : \mathbf{R}^m \to \mathbf{R}$, $f(x_1, \ldots, x_m) = w_0 + w_1 x_1 + \cdots + w_m x_m$. It will be always assumed that linear forms are non-constant, that is, $(w_1, \ldots, w_m) \neq 0$. Denote by h a *Heaviside function.* These can be the *open Heaviside function* $h = \hat{h}$=characteristic function of the open half-line $(0, \infty)$, or the *closed Heaviside function* $h = \bar{h}$=characteristic function of the closed half-line $[0, \infty)$. A *linear perceptron unit*, or simply *perceptron* when the context allows, is a composition $p = h \circ f$ with f=linear form and h=Heaviside function. Let $a, b \in \mathbf{R}^m$. The linear form f *separates* a and b if $f(a)f(b) < 0$. This means that a and b lie in opposite sides of the hyperplane $f = 0$ and is equivalent to $f(a) < 0 < f(b)$ or $f(b) < 0 < f(a)$. To emphasize that the inequalities are strict the term *strict separation* will be sometimes used. Note that if sgn : $\mathbf{R} \to \{-1, 0, 1\} \subseteq \mathbf{R}$ is the sign function, $\mathrm{sgn}(0) = 0$ and $\mathrm{sgn}(\tau) = \tau/|\tau|$ for $\tau \neq 0$, then f separates a and b if and only if $\mathrm{sgn}f(a)\mathrm{sgn}f(b) = -1$. Consider a perceptron $p = h \circ f$. The *half spaces* of p is, by definition, $H = H(p) = p^{-1}(1)$. Separation will also be expressed saying that the hyperplane $f = 0$, the perceptron $p = h \circ f$, the half space $H(p)$ and the complemetary half-space $\mathbf{R}^m - H(p)$ *separate* a and b. Note that because separation is strict, if f separates a and b then a belongs to the open half space $\mathrm{int}(p^{-1}(p(a))$, b belongs to the open half-space $\mathrm{int}(p^{-1}(p(b))$ and these half spaces are disjoint. For a, b and $p = h \circ f$ define the *separation*

index $\gamma = \gamma(f; a, b) = \gamma(p; a, b)$ of f, or of p, on the pair (a, b) as $\gamma = 1$ if f separates a and b and $\gamma = 0$ otherwise.

A collection $f_1, \ldots, f_n$ of forms *separates* $A = \{a_1, \ldots, a_s\}$ and $B = \{b_1, \ldots, b_t\}$ if for each $a_i \in A$ and $b_j \in B$ there exists at least one f_k in the collection with $\gamma(f_k; a_i, b_j) = 1$. The *separation matrix* of f is $\Gamma = \Gamma(f; A, B) = (\gamma_{ij})$ with $\gamma_{ij} = \gamma(f; a_i, b_j)$. This matrix tells which pairs (a, b) are separated by f. The total of such pairs $\nu(\Gamma(f)) = \nu(\Gamma(f; A, B)) = \sum_{i,j} \gamma(f; a_i, b_j)$ is the *separation number*. Let $\max(\alpha, \beta)$ denote the maximum of the numbers α and β and consider the entrywise the maximum of matrices given by $\max((m_{ij}), (n_{ij})) = (\max(m_{ij}, n_{ij}))$. If $\Gamma^{(n)} = \max(\Gamma(f_1), \ldots, \Gamma(f_n))$ then the forms $f_1, \ldots, f_n$ separate A and B if and only if $\Gamma^{(n)} = \mathbf{1}$ where $\mathbf{1}$ is the matrix with all entries equal to 1. All the discussion and results of this paper are formulated over tha real number system $\mathbf{R}$, but with minor modifications they remain valid for the rational number system $\mathbf{Q}$.

