ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Trieste

INFN/FM-90/02

7 giugno 1990

M. Budinich and E. Milotti

FEED-FORWARD NEURAL NETWORKS: A GEOMETRICAL PERSPECTIVE

Servizio Documentazione
dei Laboratori Nazionali di Frascati

Feed-Forward Neural Networks: a Geometrical Perspective.

Marco Budinich and Edoardo Milotti,
Dipartimento di Fisica dell'Università di Trieste and INFN Trieste,
Via Valerio 2, I-34127 Trieste, Italy.

(Submitted to Europhysics Letters, May 31, 1990)

PACS 87.10 - General, theoretical and mathematical biophysics
PACS 05.90 - Other topics in statistical physics and thermodynamics

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 feedforward neural networks. Also, the construction of the convex hull is proposed as an alternative to more traditional learning algorithms.

1. Introduction

Feed-forward neural networks are relatively simple [1] but nevertheless can calculate any boolean function of their inputs given a sufficiently high number of "hidden" neurons.

In what follows we consider feed-forward networks with:

- digital neurons (i.e. neurons may only take the values 0 or 1),
- one hidden layer,
- one output neuron,
- learning done by examples presentation.

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 [2] in the case of nets with no hidden layers.

The states of the n input neurons may be taken to be the coordinates of vector in an n-dimensional 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: 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 learning 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. Obviously one hyperplane will suffice if and only if the given sets are linearly separable: in this case the perceptron algorithm of Rosenblatt [8] will recursively adjust its only hyperplane position until separation is achieved.

In the general case one does not even know how many hyperplanes will be needed (not to mention their position) and the search becomes 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 hyperplanes by changing their orientation.

The tiling algorithm proposed by Mezard and Nadal [3] 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 (by "exact solution" one means a solution which acts properly on a subset of the - supposedly very large - set of "acceptable" input states: this means that the hyperplanes of the network select all the given examples but usually also other vertices of the hypercube). These departures from the "exact" solution are (rather vaguely) associated to the process of "generalization" in learning because there is the hope that the hyperplanes perform the selection on the basis of some sensitive criteria and that the generalization will be 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". Carnevali and Patarnello [6] addressed quantitatively the problem.

In what follows we show that one can construct a convex hull that contains all and only the wanted vertices. This convex hull is the "smallest" convex polytope that "contains" all the "acceptable" input patterns: this is - in a way - a desirable property, even if it yields no "generalization" at all. The theorem is simple and intuitive, but it 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 **H** of the 2^n vertices of an n-dimensional hypercube and the subset $\{\mathbf{x}_s\}$ 1 **H** of vertices that we want to select; we also define the complementary set $\{\mathbf{x}_c\} = \mathbf{H} - \{\mathbf{x}_s\}$.

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

The hypercube itself and a half-space are both convex sets. Each point $\mathbf{x}_c$ of the complementary set can be cut off the hypercube by intersecting it with a half-space[1]. 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 **S** such that $\{\mathbf{x}_s\}$ 1 **S** and $\{\mathbf{x}_c\}$ 9 **S** = 0. This proves that there is a convex set that contains all and only the initial vertices: but Conv$\{\mathbf{x}_s\}$ the convex hull of $\{\mathbf{x}_s\}$ is - by definition - the intersection of all the convex sets that contain $\{\mathbf{x}_s\}$, therefore $\{\mathbf{x}_c\}$ 9 Conv$\{\mathbf{x}_s\}$ = 0.■

For all its simplicity, this theorem is not obvious: e.g. figure 12.3 at page 195 in the last edition of Minsky and Papert's famous book [2] (where the sets are shown to be non-convex) is somehow misleading.

Corollary: the selected vertices $\{\mathbf{x}_s\}$ are the extreme points of Conv$\{\mathbf{x}_s\}$.

In fact if $\mathbf{x}_0 \in \{\mathbf{x}_s\}$, it can be cut off the convex hull by a half-space as in the previous proof, while leaving all the other points: then $\mathbf{x}_0$ cannot be a convex combination of the other points, therefore $\mathbf{x}_0$ is an extreme point[2]. Moreover if the

---

[1] One can take the vertices of the hypercube to be 0-1 vectors and take - without loss of generality - the origin (0,0, ... ,0) as the vector 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.

[2] Let $\mathbf{x}_1, \ldots, \mathbf{x}_k$ be k n-dimensional vectors: then $\lambda_1\mathbf{x}_1 + \ldots + \lambda_n\mathbf{x}_n$ with $\lambda_1 + \ldots + \lambda_n = 1$ and $\lambda_i \geq 0$, is a convex combination of these vectors (see, e.g. [11]). A convex combination of extreme points in a convex set spans the whole set.

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, if we go back to neural networks we see immediately how to build a feedforward 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. [3]

A relevant question concerns the number of facets (i.e. of hidden neurons) of the convex hull: a partial answer can be given using McMullen's Upper Bound Theorem on the maximum number of facets of a convex polytope ([7] and [11]). For a convex polytope with m points in n-dimensional space the theorem yields the asymptotic estimate (when $m >> n >> 1$) $\dfrac{m^{\lfloor \frac{n}{2} \rfloor}}{\lfloor \frac{n}{2} \rfloor !}$ for the maximum number of facets.

One expects, in view of the high degree of symmetry of the hypercube,that the number of facets of the convex hull of a random subset of its vertices be substantially smaller than the upper bound quoted here; still it can be used to make some estimates.

