

Neural Networks with Quadratic VC Dimension

Pascal Koiran
LIP, ENS Lyon – CNRS
46 allée d’Italie
69364 Lyon Cedex 07, France

koiran@lip.ens-lyon.fr

Eduardo D. Sontag²
Department of Mathematics
Rutgers University
New Brunswick, NJ 08903, USA

sontag@hilbert.rutgers.edu

Abstract

This paper shows that neural networks which use continuous activation functions have VC dimension at least as large as the square of the number of weights w . This result settles a long-standing open question, namely whether the well-known $O(w \log w)$ bound, known for hard-threshold nets, also held for more general sigmoidal nets. Implications for the number of samples needed for valid generalization are discussed.

Keywords: feedforward neural networks, learning, Vapnik-Chervonenkis dimension

1 Introduction

One of the main applications of artificial neural networks is to pattern classification tasks. A set of labeled training samples is provided, and a network must be obtained which is then expected to correctly classify previously unseen inputs. In this context, a central problem is to estimate the amount of training data needed to guarantee satisfactory learning performance. To study this question, it is necessary to first formalize the notion of learning from examples.

One such formalization is based on the paradigm of *probably approximately correct (PAC)* learning, due to Valiant ([16]). In this framework, one starts by fitting some function f , chosen from a predetermined class \mathcal{F} , to the given training data. The class \mathcal{F} is often called the “hypothesis class”, and for purposes of this discussion it will be assumed that the functions in \mathcal{F} take binary values $\{0,1\}$ and are defined on a common domain X . (In neural networks applications, typically \mathcal{F} corresponds to the set of all neural networks with a given architecture and choice of activation functions. The elements of X are the inputs, possibly multidimensional.) The training data consists of labeled samples (x_i, ε_i) , with each $x_i \in X$ and each $\varepsilon_i \in \{0,1\}$, and “fitting” by an f means that $f(x_i) = \varepsilon_i$ for each i . Given a new example x , one uses $f(x)$ as a guess of the “correct” classification of x . Assuming that both training inputs and future inputs are picked according to the same probability distribution on X , one needs that the space of possible inputs be well-sampled by the training data, so that f is an accurate fit. We omit the details of the formalization of PAC learning, since there are excellent references available, both in textbook (e.g. [2, 12]) and survey paper (e.g. [11]) form, and the concept is by now very well-known.

After the work of Vapnik in statistics ([17]) and of Blumer et. al. in computational learning theory ([5]), one knows that a certain combinatorial quantity, called the *Vapnik-Chervonenkis (VC) dimension* $VC(\mathcal{F})$ of the class \mathcal{F} of interest completely characterizes the sample sizes needed for learnability in the PAC sense. (The appropriate definitions are reviewed below. In Valiant’s formulation one is also interested in quantifying the computational effort required to actually fit a function to the given training data, but we are ignoring that aspect in the current paper.) Very roughly speaking, the number of samples needed in order to learn reliably is proportional to $VC(\mathcal{F})$. Estimating $VC(\mathcal{F})$ then becomes a central concern. Thus from now on, we speak exclusively of VC dimension, instead of the original PAC learning problem.

The work of Cover ([6]) and Baum and Haussler ([3]) dealt with the computation of $VC(\mathcal{F})$ when the class \mathcal{F} consists of networks built up from hard-threshold activations and having w weights; they showed that $VC(\mathcal{F}) = O(w \log w)$. (Conversely, Maass showed in [10] that there is also a lower bound of this form.) It would appear that this definitely settled the VC dimension (and hence also the sample size) question.

²This research was supported in part by US Air Force Grant AFOSR-94-0293.