3.- **Choice of forms.** Of the various possible ways to choose forms that separate points the most straightforward will be discussed now. If the data consist of a single example, $A = \{a\}$, and a single counterexample, $B = \{b\}$, separation can be realized by the hyperplane containing the middle point $\frac{1}{2}(a + b)$ and perpendicular to the line segment joining a and b. This has equation $f_{a,b} = 0$ where $f_{a,b}(x) = (b - a) \cdot x - \frac{1}{2}(a - b) \cdot (b + a)$. The form $f_{a,b}$ and the perceptron $p_{a,b} = h \circ f_{a,b}$ separate a and b. More generally, and depending on the specific recognition task, the perpendicular hyperplane can be taken through any point $a + t_0(b - a)$ $(0 < t_0 < 1)$ of the segment, the equation being $f_{a,b;t_0} = 0$ with $f_{a,b;t_0}(x) = (b - a) \cdot x(b - a) \cdot (a + t_0(b - a))$. Values of t_0 smaller than $\frac{1}{2}$ give hyperplanes closer to a and eventually, for small enough positive t_0, a tendency to overfitting. Values closer to 1 will produce hyperplanes closer to b and a tendency to underfitting. In the rest of this paper the value $t_0 = \frac{1}{2}$ will be assumed. In practice other values can be taken depending on the specific task. For sets with several elements, $A = \{a_1, \ldots, a_s\}$, $B = \{b_1, \ldots, b_t\}$, the collection of forms f_{a_i,b_j}, $1 \leq i \leq s$, $1 \leq j \leq t$ separates A and B.

4.- **An algorithm for minimality.** It will often be desirable to reduce the number of forms being used to separate A and B. A collection of forms $f_1, \ldots, f_n$ separating sets A and B is *minimal* if no proper subcollection

separates the sets. Any separating collection contains minimal subcollections. The following algorithm succesively calculates the matrices $\Gamma^{(k)} = \max_{j=1}^{k} \Gamma(f_{i_j})$ and stops when $\Gamma^{(k)} = \mathbf{1}$. The forms already examined are those with indices in I_k and the forms with indices in J_k are the ones to be included in the minimal collection. At step k if f_{i_k} separates new pairs (a, b) then it is selected to be part of the minimal collection. Otherwise it is discarded and the algorithm is iterated again.

Start with $k = 1$, $\Gamma^{(0)} = 0$, $J_0 = I_0 = \emptyset$, choose $i_1 \in \{1, \ldots, n\}$ and iterate the following.

Step k.

Data: $\Gamma^{(k)}$, $J_{k-1} \subseteq I_{k-1} \subseteq \{1, \ldots, n\}$, $i_k \in \{1, \ldots, n\} - I_{k-1}$
Let $I_k = I_{k-1} \cup \{i_k\}$
For each $(a, b) \in A \times B$
If $\operatorname{sgn}(f_{i_k}(a)) \cdot \operatorname{sgn}(f_{i_k}(b)) \cdot (1 - m_{ab}^{(k-1)}) < 0$ then $m_{ab}^{(k)} = 1$
Else, $m_{ab}^{(k)} = m_{ab}^{(k-1)}$
Next (a, b)
If $\Gamma^{(k)} =$
$Gamma^{(k-1)}$ then $J_k = J_{k-1}$, choose i_{k+1} and go to step $k + 1$
Else, $J_k = J_{k-1} \cup \{i_k\}$
If $\Gamma^{(k)} \equiv 1$ then $J = J_k$ and stop
Else, choose $i_{k+1} \in \{1, \ldots, n\} - I_k$ and go to step $k + 1$.

The algorithm stops after at most $|J| \leq n$ iterations and the forms $\{f_j | j \in J\}$ are a minimal collection separating A and B. If a small number of iterations is desired the indices i_k should be chosen in decreasing order of separation numbers $\nu(\Gamma(f_{i_1}) \geq \nu(\Gamma(f_{i_2}) \geq \cdots$. This requires the previous calculation of the numbers and ordering of the indices.

