Networks of Width One Are Universal Classifiers

Raul Rojas
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104-6390
Email: rojas@inf.fu-berlin.de

Abstract— **It is well known that a three-layered neural network can perfectly separate (classify) two classes, that is, two types of data points in n-dimensional space. The number of hidden units is adjusted according to the requirements of the classification problem and can be very high for data sets which are difficult to separate.**

This paper shows that a neural network of width one, i.e. containing at most one computing element in every layer, can perfectly separate two point sets. The network is slender, but can be long. This shows that there is tradeoff between the length and the width of a neural network. The computing elements considered here are perceptrons, and the topology of the network can be best described as a stack of perceptrons.

I. The Classification Problem

The type of classification problems that we consider in this paper are of the following type: two sets of strictly different points in n-dimensional space, i.e. $X = x_1, x_2, \ldots, x_m$ and $Y = y_1, y_2, \ldots, y_k$, are given. A classifier is a system which, given the coordinates of a point z, computes 0 if $z \in X$, and 1 if $z \in Y$. We do not use a table, because new points, not already included in X or Y, will be classified also and we would like to obtain the best possible generalization from the system in terms of the "most similar" class.

It is well known that a multilayer perceptron with a passive input layer and two layers of weighed threshold elements (perceptrons, see [Minsky, Papert 1987]) can solve this problem perfectly, that is, the two given sets can be separated *without error, if it is allowed to increase the number of units* in the hidden layer as needed (see [Hornik et al. 1989] for the continuous case, [Rojas 1986] for a simple treatment of the discrete case). We define the number of hidden units as the width of the network, since the number of input spots is fixed (n coordinates) and the number of output units too (one single output). In this paper we show that a network with *a single perceptron* in each succesive layer of computation can also solve the classification problem perfectly. Such a network is a stack of perceptrons. Therefore, another way of formulating the result we want to prove is by stating that a stack of perceptrons is a universal classifier. Fig. 1 shows the architecture of such a stack.

II. Four-Layered Networks

Let us show, as an introduction, that a four layered network (three layers of perceptrons and a passive input layer) is a universal classifier. Then we will use a similar approach to prove the main result of this paper.

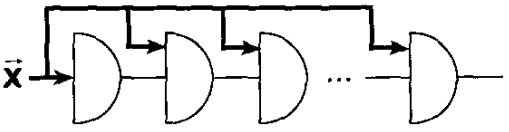

Fig. 1. A stack of perceptrons

A perceptron with n inputs divides an n-dimensional space into two half-spaces, the positive (closed) and the negative (open) region. For a given point in n-dimensional space with coordinates $x_1, x_2, \ldots, x_n$, the perceptron with weights $w_1, \ldots, w_n$ and threshold θ computes a single bit, which is 1 when $\sum_{i=1}^{n} w_i x_i \geq \theta$ and 0 otherwise.

For classification purposes, it is possible to enclose any finite region in n-dimensional space using a polytope with m faces. If two finite classes of data points are given (circles and crosses), we can enclose the circles in one or several polytopes that do not contain crosses. In the worst case, each circle will have a polytope enclosing it (see Fig. 2).

We consider the interior of each polytope with ℓ faces to be a positive convex region because for each of them we can find ℓ perceptrons, one perceptron P_i for each face F_i of the polytope, so that any point in the interior of the polytope belongs to the positive region of P_i, whereas any point on the other side of the face belongs to the negative region of P_i. A multilayer perceptron can then be wired to identify points inside the convex polytopes enclosing one class of data points, as follows:

- The input layer has an input spot for each of the n coordinates of a point (see Fig. 3).
- The first hidden layer consists of perceptrons whose