However, the above estimate assumes an architecture based on hard-threshold (“Heaviside”) neurons. In contrast, the usually employed gradient descent learning algorithms (“backpropagation” method) rely upon *continuous* activations, that is, neurons with graded responses. As pointed out in [14], the use of analog activations, which allow the passing of rich (not just binary) information among levels, may result in higher memory capacity as compared with threshold nets. This has serious potential implications in learning, essentially because more memory capacity means that a given function f may be able to “memorize” in a “rote” fashion too much data, and less generalization is therefore possible. Indeed, the paper [15] showed that there are conceivable (though not very practical) neural architectures with extremely high VC dimensions. Thus the problem of studying $\text{VC}(\mathcal{F})$ for analog networks is an interesting and relevant issue. Two important contributions in this direction were the papers by Maass ([10]) and by Goldberg and Jerrum ([7]), which showed upper bounds on the VC dimension of networks that use piecewise polynomial activations. The last reference, in particular, established for that case an upper bound of $O(w^2)$, where, as before, w is the number of weights. However it was an open problem (specifically, “open problem number 7” in the recent survey [11]) if there is a matching w^2 lower bound for such networks, and more generally for arbitrary continuous-activation nets. It could have been the case that the upper bound $O(w^2)$ is merely an artifact of the method of proof in [7], and that reliable learning with continuous-activation networks is still possible with far smaller sample sizes, proportional to $O(w \log w)$. But this is not the case, and in this paper we answer Maass’ open question in the affirmative.

Assume given an activation σ which has different limits at $\pm\infty$, and is such that there is at least one point where it has a derivative and the derivative is nonzero (this last condition rules out the Heaviside activation). Then there are architectures with arbitrary large numbers of weights w and VC dimension proportional to w^2 . The proof relies on first showing that networks consisting of two types of activations, Heavisides and linear, already have this power. This is a somewhat surprising result, since purely linear networks result in VC dimension proportional to w , and purely threshold nets have, as per the results quoted above, VC dimension bounded by $w \log w$. Our construction was originally motivated by a related one, given in [7], which showed that real-number programs (in the Blum-Shub-Smale model of computation) [4] with running time T have VC dimension $\Omega(T^2)$. The desired result on continuous activations is then obtained, approximating Heaviside gates by σ -nets with large weights and approximating linear gates by σ -nets with small weights. This result applies in particular to the standard sigmoid $1/(1 + e^{-x})$. (However, in contrast with the piecewise-polynomial case, there is still in that case a large gap between our $\Omega(w^2)$ lower bound and the $O(w^4)$ upper bound which was recently established in [8].) A number of variations, dealing with Boolean inputs, or weakening the assumptions on σ , are also discussed. The last section includes some brief remarks regarding an interpretation of our results in terms of threshold-only networks with “shared” weights.

Basic Terminology and Definitions

It is possible to formulate a general definition of “network architecture” that allows for very arbitrary nets; see [11]. However, in order to streamline the presentation we will only provide a simpler definition which is sufficient for dealing with first-order (additive-synapse) nets. (At one point we do need to deal technically with product units, but we treat that case in an ad-hoc manner.)

Formally, a (first-order, feedforward) *architecture* or *network* \mathcal{A} is a connected directed acyclic graph together with an assignment of a function to a subset of its nodes. The nodes are of two types: those of fan-in zero are called *input nodes* and the remaining ones are called *computation nodes* or *gates*. An *output node* is a node of fan-out zero. To each gate g there is associated a function $\sigma_g : \mathbb{R} \rightarrow \mathbb{R}$, called the *activation* or *gate function* associated to g .

The *number of weights* or *parameters* associated to a gate g is the integer n_g equal to the fan-in of g plus one. (This definition is motivated by the fact that each input to the gate will be multiplied by a weight, and the results are added together with a “bias” constant term, seen as one more weight; see below.) The (*total*) *number of weights* (or parameters) of \mathcal{A} is by definition the sum of the numbers n_g , over all the gates g of \mathcal{A} . The *number of inputs* m of \mathcal{A} is the total number of input nodes (one also says that “ \mathcal{A} has inputs in \mathbb{R}^m ”); it is assumed that $m > 0$. The *number of outputs* p of \mathcal{A} is the number of output nodes (unless otherwise mentioned, we assume by default that all nets considered have one-dimensional outputs, that is, $p = 1$).

Two examples of gate functions that are of particular interest are the identity or *linear* gate: $\text{Id}(x) = x$ for all x , and the *threshold* or *Heaviside* function: $H(x) = 1$ if $x \geq 0$, $H(x) = 0$ if $x < 0$.