5.- **Cells.** A *linear cell* in $\mathbf{R}^m$, or simply *cell*, is a finite intersection of linear half-spaces. Consider a collection $f_1, \ldots, f_n$ of forms in $\mathbf{R}^m$ defining perceptrons $p_1 = h_1 \circ f_1, \ldots, p_n = h_n \circ f_n$, h_i a Heaviside function. These perceptrons define linear half-spaces $H_i = p_i^{-1}(1)$. The *kernel* of the collection of forms, also kernel of the perceptrons, is defined as $\ker = \ker(f_1, \ldots, f_n) = \ker(p_1, \ldots, p_n) = \bigcap_{i=1}^{n} f_i^{-1}(0)$. This kernel is an affine subspace of $\mathbf{R}^m$, pos-

sibly empty.

A *cell index* is a pair $\Sigma = (I_0, I_1)$ with $I_i \subseteq \{1, \ldots, n\}$ and $I_0 \cap I_1 = \emptyset$. By definition the linear cell of Σ is $C_\Sigma = (\bigcap_{i \in I_0}(\mathbf{R}^m - H_i)) \cap (\bigcap_{i \in I_1} H_i)$.

Given forms as above for any $a \in \mathbf{R}^m$ let $I_0(a) = \{i | f_i(a) < 0\}$ and $I_1(a) = \{i | f_i(a) > 0\}$. Note that a is in the kernel if and only if $I_0 = I_1 = \emptyset$. Define the cell index of a as $\Sigma(a) = (I_0(a), I_1(a))$ and the cell of a as $C(a) = C_{\Sigma(a)} = (\bigcap_{i \in I_0(a)}(\mathbf{R}^m - H_i)) \cap (\bigcap_{i \in I_1(a)} H_i)$. The condition $a \notin \ker$ implies that $a \in \text{int}C(a) \subseteq C(a) \neq \mathbf{R}^m$. But if the forms separate a and b then then neither a nor b are in ker, so that $b \in \text{int}C(b) \subseteq C(b) \neq \mathbf{R}^m$ and $C(a) \cap C(b) = \emptyset$.

6.- **Polyhedra.** A *linear polyhedron* in $\mathbf{R}^m$, or simply *polyhedron*, is a finite union of linear cells. Consider again linear forms $f_i : \mathbf{R}^m \to \mathbf{R}$ with perceptrons $p_i = h_i \circ f_i$ and half spaces $H_i = p_i^{-1}(1)$. A *polyhedron index* is a collection $T = \{\Sigma_1, \ldots, \Sigma_q\}$ of cell indices; it defines a polyhedron $P(T) = \bigcup_{i=1}^q C_{\Sigma_i}$. Given a finite non-empty set $A = \{a_1, \ldots, a_s\} \subseteq \mathbf{R}^m$ define the *polyhedron index* of A as $T(A) = \{\Sigma(a_1), \ldots, \Sigma(a_q)\}$ and the ıpolyhedron of A as $P(A) = P(T(A)) = \bigcup_{i=1}^s C_{\Sigma(a_i)}$. If $A \cap \ker = \emptyset$ then $A \subseteq \text{int}P(A)$. Let $B = \{b_1, \ldots, b_t\} \subseteq \mathbf{R}^m$ be finite, non-empty and disjoint from A. If the forms f_i separate A and B then $A \subseteq \text{int}P(A)$, $B \subseteq \text{int}P(B)$ and $P(A) \cap P(B) = \emptyset$. Therefore $P(A)$ (and $\mathbf{R}^m - P(B)$ as well) fit the data.

The *perceptron algebra* $\mathcal{P}$=$\mathcal{P}$ $(f_1, \ldots, f_n)$=$\mathcal{P}$ $(p_1, \ldots, p_n)$ is the algebra of subsets of $\mathbf{R}^m$ generated by the half-spaces of the perceptrons. Recall that this is the smallest collection of subsets of $\mathbf{R}^m$ that contains the half spaces and is closed under intersections, unions and complements. This algebra has finitely many members, these can always be expressed as a finite union of finite intersections of the generating half-spaces and therefore they are linear polyhedra. For any finite set $A \subseteq \mathbf{R}^m$, $A \subseteq P(A) \in \mathcal{A}$. It is proved in [3] that for any linear perceptron neural network the outputs are characteristic functions of polyhedra belonging to $\mathcal{P}$ $(p_1, \ldots, p_n)$ where $p_1, \ldots, p_n$ are the perceptrons of the first layer.