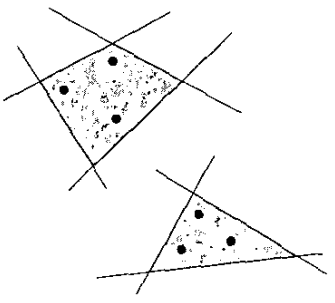

Fig. 2. Convex polytopes enclosing a data set

0-7803-7898-9/03/$17.00 ©2003 IEEE

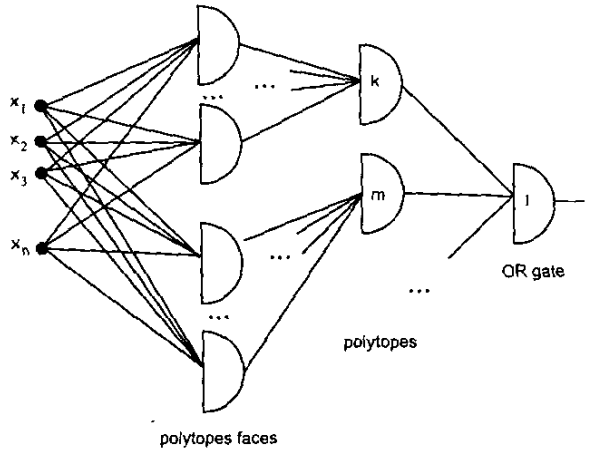

Fig. 3. A four layered universal classifier

weights are chosen as described above, that is, so that the separation plane between the positive and negative half-spaces correspond to one face of a polytope.

- In the second hidden layer, each unit just counts if a point is inside the positive region of all faces of a polytope. In Fig. 2 we show an example where two polytopes with k and m faces, respectively, have been wired in the second hidden layer. Each unit counts if k (m) or less ones arrive from the wired perceptrons in the first hidden layer.
- The output layer consists of a single unit that detects if the point being processed is inside any of the polytopes (it is a logical OR gate).

This shows that a multilayer perceptron with two hidden layers can perfectly classify the given points. Since any problem with multiple classes can be reduced to a problem with two classes (class A against class not-A, class B against class not-B, etc.), it is easy to see that a four layered network can separate any number of points, belonging to several classes. The proof for a multilayer perceptron with a single hidden layer is similar, but a little more intricate [Rojas 1996].

III. Nested Convex hulls

Another way of enclosing the points to be classified inside polytopes is the following. Assume that two finite sets of points are given (circles and crosses, as in Fig. 4). First, the convex hull of the two data sets can be computed ($R1$ in the figure). In the next step, the convex hull of the crosses can be found ($R2$). Then, the convex hull of all circles inside the last constructed convex hull ($R3$). Then the convex hull of all crosses inside the last constructed convex hull ($R4$), and so on. The process end when a convex hull contains only points of one class.

A classifier that could mirror this process could perfectly separate the two sets: if, for example, the last convex hull contained only crosses, any point inside this convex hull (R_4) is classified as a cross. Any point outside the last convex hull, but inside the next to last (the region $R3 - R4$), must be a circle. Any point outside the next to last convex hull, but inside the previously considered region (the region $R2 - (R3 - R4)$, or equivalently, $R2 - R3 + R4$), must be a cross, and so on.

Our problem, therefore, is to show that a stack of perceptrons can reproduce this classification strategy.

IV. Architecture of a Stack of Perceptrons

The architecture of the stack is shown in Fig. 1. Notice that the input is forwarded to each unit in the chain of perceptrons because, if only a bit was forwarded, no further interesting computation would be possible. This is sometimes called a "shortcut" from one layer to another in the neural networks literature. In the strict sense, there are no hidden layers in this network, because every layer receives the coordinates of the point and "sees" the input layer, but the classification is done sequentially, layer by layer, as in conventional networks. This architecture is similar to cascade correlation networks [Fahlman, Lebiere 1990].

The network therefore receives as information in each layer: a) the coordinates of the point being classified, and b) a single bit. The bit will be flipped at different layers. At the end a single bit is produced, the final classification.