If \mathcal{G} is some set of functions $\mathbb{R} \rightarrow \mathbb{R}$ so that, for each gate g of \mathcal{A} , the function $\sigma_g \in \mathcal{G}$, then we say that \mathcal{G} is a *set of gates* for \mathcal{A} . We use informal terminology as well; for instance if we say that \mathcal{A} consists (or is made of) of linear and threshold gates, we mean that $\mathcal{G} = \{H, \text{Id}\}$ is a set of gates for \mathcal{A} .

Let \mathcal{A} be an architecture. Assume that nodes of \mathcal{A} have been linearly ordered as $\pi_1, \dots, \pi_m, g_1, \dots, g_l$, where the π_j 's are the input nodes and the g_j 's the gates. For simplicity, write $n_i := n_{g_i}$, for each $i = 1, \dots, l$. Note that the total number of parameters is $n = \sum_{i=1}^l n_i$ and the fan-in of each g_i is $n_i - 1$. To each architecture \mathcal{A} (strictly speaking, an architecture together with such an ordering of nodes) we associate a function

$$F : \mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^p,$$

where p is the number of outputs of \mathcal{A} , defined by first assigning an “output” to each node, recursively on the distance from the the input nodes. Assume given an input $x \in \mathbb{R}^m$ and a vector of weights $w \in \mathbb{R}^n$. We partition w into blocks (w_1, \dots, w_l) of sizes n_1, \dots, n_l respectively. First the coordinates of x are assigned as the outputs of the input nodes π_1, \dots, π_m respectively. For each of the other gates g_i , we proceed as follows. Assume that outputs y_1, \dots, y_{n_i-1} have already been assigned to the predecessor nodes of g_i (these are input and/or computation nodes, listed consistently with the order fixed in advance). Then the output of g_i is by definition

$$\sigma_{g_i}(w_{i,0} + w_{i,1}y_1 + w_{i,2}y_2 + \dots + w_{i,n_i-1}y_{n_i-1}),$$

where we are writing $w_i = (w_{i,0}, w_{i,1}, w_{i,2}, \dots, w_{i,n_i-1})$. The value of $F(x, w)$ is then by definition the vector (scalar if $p = 1$) obtained by listing the outputs of the output nodes (in the agreed-upon fixed ordering of nodes). We call F the *function computed by the architecture* \mathcal{A} . For each choice of weights $w \in \mathbb{R}^n$, there is a function $F_w : \mathbb{R}^m \rightarrow \mathbb{R}^p$ defined by $F_w(x) := F(x, w)$; by abuse of terminology we sometimes call this also the function computed by \mathcal{A} (if the weight vector has been fixed).

Assume that \mathcal{A} is an architecture with inputs in \mathbb{R}^m and scalar outputs, and that the (unique) output gate has range $\{0, 1\}$. A subset $A \subseteq \mathbb{R}^m$ is said to be *shattered* by \mathcal{A} if for each Boolean function $\beta : A \rightarrow \{0, 1\}$ there is some weight $w \in \mathbb{R}^n$ so that $F_w(x) = \beta(x)$ for all $x \in A$. The *Vapnik-Chervonenkis (VC) dimension* of \mathcal{A} is the maximal size of a subset $A \subseteq \mathbb{R}^m$ that is shattered by \mathcal{A} . If the output gate can take non-binary values, we implicitly assume that the result of the computation is obtained by comparing the output to a fixed threshold θ . That is, when we say that a subset $A \subseteq \mathbb{R}^m$ is shattered by \mathcal{A} , we really mean that there exists $\theta \in \mathbb{R}$ such that A is shattered by the architecture $H(\mathcal{A} - \theta)$ in which the output of \mathcal{A} is decreased by an amount of θ , and then fed to a sign gate. Hence we actually construct architectures with quadratic *pseudo-dimension* (and in fact with quadratic “V-dimension” since the same threshold is used for all points in the shattered set; see [1] for definitions of these and other generalizations of the VC dimension).