It is reasonable to assume that in practice the number of hidden neurons never exceeds some power n^k of the number n of input neurons, then the network has

$$n^k \approx \frac{m^{\lfloor \frac{n}{2} \rfloor}}{\lfloor \frac{n}{2} \rfloor !} \qquad (\text{if} \quad m >> n >> 1)$$

hidden neurons and using logarithms and Stirling's approximation for factorials

[3] 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 faces).

$$k \log n \approx \lfloor \tfrac{n}{2} \rfloor \log m - \lfloor \tfrac{n}{2} \rfloor \left(\log(\lfloor \tfrac{n}{2} \rfloor) - 1 \right)$$

and for $n >> 1$ this gives $m \approx \frac{n}{2e} \approx 0.18\, n$ for the minimum number of patterns that such a network can store.

It is interesting to remark that some well-understood network models (compatible with the hypothesis (# of hidden neurons) $\approx n^k$) can store no more than $m \approx \alpha\, n$ examples, with $\alpha \approx 0.2$ (e.g. [9]).

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 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 $[1 - O(\frac{1}{\sqrt{n}})]$ when $n >> 1$ 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.

3. Practical considerations.

We have seen that constructing the convex hull "replaces learning", but how good is all this for practical purposes? To compute the convex hull of a set of m points in general position in n-dimensional space there are known, good, algorithms (see [4] and [5]). One of them requires, in the worst cases, a time of the order $O(m^{\lfloor \frac{n}{2} \rfloor + 1})$ and a space of the same order of magnitude. This algorithm works in incremental mode: it starts by making the convex hull of two points and then it adds points and recalculates the updated convex hull. This structure seems to fit quite nicely with the needs of neural networks learning.

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:

- it is free from problems of convergence;
- it adapts the size of the network to the problem and it does not get stuck into local minima;
- every example has to be given only once;
- 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";
- 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 $\mathbf{O}(m^{\lfloor\frac{n}{2}\rfloor+1})$: this is an upper bound, but already better than the performance of many "biologically motivated" algorithms that scale like $\mathbf{O}(\,[c(n)]^m\,)$ ($c(n)$ is some parameter which depends on the network architecture, see e.g. [12])

The worst-case behaviour derived from McMullen's theorem also suggests a highly non-linear behaviour for the growth of the number of facets (i.e. hidden neurons). Even if that were mitigated by the hypercube symmetry it is reasonable to expect a polynomial law like αm^{β} (with $\beta \leq \lfloor\frac{n}{2}\rfloor$, $\alpha = \alpha(n)$) 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 $\frac{m}{k}$ examples each, one can again recover all the examples by AND-ing for each of the resulting k convex sets and thereafter OR-ing 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 $k + k\,\alpha\left(\frac{m}{k}\right)^{\beta}$ 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 $k \approx (\alpha(\beta-1))^{\frac{1}{\beta}}\,m$. If one takes the asymptotic behaviour from McMullen's theorem ($\alpha \approx \frac{1}{\left(\frac{n}{2}\right)!}$, $\beta \approx \frac{n}{2}$) then $k \approx 2e\,\frac{m}{n} \approx 5.4\,\frac{m}{n}$.

4. Conclusions.

We have related the problems of feed-forward neural networks to the theory of n-dimensional convex polytopes. We have considered a simple network architecture with n inputs and we have shown that it can store at least 0.18n patterns: this bound - derived from geometry - is remarkably similar to bounds in other network architectures. We have also shown that simple perceptrons are "almost always" inefficient. Finally, we have put forward what we believe to be a novel algorithm to build feed-forward networks. On the basis of general considerations, it appears to be a promising alternative to traditional learning algorithms. A forthcoming paper will report on practical results.

References:

[1] Rumelhart David E., McClelland James L. and the PDP Research Group, *Parallel distributed processing*, Volume I: Foundations, MIT Press (1986), Cambridge, Massachussets, USA, pp. xx-548

[2] Minsky M.L. and Papert S., *Perceptrons*, MIT Press (1988), Cambridge, Massachussets, USA, pp. xv-292

[3] Mezard Marc and Nadal Jean-Pierre, *Learning in Feedforward Layered Networks: the Tiling Algorithm*, Journal of Physics A (Mathematical and General), **22** (1989) pp. 2191-2203

[4] Edelsbrunner Herbert, *Algorithms in Combinatorial Geometry*, Springer-Verlag (1987), pp. xvi-424

[5] Preparata Franco P. and Shamos Michael Ian, *Computational Geometry*, Springer-Verlag (1985), pp. xii-390

[6] Carnevali P. and Patarnello S., *Exhaustive Termodynamical Analysis of Boolean Learning Networks*, Europhysics Letters, **4** (10) (1987), pp. 1199-1204

[7] McMullen P., *The Maximum Number of Faces of a Convex Polytope*, Mathematika **17** (1970), pp. 179-184

[8] Rosenblatt Frank, *Principles of Neurodynamics*, Spartan Books, New York 1962

[9] Meyr R. and Domany E., *Exact Solution of a Layered Neural Network Model*, Phys. Rev. Lett. **59** (1987) 359-362

[10] Füredi Z., *Random Polytopes in the d-Dimensional Cube*, Discrete Comput. Geom. **1** (1986), pp. 315-319

[11] Lawden G.H. *Convexity* in *Handbook of Applicable Mathematics* (Ledermann W and Vajda S. editors) Wiley, New York (1985), pp. 105-137

[12] Tesauro G. and Janssens B., *Scaling Relationships in Back-Propagation Learning*, Complex Systems **2** (1988), pp. 39-44