The following algorithm calculates the polyhedron index $T(A)$. Start with $k = 1$, $A_1 = A$, choose $a_{j_1} \in A_1$ and iterate the following:

Step k.

Data: A_k, $a_{j_k} \in A_k$
Calculate $p_1(a_{j_k}), \ldots p_n(a_{j_k})$
Let $I_0(k) = \{i | p_i(a_{j_k}) = 0\}$, $I_1(k) = \{i | p_i(a_{j_k}) = 1\}$.
Let $\Sigma_k = (I_0(k), I_1(k))$, $A_{k+1} = \{a_j \in A_k | a_j \notin C_{\Sigma_k}\}$.
If $A_{k+1} = \emptyset$ then stop
Else, choose $a_{j_{k+1}} \in A_{k+1}$ and go to step $k+1$.

Let $T = \{\Sigma_1, \ldots, \Sigma_q\}$. By construction $a_k \in C_{\Sigma_k}$, $|A_k| < |A_{k+1}|$, the iteration terminates after $q \leq s$ steps and $A \subseteq C_{\Sigma_1} \cup \cdots \cup C_{\Sigma_q} = P_T$. In general, different choices of $a_k \in A_k$ result in different polyhedra.

An equivalent version of the algorithm in terms of the matrix $M = (p_i(a_j))$ is the following. Let M_j be the $j-th$ column of M. Recall that the *support* of a vector $x = (x_1, \ldots, x_n)$ is $\text{supp}(x) = \{i | x_i \neq 0\}$. Start with $k = 1$, $J_1 = \{1, \ldots, n\}$, choose $j_1 \in J_1$ and iterate the following:

Step k.

Data: $J_k \subseteq \{1, \ldots, n\}$, $j_k \in J_k$, M_{j_k}
Let $I_1(k) = \text{supp}(M_{j_k})$, $I_0(k) = \{1, \ldots, n\} - I_1(k)$
Let $\Sigma_k = (I_0(k), I_1(k))$, $J_{k+1} = \{j \in J_k | \text{supp}(M_j) \neq I_1(k)\}$
If $J_{k+1} = \emptyset$ then stop
Else, choose $j_{k+1} \in J_{k+1}$, calculate $M_{j_{k+1}}$ and go to step $k+1$.

According to [4], the characteristic function of $P(A)$ is equal to a three layer linear perceptron neural network. The network architecture and weights are given explicitly. Therefore the previous procedure together with [4] provides an efficient algorithm to solve pattern recognition problems by means of perceptron networks, provided that the data consists of examples *and* counterexamples. The various steps involved in the calculation of the neural network will be resumed now.

STEP A. Calculation of first layer.
Data: Finite non-empty disjoint sets A, B in $\mathbf{R}^m$.
Algorithm: Bias weight $-\frac{1}{2}(b-a)\cdot(b+a)$ and variable inputs weights the components of $(b-a)$, where $a \in A$, $b \in B$. See section 3.

Output: Coefficients of the forms separating A and B, equal to the weights of the first layer.

STEP B. (OPTIONAL) Ordering the forms.
Data: Set of forms separating A and B.
Algorithm: Left to the reader. See end of section 4.
Output: Set of forms separating A and B ordered by decreasing separation numbers.

STEP C. (OPTIONAL) Minimizing the number of forms.
Data: Set of forms separating A and B.
Algorithm: See section 4.
Output: Minimal set of forms separating A and B.

STEP D. Calculation of polyhedron index.
Data: Set of forms separating A and B.
Algorithm: See section 5.
Output: A polyhedron index with corresponding polyhedron that fits the original data.

STEP E. Definition of the neural network.
Data: Polyhedron index.
Algorithm: See sections 3 and 4 of [4].
Output: Architecture and weights of a three linear perceptron neural network equal to the characteristic function of a linear polyhedron that fits the data sets A, B.