The above formalism is too cumbersome for all proofs, so we often use obvious shortcuts. For instance, if we say “ \mathcal{A} is the net $w_0 + w_1 H(2x - 1)$ ” we really mean that this expression represents the scalar function computed by the obvious net with one input (the variable x) and two gates (one Heaviside, one linear); note that the number of weights is 4 (the weights are $2, -1, w_0, w_1$).

Another convention, consistent with standard computer science usage, is that we may use a phrase like “for each $n \geq 1$ there is an architecture \mathcal{A} with $O(\sqrt{n})$ weights and gates in \mathcal{G} ” to assert the existence of a sequence of architectures \mathcal{A}_n so that \mathcal{G} is a set of gates for each \mathcal{A}_n and so that the number of weights of \mathcal{A}_n is $O(\sqrt{n})$ as $n \rightarrow \infty$. In this context, when we say that \mathcal{A} shatters a set of size $\theta(n)$ we really mean that there is a sequence of sets A_n so that \mathcal{A}_n shatters A_n and the cardinality of each A_n is $\theta(n)$.

2 Networks Made up of Linear and Threshold Gates

Proposition 1 *For every $n \geq 1$, there is a network architecture \mathcal{A} with inputs in \mathbb{R}^2 and $O(\sqrt{N})$ weights that can shatter a set of size $N = n^2$. This architecture is made only of linear and threshold gates.*

Proof. Our architecture has n parameters W_1, \dots, W_n ; each of them is an element of $T = \{0.w_1 \dots w_n; w_i \in \{0, 1\}\}$. The shattered set will be $S = [n]^2 = \{1, \dots, n\}^2$.

For a given choice of $W = (W_1, \dots, W_n)$, \mathcal{A} will compute the boolean function $f_W : S \rightarrow \{0, 1\}$ defined as follows: $f_W(x, y)$ is equal to the x -th bit of W_y . Clearly, for any boolean function f on S , there exists a (unique) W such that $f = f_W$.

We first consider the obvious architecture which computes the function:

$$f_W^1(y) = W_1 + \sum_{z=2}^n (W_z - W_{z-1})H(y - z + 1/2) \quad (1)$$

sending each point $y \in [n]$ to W_y . This architecture has $n - 1$ threshold gates, $3(n - 1) + 1$ weights, and just one linear gate.

Next we define a second multi-output net which maps $w \in T$ to its binary representation $f^2(w) = (w_1, \dots, w_n)$. Assume by induction that we have a net \mathcal{N}_i^2 that maps w to $(w_1, \dots, w_i, 0.w_{i+1} \dots w_n)$. Since $w_{i+1} = H(0.w_{i+1} \dots w_n - 1/2)$ and $0.w_{i+2} \dots w_n = 2 \times 0.w_{i+1} \dots w_n - w_{i+1}$, \mathcal{N}_{i+1}^2 can be obtained by adding one threshold gate and one linear gate to \mathcal{N}_i^2 (as well as 4 weights). It follows that \mathcal{N}_n^2 has n threshold gates, n linear gates and $4n$ weights.

Finally, we define a net \mathcal{N}^3 which takes as input $x \in [n]$ and $w = (w_1, \dots, w_n) \in \{0, 1\}^n$, and outputs w_x . We would like this network to be as follows:

$$f^3(x, w) = w_1 + \sum_{z=2}^n w_z H(x - z + 1/2) - \sum_{z=2}^n w_{z-1} H(x - z + 1/2).$$

This is not quite possible, because the products between the w_i 's (which are inputs in this context) and the Heavisides are not allowed. However, since we are dealing with binary variables one can write $uv = H(u + v - 1.5)$. Thus \mathcal{N}_3 has one linear gate, $4(n - 1)$ threshold gates and $12(n - 1) + n$ weights. Note that $f_W(x, y) = f^3(x, f^2(f_W^1(y)))$. This can be realized by means of a net that has $n + 2$ linear gates, $(n - 1) + n + 4(n - 1) = 6n - 5$ threshold gates, and $(3n - 2) + 4n + (12n - 11) = 19n - 13$ weights. \square

The following is the main result of this section:

