Feed-forward neural networks: a geometrical perspective

1991 J. Phys. A: Math. Gen. 24 881

(http://iopscience.iop.org/0305-4470/24/4/020)

J. Phys. A: Math. Gen. 24 (1991) 881-888. Printed in the UK

Feed-forward neural networks: a geometrical perspective

Marco Budinich and Edoardo Milotti

Dipartimento di Fisica dell'Università di Trieste and Istituto Nazionale di Fisica Nucleare, Trieste, Via Valerio 2, I-34127 Trieste, Italy

Received 15 October 1990

Abstract. The convex hull of any subset of vertices of an n-dimensional hypercube contains no other vertex of the hypercube. This result permits the application of some theorems of n-dimensional geometry to digital feed-forward neural networks. Also, the construction of the convex hull is proposed as an alternative to more traditional learning algorithms. Some preliminary simulation results are reported.

1. Introduction

Feed-forward neural networks are simple [1] but nevertheless powerful and can calculate any function of their inputs given enough 'hidden' neurons (see [2] and [3] for example).

Here we consider feed-forward networks with the following characteristics:

(i) linear threshold digital neurons, i.e. hidden neurons h_i are function of inputs i_j via the classical law:

$$h_i = \Theta\left(\sum_{j=1}^{n} w_{ij} i_j - \theta_i\right) \tag{1}$$

(ii) one hidden layer;

(iii) one output neuron;

(iv) *learning done by presentation of examples.*

This kind of network is completely described by the network topology and by the set of weights (and threshold) of each neuron and has been extensively studied by Minsky and Papert [4] in the case of nets with no hidden layers.

Such networks can typically synthesize an n-variable Boolean function: given a subset of the possible 2^n inputs, the examples, one is required to find a network that answers 1 when presented with one of these cases and 0 otherwise. Very often this problem is solved by trying to minimize the number of neurons of the hidden layer or trying to maximize the information content of the solution network.

An important offspring is that, usually, the net will answer 1 not only to the examples but also to a certain number of other cases. These departures from the 'exact' solution are (rather vaguely) associated with the process of 'generalization' in learning because there is the hope that the net answers are done on the basis of some reasonable criteria and that the generalization is meaningful: this is also one of the most fascinating features of the whole process. Nevertheless it is clear that this process is not controlled and that generalizing properties can be quite 'surprising'. Cover [5] and more recently

0305-4470/91/040881+08$03.50

Carnevali and Patarnello [6] addressed the problem quantitatively. They showed that, starting from probabilistic considerations, one can estimate the probability of a meaningful generalization. It appears quantitatively that, in all non-trivial cases, the net cannot guess what it has not been told explicitly. These considerations suggest to us a net without generalization properties or with these properties added and controlled at will.

A word of caution also about the procedure for the production of the solution network: usually at the start one does not even know how many hidden neurons are needed (not to mention their weights) and the search for a viable configuration is combinatorially difficult. This is the source of the known problems of local minima in the back-propagation algorithm [1] where one tries to minimize the number of errors produced by a fixed number of hidden neurons by changing their weights.

We take here a different stand: we think that at present the problem of the optimization of the network has a low priority while there are more important things to work on:

(i) one should be able to find a solution with no problem of iterative convergence;

(ii) in most real world problems the patterns are strongly correlated with each other: one should take into account this important property;

(iii) generalization properties are very important but usually they are not controlled at all; moreover quantitative results on this aspect (e.g. [5] and [6]) are for random problems and so are not directly applicable to real world problems where patterns are correlated. A good solution should be able to somehow control the generalization property of the network.

Trying to take into account all of this we started to consider the linear space described by the input neurons. The states of the n input neurons may be taken to be the coordinates of an n-dimensional linear space: in this case every possible configuration of the input neurons corresponds to a vertex of an n-dimensional hypercube of side 1. The set of 'acceptable' input patterns is thus a subset of the family of hypercube vertices.