7.- **Multiple recognition.** Let $A_1, \ldots, A_r$ be finite, non-empty and mutually disjoint sets in $\mathbf{R}^m$, to be called *data sets* or simply *data*. Define $B_k = \bigcup_{j \neq k} A_j$ so that $A_k \cup B_k = A_1 \cup \cdots \cup A_r$ and $A_k \cap B_k = \emptyset$. The regions $R_1, \ldots, R_r$ *recognize* or *fit* the data if they have mutually disjoint interiors and, for all k, $A_k \subseteq R_k$. The multiple recognition problem is: Given data sets define recognizing regions. As before, the regions are *good* if they predict new data. For example, if A, B, R are as in section 1, take $A_1 = A$, $A_2 = B$, $R_1 = R$ and $R_2 = \mathbf{R}^m - R$. Then R_1, R_2 recognize the data sets A_1, A_2. Characters of an alphabet give rise to typical multiple recognition problems.

A collection $f_1, \ldots, f_n$ of linear forms *(mutually) separates* $A_1, \ldots, A_r$ if for all $i \neq j$, given $a \in A_i$ and $a' \in A_j$ there exists a f_k that separates a and a'. Equivalently, if for all k the forms separate A_k and B_k. It follows that

Theorem. *Data sets $A_1, \ldots, A_r$ are recognized by the polyhedra $P(A_1)$, ..., $P(A_r)$ if and only if the linear forms mutually separates the sets.*

Neural networks for multiple recognition of data sets $A_1, \ldots, A_r$ involve the following procedures:

STEP MA. Calculation of first layer.
Data: Sets $A_1, \ldots, A_r$ in $\mathbf{R}^m$.
Algorithm: For all k and for all $a \in A_k$ and $b \in B_k$, bias weight$=-\frac{1}{2}(b-a)\cdot(b+a)$; variable inputs weights=components of $(b-a)$. See section 3.
Output: Coefficients of the linear forms mutually separating the data, equal to the weights of the first layer. So, the maximum number of perceptron units in the first layer equals $\sum_{i<j}(|A_i| \cdot |A_j|)$. In general minimization lowers this number.

STEP MB. (OPTIONAL) Ordering the forms.
Data: Coefficients of forms mutually separating the data.
Algorithm: For each form f and for all k calculate $\nu(f; A_k, B_k)$, and let the *multiple separation number* be $\nu(f) = \nu(f; A_1, \ldots, A_r)$ be the maximum over k. See end of section 4.
Output: Coefficients of forms mutually separating data, ordered by decreasing multiple separation numbers $\nu(f_1) \geq \nu(f_2) \geq \cdots$.

STEP MC. (OPTIONAL) Minimizing the number of forms.
Data: Coefficients of forms mutually separating data.
Algorithm: Minimize the set of forms needed to separate A_1 and B_1; add forms, as necessary, to separate also A_2 and B_2; add still more, to separate A_3 and B_3, and so on. See section 4.
Output: Minimal set of forms mutually separating the data.

STEP MD. Calculation of polyhedron index.
Data: Set of forms mutually separating data.

Algorithm: For each k apply section 5 to A_k, B_k.
Output: A set of polyhedron indices with corresponding polyhedra that fit the original data.

STEP ME. Definition of the neural network.
Data: Set of polyhedron indices.
Algorithm: See [4], sections 3 and 4.
Output: Architecture and weights of a three layer linear perceptron neural network equal to the indicator of linear polyhedra that fit the data sets $A_1, \ldots, A_r$.

REFERENCES

[1] Crespin, D. Neural Network Formalism. To appear.
[2] Crespin, D. Generalized Backpropagation. To appear.
[3] Crespin, D. Geometry of Perceptrons. To appear.
[4] Crespin, D. Neural Polyhedra.
[5] Crespin, D. Pattern recognition with untrained perceptrons (this present paper).
[6] Crespin, D. Feature Extraction. To appear.
[7] Lippmann, D. Introduction to Neural Networks.

Daniel Crespin
dcrespin@euler.ciens.ucv.ve
Caracas, December 4, 1995.