Theorem 1 *For every $n \geq 1$, there is a network architecture \mathcal{A} with inputs in \mathbb{R} and $O(\sqrt{N})$ weights that can shatter a set of size $N = n^2$. This architecture is made only of linear and threshold gates.*

Proof. The shattered set will be $S = \{0, 1, \dots, n^2 - 1\}$. For every $x \in S$, there are unique integers $x, y \in \{0, 1, \dots, n - 1\}$ such that $x = ny + y$. The idea of the construction is to compute x and y , and then feed $(x + 1, y + 1)$ to the network constructed in Proposition 1. Note that x is the unique integer such that $u - nx \in \{0, 1, \dots, n - 1\}$. It can therefore be computed by brute force search as follows:

$$x = \sum_{k=0}^{n-1} kH[H(u - nk) + H(n - 1 - (u - nk)) - 1.5].$$

This network has $3n$ threshold gates, one linear gate and $8n$ weights. Then of course $y = u - nx$. \square

A Boolean version is as follows.

Theorem 2 *For every $d \geq 1$, there is a network architecture \mathcal{A} with $O(\sqrt{N})$ weights that can shatter the $N = 2^{2d}$ points of $\{0, 1\}^{2d}$. This architecture is made only of linear and threshold gates.*

Proof. Given $u \in \{0, 1\}^{2d}$, one can compute $x = 1 + \sum_{i=1}^d 2^{i-1}u_i$ and $y = 1 + \sum_{i=1}^d 2^{i-1}u_{i+d}$ with two linear gates. Then (x, y) can be fed to the network of Proposition 1 (with $n = 2^d$). \square

In other words, there is a network architecture with 2^d weights that can compute all boolean functions on $2d$ variables.

2.1 Constant Number of Linear Gates

We do not know whether similar constructions using only a constant number of linear gates are possible. However, by making a distinction between “fixed weights” and “programmable weights” one can prove a result in that direction. Given a sequence of architectures \mathcal{A}_j , with numbers of weights $n^{(j)}$ respectively, assume that some partition of the indices $i = 1, \dots, n^{(j)}$ into two subsets $S_f^{(j)}$ and $S_p^{(j)}$ has been given. The cardinality of the first set is then called the number of fixed weights and the cardinality of $S_p^{(j)}$ is the number of programmable weights of the sequence of architectures. For each j we then have a function $F^{(j)}(x, v)$ obtained by fixing the coordinates in $S_f^{(j)}$ to some values while allowing the remaining coordinates (collected into the vector v) to be variable. Shattering and VC dimension are then defined in terms of these restricted functions.

Theorem 3 *For every $n \geq 1$, there is a network architecture \mathcal{A} with inputs in \mathbb{R}^2 , $O(\sqrt{N})$ programmable weights and $O(N)$ fixed weights that can shatter a set of size $N = n^2$. This architecture is made only of $O(\sqrt{N})$ threshold gates and of a constant number of linear gates.*

Proof. Note that \mathcal{N}_n^2 is the only part of the network constructed in Proposition 1 that uses a non-constant number of linear gates. These gates carry intermediate results of the form $0.w_{i+1} \dots w_n$. However, this number can be computed “from scratch” using the input $w \in T$ and the previously computed bits w_1, \dots, w_i . More precisely the next bit can be computed as follows: $w_{i+1} = H[2^i(w - \sum_{j=1}^i 2^{-j} w_j) - 1/2]$. The number of weights in this new construction is $\sum_{i=1}^n (i+2) = O(n^2)$, and none of them is programmable. This replacement for \mathcal{N}^2 has n threshold gates. So the total number of gates needed in this alternative construction is $6n - 5$ threshold gates and two linear gates. \square

3 Arbitrary Sigmoids

We now extend the preceding VC dimension bounds to networks that use just one activation function σ (instead of both linear and threshold gates). All that is required is that the gate function have a sigmoidal shape and satisfy a very weak smoothness property:

1. σ is differentiable at some point x_0 (i.e., $\sigma(x_0 + h) = \sigma(x_0) + \sigma'(x_0)h + o(h)$) where $\sigma'(x_0) \neq 0$.
2. $\lim_{x \rightarrow -\infty} \sigma(x) = 0$ and $\lim_{x \rightarrow +\infty} \sigma(x) = 1$ (the limits 0 and 1 can be replaced by any distinct numbers).