The set of weights and the threshold associated to a given hidden neuron define a hyperplane in this n-space (see (1)): the hyperplane partitions space into two half-spaces, and the hidden-neuron outputs 1 if the input configuration is in the 'positive' half-space and 0 otherwise (this labelling is obviously arbitrary).

The typical problem is to synthesize a network that is capable of reproducing a given set of examples. The learning process thus yields a set of hyperplanes that isolate at least the given examples. Obviously one hyperplane will suffice if and only if the given sets are linearly separable: in this case the perceptron algorithm of Rosenblatt [7] will recursively adjust its only hyperplane position until separation is achieved. In other cases more than one hyperplane will be needed but their number is *a priori* unknown†. For a false two-dimensional example (points are not vertices of a square) see left part of figure 1.

The starting point for our approach originates from the observation that if the patterns are clustered in space (a reasonable hypothesis for real world problems) then one can try to isolate this region of space with an hypersphere or with its approximation given by a set of hyperplanes.

† The tiling algorithm proposed by Mezard and Nadal [2] addresses the problem of the unknown number of hyperplanes needed to separate the sets: it continues to add neurons (hyperplanes) in several layers until an exact solution is found.

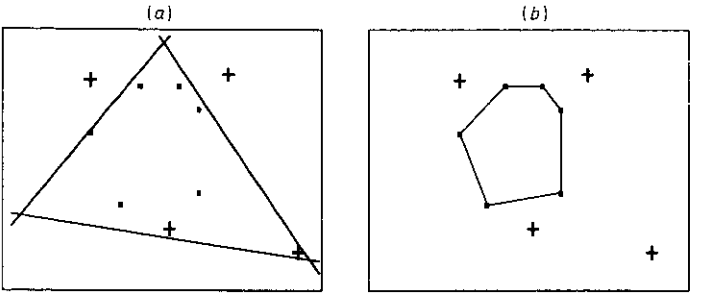

Figure 1. ■ Good cases; + bad cases. (*a*) A 'standard' solution: uncontrolled hyperplanes position; uncontrolled generalization; small number of hyperplanes. (*b*) The convex hull solution: fixed hyperplanes position; no generalization introduced; absolute minimal volume but not minimal number of hyperplanes.

The smallest convex figure, delimited by hyperplanes, that contains all of the given points is the 'convex hull' [8] that by definition is the minimal volume convex polytope that contains all the given points and that can be also thought of as the intersection of the half-spaces defined by its facets.

It is simple to prove that the convex hull of any subset of hypercube vertices do not contain any other hypercube vertex: this guarantees that all 'good' points remain within the convex hull while all the other points remain out (see figure 1). We will see, starting from here, how we can easily build a one-hidden-layer feed-forward net that answers 1 to all 'good' points and 0 to all others thereby producing no uncontrolled generalization.

In what follows we prove this simple proposition that allows us to use the large body of knowledge on convex polytopes to draw some other interesting conclusions.

2. Convexity and some of its consequences

We consider the set $\boldsymbol{H}$ of the 2^n vertices of an n-dimensional hypercube and its subset $\{\boldsymbol{x}_s\}$ formed with the vertices we want to select: we also define the complementary set $\{\boldsymbol{x}_c\} = \boldsymbol{H} - \{\boldsymbol{x}_s\}$.

Theorem. The convex hull of any subset $\{\boldsymbol{x}_s\}$ of vertices of an n-dimensional hypercube contains no other vertices of the hypercube.

Proof. The hypercube itself and a half-space are both convex sets. Each point $\boldsymbol{x}_c$ of the complementary set can be cut off the hypercube by intersecting it with a half-space†. Since the intersection of two convex sets is a convex set at each pass we produce a new convex set. At the end of the process we are left with a convex set with no elements of the complementary set, i.e. we have built a convex set $\boldsymbol{S}$ such that $\boldsymbol{S} \supset \{\boldsymbol{x}_s\}$ and $\{\boldsymbol{x}_c\} \cap \boldsymbol{S} = \varnothing$. This proves that there is a convex set that contains all and only the initial vertices: but $\mathrm{Conv}\{\boldsymbol{x}_s\}$ the convex hull of $\{\boldsymbol{x}_s\}$ is, by definition, the intersection of all the convex sets that contain $\{\boldsymbol{x}_s\}$, therefore $\{\boldsymbol{x}_c\} \cap \mathrm{Conv}\{\boldsymbol{x}_s\} = \varnothing$. □