Now, we proceed to prove the main result by building a sequence of classifiers. We will use perceptrons for which the negative half-space is closed, and the positive half-space open. That is, given the coordinates $x_1, x_2, \ldots, x_n$ of a point and the weights and threshold $w_1, w_2, \ldots, w_n, \theta$, the perceptron will compute a 1 if $\sum_{i=1}^{n} w_i x_i > \theta$ and 0 otherwise (notice that the inequality comparison is strict).

Given the two sets to be classified, we construct a set of nested convex hulls in the way described above. We first show that it is possible to classify all points inside the last (most internal) convex hull. The points inside that last polytope will constitute the negative class for the classifier P_α to be designed. The classifier is a stack of perceptrons.

The weights of each perceptron in the stack P_α are chosen in such a way that each separating plane coincides with one of the faces of the polytope (R4 in Fig. 4). The negative class lies now towards the interior of the polytope. Any point in the positive region (the exterior) produces a 1 as output of the perceptron. The weight from the output of a perceptron to the next perceptron is fixed as a large number (I will denote this number by ∞, but it is just a large number than can

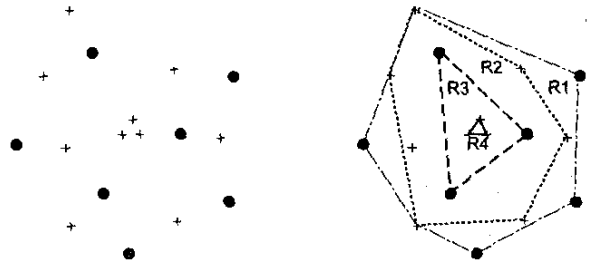

Fig. 4. Nested convex hulls for a two-class data set

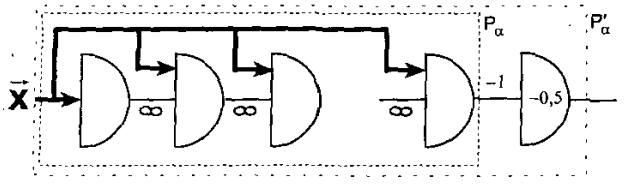

Fig. 5. Nested convex hulls for a two-class data set

obliterate all other calculations. The weights of the perceptron are selected in the interval $[-1, 1]$, and the points to be classified are in a compact region of space. This guarantees that such a large number can be found).

Now consider a point inside the polytope. The first perceptron in the stack P_α produces a zero, the second too, and so on. The last perceptron in the stack will produce a zero, which is the correct result.

If a point is not inside the negative region of the first perceptron, it produces a one as output. Then this one will shut down the chain of perceptrons, because the induced infinite input in the next perceptron produces a 1, which yields an infinite input to the next perceptron and so on. In general: at any point in the chain, when the output of a perceptron is 1, then the whole chain will yield 1 as the final result.

Therefore it is possible to find appropriate weights for a stack of perceptrons such that the output of the stack is 1 for points outside the polytope and 0 for points inside. Now we use this stack P_α as a module.

In the next step, we want to construct a module which yields 0 for points inside the region $R3 - R4$ and 1 for points outside. We first use the stack P_α built in the previous step and reverse its output (using an additional perceptron). The module P'_α yields 1 for points in region R4 and 0 for points outside (see Fig. 5). P'_α is then connected to the next stack P_β of perceptrons, which has the same architecture as P_α (see Fig. 6 in the next page). The output of each perceptron in P_β is connected with an infinite large weight to the next perceptron. The separating planes are selected to coincide with the faces of the polytope $R3$ in our example (Fig. 4). If a point lays outside a face of $R3$, the output of the corresponding perceptron is 1, and this saturates the following perceptrons. With this construction and because P'_α is at the front of the chain, the second module yields a 1 if the point is outside $R3$ or *inside* R4. The output of the double stack just built (P_β) is reversed (with an additional perceptron to give the module P'_β). P'_β is now a classifier for points in the region $R3 - R4$.