A function satisfying these two conditions will be called *sigmoidal*. Given any such σ , we will show that networks using only σ gates provide quadratic VC dimension.

Theorem 4 *Let σ be an arbitrary sigmoidal function. There exist architectures \mathcal{A}_1 and \mathcal{A}_2 with $O(\sqrt{N})$ weights made only of σ gates such that:*

- \mathcal{A}_1 can shatter a subset of \mathbb{R} of cardinality $N = n^2$;
- \mathcal{A}_2 can shatter the $N = 2^{2d}$ points of $\{0, 1\}^{2d}$.

This follows directly from Theorems 1 and 2, together with the following simulation result:

Theorem 5 *Let σ be an arbitrary sigmoidal function. Let \mathcal{N} be a network of T threshold and L linear gates, with a threshold gate at the output. Then \mathcal{N} can be simulated on any given finite set of inputs by a network \mathcal{N}' of $T + L$ gates that all use the activation function σ (except the output gate which is still a threshold). Moreover, if \mathcal{N} has n weights then \mathcal{N}' has $O(n)$ weights.*

Proof. Let S be a finite set of inputs. We can assume, by changing the thresholds of threshold gates if necessary, that the net input $I_g(x)$ to any threshold gate g of \mathcal{N} is different from 0 for all inputs $x \in S$.

Given $\epsilon > 0$, let \mathcal{N}_ϵ be the net obtained by replacing the output functions of all gates by the new output function $x \mapsto \sigma(x/\epsilon)$ if this output function is the sign function, and by $x \mapsto \sigma_\epsilon(x) = [\sigma(x_0 + \epsilon x) -$

$\sigma(x_0)]/[\epsilon\sigma'(x_0)]$ if it is the identity function. Note that for any $a > 0$, $\lim_{\epsilon \rightarrow 0^+} \sigma(x/\epsilon) = H(x)$ uniformly for $x \in]-\infty, -a] \cup [a, +\infty[$ and $\lim_{\epsilon \rightarrow 0} \sigma_\epsilon(x) = x$ uniformly for $x \in [-1/a, 1/a]$.

This implies by induction on the depth of g that for any gate g of \mathcal{N} and any input $x \in S$, the net input $I_{g,\epsilon}(x)$ to g in the transformed net \mathcal{N}_ϵ satisfies $\lim_{\epsilon \rightarrow 0} I_{g,\epsilon}(x) = I_g(x)$ (here, we use the fact that the output function of every g is continuous at $I_g(x)$). In particular, by taking g to be the output gate of \mathcal{N} , we see that \mathcal{N} and \mathcal{N}_ϵ compute the same function on S if ϵ is small enough. Such a net \mathcal{N}_ϵ can be transformed into an equivalent net \mathcal{N}' that uses only σ as gate function by a simple transformation of its weights and thresholds. The number of weights remains the same, except at most for a constant term that must be added to each net input to a gate; thus if \mathcal{N} has n weights, \mathcal{N}' has at most $2n$ weights. \square

4 More General Gate Functions

The objective of this section is to establish results similar to Theorem 4, but for even more arbitrary gate functions, in particular weakening the assumption that limits exist at infinity. The main result is, roughly, that any σ which is piecewise twice (continuously) differentiable gives at least quadratic VC dimension, save for certain exceptional cases involving functions that are almost everywhere linear.

In deriving the results of this section, we find it useful to allow networks with multiplication and division gates, which strictly speaking are not networks in the way that we have defined the concept. By this we will mean that the definition of architecture is extended so that the symbols “ $*$ ” and “ $/$ ” are also allowed as labels for gates. The gates labeled “ $/$ ” have fan-in two and number of weights also two (even though there is no natural numerical parameter associated to the gate; we need to assign weights to multiplication and division gates to account for the numerical parameters that will occur when simulating these gates by σ -gates); the output of such a gate is defined as the quotient of its two inputs, assuming that the second input is nonzero. The multiplication gates may have arbitrary fan-in, and “number of weights” equal to this fan-in; their output is defined as the product of the inputs. An input to a circuit is said to be *valid* if it does not cause a division by zero at any division gate. We will only work with sets of valid inputs (so the domain of the function computed by such a generalized network is a subset of \mathbb{R}^m and shattering is only defined for subsets of this domain).