† Without loss of generality one can choose the origin $(0, 0, \ldots, 0)$ as the vertex to be discarded. If we let $x_1, \ldots, x_n$ be the coordinates, then the half-space $x_1 + x_2 + \ldots + x_n \geq \lambda$ $(0 < \lambda \leq 1)$ selects all the other vertices of the hypercube while discarding the origin.

For all its simplicity, this theorem is not obvious: e.g. figure 12.3 on page 195 of the last edition of Minsky and Papert's famous book [4] (where the sets are shown to be non-convex) is somehow misleading.

Corollary. The selected vertices $\{x_s\}$ are the extreme points of $\mathrm{Conv}\{x_s\}$.

In fact if $x_0 \in \{x_s\}$, it can be cut off the convex hull by a half-space as in the previous proof, while leaving all the other points: then x_0 cannot be a convex combination of the other points, therefore x_0 is an extreme point†. Moreover if the convex hull had any other extreme point, it could likewise be cut off, yielding another, 'smaller', convex set, against the hypothesis that the original set is the convex hull.

Since the convex hull is also the intersection of the half-spaces determined by its supporting hyperplanes, we see immediately how to build a feed-forward neural network with the property of selecting all and only the given examples. This network has just one hidden layer and each of its hidden neurons is associated to one of the supporting hyperplanes (facets) of the convex hull. The output layer performs the logical AND of all the neurons of the hidden layer. The output of this net is 1 if and only if all the hidden neurons of the layer output 1 i.e. if and only if the input state is in the convex hull defined by the examples‡. The network so constructed can store any number m ($1 \leqslant m \leqslant 2^n$) of patterns desired.

A relevant question concerns the number of facets N_f (i.e. of hidden neurons) of the convex hull of m points (examples) in n-dimensional space. By the McMullen upper bound theorem [9] the maximum value N_f can have is§:

$$N_f \leqslant \frac{n+1}{m-(n-1)/2}\binom{m-(n-1)/2}{1+(n-1)/2} \qquad (n \text{ odd})$$

$$N_f \leqslant \frac{m}{m-n/2}\binom{m-n/2}{n/2} \qquad (n \text{ even})$$

$$m \geqslant n+1.$$

This number of facets is reached when the m points in n-dimensional space are in general position and form a 'cyclic polytope' that has the maximum number of facets a convex polytope can have [8].

A set of m (with $m \geqslant n+1$) n-dimensional points are said to be in general position when no subset of $n+1$ points lie on a $(n-1)$-dimensional hyperplane.

In our case we are dealing with hypercube vertices where not any choice of $2n$ points can be in general position. This makes our case very much simpler than a cyclic polytope and we expect a number of facets substantially smaller than the upper bound quoted here.

As we mentioned in the introduction, simple perceptrons must deal with the linear separability of disjoint sets of 'positive' and 'negative' examples: as we have seen they

† Let $x_1, \ldots, x_k$ be k n-dimensional points: then $\lambda_1 x_1 + \ldots + \lambda_n x_n$ with $\lambda_1 + \ldots + \lambda_n = 1$ and $\lambda_i \geqslant 0$, is a convex combination of these points (see, e.g., [8]). A convex combination of extreme points in a convex set spans the whole set.

‡ The complementary network (i.e. that with opposite output function) can be easily built starting from the convex hull of the complement of our initial set. The two nets are logically equivalent and one can choose the one using the less hidden neurons (i.e. the convex hull with less facets).