Now consider the region R2. P'_β is connected to a new stack P_γ. The connection has infinite weight. The separating planes of the perceptrons coincide with the faces of the polytope R2. If a point is inside the region $R3 - R4$, or outside R2, the output of the chain of perceptrons is 1. Inverting this output, yields a correct classifier for all crosses inside $R2 - (R3 + R4)$ which is $R2 - R3 + R4$.

As can be seen, the above method can be extended to any number of nested polytopes. So it is indeed possible to perfectly separate a two-classes data point set using a stack of perceptrons.

V. Degenerate cases

Notice that a polytope can be degenerate when following the above strategy of nesting convex hulls. If, for example, we only have two points in a two-dimensional plane, the convex hull is a segment. However, we can represent this using two perceptrons with the same separating line but inverted positive and negative regions, and two more perceptrons to delimit the segment. That is, only those points on the delimited segment will be in all four positive regions. Similarly, a single point in a two-dimensional plane can be enclosed using three non-colinear separating lines. In higher dimensions, we would need more separating hyperplanes, but this poses no problem, in principle.

It can also happen that a circle, for example, lays in one of the faces of the convex hull of crosses. However, this is not a real problem, since the next convex hull generates a new region which contains this circle. We could have required all data points to be in general position, but it is not necessary for this proof.

VI. Summary and consequences

We have shown in this paper that a stack of perceptrons can perfectly separate a finite set of points in an n-dimensional space. The general idea is to construct a set of nested convex hulls that can be identified sequentially.

The classification method can be thought of as one in which a single bit is "attached" to the coordinates of the data point being examined. Each perceptron in the chain looks at one single separating plane and the bit attached to the data point. The bit tells the perceptron if the point belongs to the same class as the points outside the face being considered or not. The perceptron checks both if the data point is outside the face, and if the attached bit is set.

The main result of this paper shows that there is a tradeoff between width and length of a neural network. It is interesting in the sense that it deals with a "minimum" network. It is well known that networks of minimal length (one hidden layer) are universal classifiers. Now we know that networks of minimal width can be also universal classifiers.

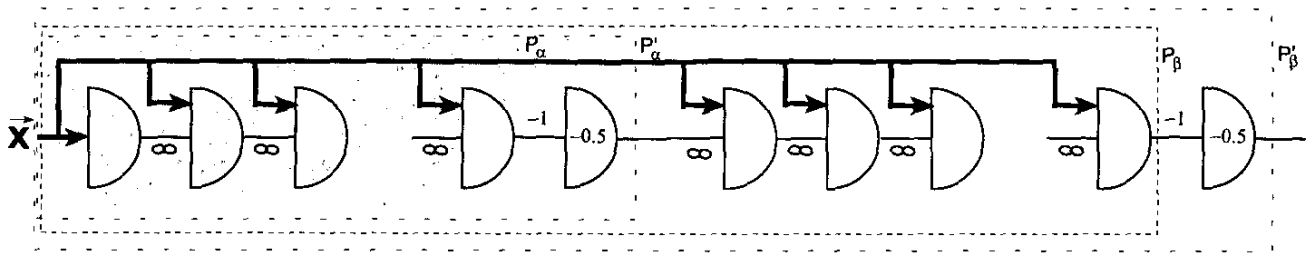

Fig. 6. A classifier for points in $R3 - R4$

REFERENCES

[1] Fahlman, S., Lebiere, Ch., "The Cascade Correlation Learning Architecture", in Touretzky, D.S. (ed.), *Advances in Neural Information Processing Systems*, Vol. 2, Morgan-Kaufmann, San Mateo, pp. 524–432, 1990.

[2] Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," *Neural Networks*, Vol. 2, pp. 359–366, 1989.

[3] Minsky, M., and Papert, S., "Perceptrons", MIT Press, Cambridge, Mass., 1987.

[4] R. Rojas, *Neural Networks*. Berlin, Germany: Springer-Verlag, 1996.