We may work indifferently with multiplication gates of fan-in 2 or of unbounded fan-in. The number of weights is unchanged up to a constant factor since a k -ary multiplication $x_1 \dots x_k$ can be replaced for $k > 2$ by a depth- $(k-1)$ circuit $x_1(x_2(x_3(\dots x_n))) \dots$ of binary gates with $2k-2$ weights.

Theorem 6 *For every $n \geq 1$, there is a network architecture with inputs in \mathbb{R}^2 and $O(\sqrt{N})$ weights that can shatter a set of size $N = n^2$. This architecture is made only of linear, multiplication, and division gates.*

The construction is based on that of Proposition 1. In particular, the shattered is still $S = [n]^2$. We first show how to interpolate approximately.

Lemma 1 *For every $n \geq 1$, there is an architecture \mathcal{A}'_1 with inputs (x, W_1, \dots, W_n) in \mathbb{R}^{n+1} and $O(n)$ weights such that the following property holds: for every $\epsilon > 0$ there exists a choice of the weights of \mathcal{A}'_1 such that the function f'_ϵ implemented by the network satisfies $\lim_{\epsilon \rightarrow 0} f'_\epsilon(i, W_1, \dots, W_n) = W_i$ for $i = 1, \dots, n$.*

Proof. Let us consider the function

$$f'_\epsilon(x, W_1, \dots, W_n) = \prod_{i=1}^n (x - \epsilon - i) \cdot \sum_{i=1}^n \frac{a_i W_i}{x - \epsilon - i}$$

where $a_i = 1/\prod_{j \neq i} (i-j)$ and $\epsilon \neq 0$. It can be implemented by an architecture made of one multiplication gate with $n+1$ inputs, one linear gate with n inputs, n division gates, and n linear gates with one input each (they compute the terms $x - \epsilon - i$). The total number of weights is thus $(n+1) + n + 2n + 2n = 6n+1$. Whenever f'_ϵ is defined,

$$f'_\epsilon(x, W_1, \dots, W_n) = \sum_{i=1}^n W_i \cdot \frac{\prod_{j \neq i} (x - \epsilon - j)}{\prod_{j \neq i} (i - j)}.$$

Hence $\lim_{\epsilon \rightarrow 0} f'_\epsilon(i, W_1, \dots, W_n) = W_i$. \square

Lemma 2 *There exists an architecture of linear and multiplication gates with inputs in \mathbb{R} , n output units and $O(n)$ weights such that the following property holds: for every $\epsilon \in \{0, 1\}^n$, there exists an input $w \in [0, 1]$ such that the output of the network $f'^2(w) = (f'^2(w)_1, \dots, f'^2(w)_n)$ of the network satisfies $f'^2(w)_i \in [0, 1/2[$ if $\epsilon_i = 0$, and $f'^2(w)_i \in]1/2, 1]$ if $\epsilon_i = 1$.*

Proof. The construction is based on a simple idea from symbolic dynamics. Consider the logistic map $\phi : [0, 1] \rightarrow [0, 1]$ such that $\phi(x) = 4x(1 - x)$. We claim that for every $\epsilon \in \{0, 1\}^n$, there exists $w \in [0, 1]$ such that $\phi^{i-1}(w) \in [0, 1/2[$ if $\epsilon_i = 0$, and $\phi^{i-1}(w) \in]1/2, 1]$ if $\epsilon_i = 1$. The result follows from this claim, using the iterates $\phi^{i-1}, i = 1, \dots, n$ as the coordinates of f'^2 , since the logistic map can be implemented by a subnetwork of linear and multiplication gates.