§ These formulae have the asymptotic value $N_f = O(m^{\lfloor n/2 \rfloor}/\lfloor n/2 \rfloor !)$ when $m \gg n \gg 1$. These formulae could also be used to estimate a lower bound on the number m of patterns a given convex hull with N_f facets can store. Anyhow, as for the upper bound case, we expect this lower bound not to be tight at all.

can both be enclosed in their convex hulls. That the sets of examples are almost never linearly separable can be seen from a theorem of Füredi [10] who proved that the probability that the hypercube centre belongs to the interior of convex hull of a random subset of m hypercube vertices in n-dimensional space is $[1-O(1/\sqrt{n})]$ when $n \gg 1$ and $m > 2n$.

When translated to our present context, this means that given any two sets of more than $2n$ examples, their convex hulls contain a common point with probability approaching 1 for large n, therefore the two sets are almost always linearly inseparable, i.e. a simple perceptron with n inputs has almost always—for sufficiently large example sets—a bad performance.

This result is different from the classical one of Cover [5] (that states that $2n$ random points in general position are almost never linearly separable) in the sense that here the strong request that the m points be in general position is not necessary: our points are vertices of an hypercube: a situation much more natural for neural network problems. This result is supported by the observation that on average one needs more hyperplanes to separate sets of points if they are not in general position than if they are [11].

3. Practical considerations

We have seen that constructing the convex hull 'replaces learning', but how good is all this for practical purposes?

The computation of the convex hull of a set of m points in n-dimensional space is a well known problem for which there are well studied, exact, algorithms described at length in the books by Edelsbrunner [12] and Preparata and Shamos [13]. The algorithm we have chosen for our numerical simulation requires, in the worst cases (cyclic polytopes), a time of the order $O(m^{\lfloor n/2 \rfloor + 1})$ and a memory of the same order of magnitude. This algorithm works in incremental mode: it starts by building the convex hull of two points and then it adds the other points one at the time recalculating the updated convex hull at each step. This structure seems to fit quite nicely with the needs of neural networks learning.

In practice, having fixed at n the input size of the network (the output size is fixed at 1 in our discussion), one feeds one at a time the m examples to the algorithm that at each step returns the N_f hyperplane equations that define the convex hull of all points supplied up to that moment. Given this structure one needs to give only the positive examples.

The construction of the convex hull is a solution to the problem of learning: if one compares it to more traditional learning algorithms one notices that:

(i) it is free from problems of convergence;

(ii) it adapts the size of the network to the problem and it does not get stuck into local minima;

(iii) every example has to be given only once;

(iv) it generates an exact (in the sense described before) solution that is a clear, intuitive, geometrical figure of simple interpretation, though it is not necessarily the most 'economical';

(v) it does not produce any unknown, random, generalization but given the simple geometrical structure of the solution it allows to add known, understandable, generalization properties.

We remark that the 'training time' for the convex hull algorithm is at worst of the order $O(m^{\lfloor n/2 \rfloor + 1})$ (see [12] and [13] for a proof): this is an upper bound, but already better than the performance of many 'biologically motivated' algorithms that scale like $O([c(n)]^m)$ ($c(n)$ is some parameter which depends on the network architecture, see e.g. [14]).

After having built the convex hull one has a solution which recognizes all and only the given examples, thus no generalization is produced. Several ways to introduce this property in a controlled fashion now can be derived. The most obvious is to 'inflate' the convex hull by shifting the hyperplanes of the facets along their orthogonal direction and far from the interior of the convex hull. This and other possibilities are at present under study.

4. Numerical simulations

We performed a Monte Carlo simulation choosing randomly m patterns among the vertices of the n-dimensional hypercube. The number of facets of their convex hull was computed with the standard algorithm and the procedure was repeated several times for each value of m and subsequently the average number of facets was calculated.

Figure 2 shows the average number of hidden neurons for a set with $n = 8$ (i.e. a network with 8 inputs), with m varying between 9 and 255. The continuous curve on the left is the McMullen upper bound.

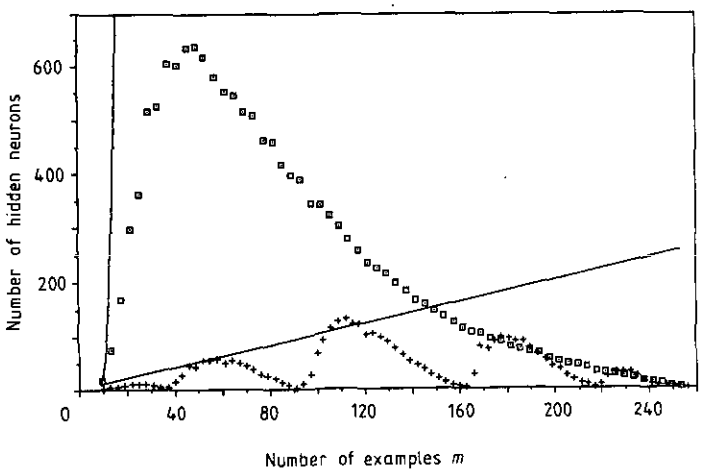

Figure 2. Number of convex-hull facets for random examples (⊡) and for correlated examples (+), for a network with n inputs. The upper full line is the McMullen bound, while the lower full line is the number of facets for the trivial solution (one neuron for each example).

In the upper series of data points (squares) the m different examples were selected randomly. This means that this curve represents the number of hidden neurons for a random Boolean function.

The lower data points (crosses) show how the number of facets depends on m when the examples proposed to the algorithm are strongly correlated. In this case the examples are selected with the following criteria. The first 'point' chosen is always the

origin; then one chooses all the points with Hamming distance 1 from the origin; then all the points with Hamming distance 2 and so on until all the necessary m patterns are found. For each value of the Hamming distance the points are selected randomly and are different from each other: when no more points can be extracted with a certain Hamming distance the value of the Hamming distance is increased by one. It is clear that in correlated cases the convex hull algorithm is much more efficient. The periodic drop in the number of facets is reached when m is such that the points 'fill' all the available space up to a certain Hamming distance: in fact it is easy to prove that in this case one hyperplane plus n hypercube facets (which can be dropped) form the convex hull of the set.

The straight line represents the number of hidden neurons needed by the solution that assigns a specific hidden neuron to each of the examples: the so-called grandmother neuron representation. This representation, though it always provides a solution with a linear increase of the number of hidden neurons, is a trivial one and does not take advantage of any correlation between the examples.

Figure 3 shows the average number of facets for random Boolean function repeated for several nets with the number of inputs n varying from 4 to 8†. On the x axis there is the normalized coordinate $m/2^n$ and the curves have been normalized for easy comparison.

It appears that maxima shift to lower values of $m/2^n$ and also becomes narrower as n increases. A preliminary fit to the position of maxima shows that their normalized coordinates are proportional to $n^2/2^n$.

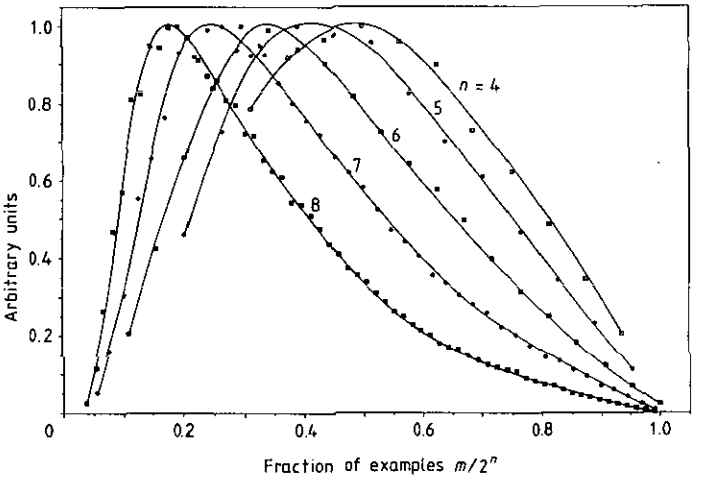

Figure 3. Average number of facets for random Boolean functions.

5. Conclusions

We have related the problems of feed-forward neural networks to the theory of n-dimensional convex polytopes and we have shown that the convex hull of the examples can provide a feed-forward net that solves the problem without uncontrolled

† We have not been able to do these simulations for $n > 8$ due to memory limitations of our personal computers.

generalization. Starting from this we have shown that simple perceptrons are 'almost always' inefficient when examples are taken in the hypercube (and thus not in general position). Finally, we have put forward what we believe to be a novel algorithm to build feed-forward nets that, on the basis of general considerations, could be a promising alternative to traditional learning algorithms.

Preliminary results on the practical performances of the algorithm show that, in the case of uncorrelated patterns and large input space n (the design criteria we followed), the algorithm could be competitive with traditional ones with the added bonus of geometrical clarity and tuneable generalization properties.

Appendix

We propose here a solution that, by adding another hidden layer, reduces the total number of hidden neurons.

The worst-case behaviour derived from McMullen's theorem suggests a highly nonlinear behaviour for the growth of the number of facets (i.e. hidden neurons). Even if mitigated by the hypercube symmetry one can expect a polynomial law such as αm^{β} (with $\beta \leqslant \lfloor n/2 \rfloor$, $\alpha = \alpha(n)$) for the (average) number of facets: this, in turn, suggests a way to devise a more 'economical' network. If one splits the set of examples into k subsets of approximately m/k examples each, one can again recover all the examples by ANDing for each of the resulting k convex sets and thereafter ORing the k results. In this way one constructs a network with two hidden layers that still performs the same task as the original network and has a total of approximately $k + k\alpha(m/k)^{\beta}$ neurons in the first and second hidden layers. By deriving with respect to k one finds that the minimum for the combined number of neurons in the first and second layer is obtained for $k \approx (\alpha(\beta-1))^{1/\beta} m$. If one takes the asymptotic behaviour from McMullen's theorem ($\alpha \approx 1/(n/2)!$, $\beta \approx n/2$) then $k \approx 2e(m/n) \approx 5.4(m/n)$.

References

[1] Rumelhart D E, McClelland J L and the PDP Research Group 1986 *Parallel Distributed Processing* vol 1 (Cambridge, MA: MIT Press) pp xx–548

[2] Mezard M and Nadal J-P 1989 Learning in feedforward layered networks: the tiling algorithm *J. Phys. A: Math. Gen.* **22** 2191–203

[3] Baum E B 1988 On the capabilities of multilayer perceptrons *Complexity* **4** 193–215

[4] Minsky M L and Papert S 1988 *Perceptrons* (Cambridge, MA: MIT Press) pp xv–292

[5] Cover T M 1965 Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition *IEEE Trans. Electron. Comput.* **EC14** 326–34

[6] Carnevali P and Patarnello S 1987 Exhaustive thermodynamical analysis of Boolean learning networks *Europhys. Lett.* **4** 1199–204

[7] Rosenblatt F 1962 *Principles of Neurodynamics* (New York: Spartan)

[8] Lawden G H 1985 *Convexity* in *Handbook of Applicable Mathematics* ed W Ledermann and S Vajda (New York: Wiley) pp 105–37

[9] McMullen P 1970 The maximum number of faces of a convex polytope *Mathematika* **17** 179–84

[10] Füredi Z 1986 Random polytopes in the d-dimensional cube *Discrete Comput. Geom.* **1** 315–9

[11] Mitchison G J and Durbin R M 1989 Bounds on the learning capacity of some multi-layer networks *Biol. Cybernet.* **60** 345–56

[12] Edelsbrunner H 1987 *Algorithms in Combinatorial Geometry* (Berlin: Springer) pp xvi–424

[13] Preparata F P and Shamos M I 1985 *Computational Geometry* (Berlin: Springer) pp xii–390

[14] Tesauro G and Janssens B 1988 Scaling relationships in back-propagation learning *Complex Systems* **2** 39–44