The proof of the claim is as follows. First, note that each element of $[0, 1]$ has two distinct preimages by ϕ , except 1; and that $\phi(1/2) = 1$, $\phi(1) = 0$, and $\phi(0) = 0$. If $\epsilon_n = 0$, choose an element w_n in $]0, 1/2[$; otherwise, choose w_n in $]1/2, 1[$. We construct a sequence w_1, \dots, w_{n-1} by “going backward in time” as follows: w_i is defined to be the preimage of w_{i+1} which is in $]0, 1/2[$ if $\epsilon_i = 0$, and the preimage which is in $]1/2, 1[$ otherwise. By construction, one can take $w = w_1$. \square

The networks constructed in the two previous lemmas will now be used as building blocks in the proof of Theorem 6.

Proof of Theorem 6. Let f be an arbitrary boolean function on $[n]^2$. Let $W = (W_1, \dots, W_n)$ be a sequence of inputs to the circuit f'^2 of Lemma 2 such that $H(f'^2(W_j)_i - 1/2) = f(i, j)$ for $i, j \in [n]$.

Consider the map $\mathcal{N}_\epsilon : (x, y) \mapsto f'_\epsilon(x, f'^2(f'_\epsilon(y, W))_1, \dots, f'^2(f'_\epsilon(y, W))_n)$. It can be implemented by a network of $O(n)$ weights where the output of f'_ϵ is fed to f'^2 and the n outputs of f'^2 are then fed (together with x) to f'_ϵ . By Lemma 1, $\lim_{\epsilon \rightarrow 0} f'_\epsilon(j, W) = W_j$. By continuity of f'^2 , when ϵ is small enough $f'^2(f'_\epsilon(j, W))_i < 1/2$ if $f(i, j) = 0$ and $f'^2(f'_\epsilon(j, W))_i > 1/2$ if $f(i, j) = 1$. Hence it follows from Lemma 1 that when ϵ is small enough, $\mathcal{N}_\epsilon(i, j) < 1/2$ if $f(i, j) = 0$ and $\mathcal{N}_\epsilon(i, j) > 1/2$ if $f(i, j) = 1$. The boolean function f can thus be computed by comparing the output of \mathcal{N}_ϵ to $1/2$. \square

We now explain how to get rid of division gates.

Theorem 7 *Let \mathcal{N} be a network made of linear, multiplication and division gates, with one threshold gate at the output and w weights. Then \mathcal{N} can be simulated on its set of valid inputs by a network \mathcal{N}' of linear and multiplication gates with one threshold gate at the output and $w' = O(w)$ weights.*

Proof. Let us assume without loss of generality that all linear and multiplication gates of \mathcal{N} are binary, and that the output gate is unary. Each non-output gate g of \mathcal{N} will be replaced in \mathcal{N}' by two gates g_1 and g_2 such that on any valid input of \mathcal{N} , the values assumed by these 3 gates satisfy $g(x) = g_1(x)/g_2(x)$. Each input to the new gates g_1 and g_2 is now a pair (x_1, x_2) , representing the corresponding input x_1/x_2 to the original gate g . The outputs of g_1 and g_2 are passed as a pair to the next gates in the graph. The rules for forming g_1 and g_2 from each g follow these three simple rules:

1. $a.x_1/x_2 + b.y_1/y_2 + c = (ax_1y_2 + bx_2y_1 + cx_2y_2)/(x_2y_2)$ (simulation of linear gates);
2. $(x_1/x_2).(y_1/y_2) = (x_1y_1)/(x_2y_2)$ (simulation of multiplication gates);
3. $(x_1/x_2)/(y_1/y_2) = (x_1y_2)/(x_2y_1)$ (simulation of division gates).

Finally, a linear gate and a multiplication gate are added before taking the sign at the output since

$$H(x_1/x_2 - \theta) = H(x_2(x_1 - \theta x_2))$$

whenever $x_2 \neq 0$. \square

Corollary 1 *For every $n \geq 1$, there is a network architecture with inputs in \mathbb{R}^2 and $O(\sqrt{N})$ weights that can shatter a set of size $N = n^2$. This architecture is made only of linear and multiplication gates.*

Proof. Follows immediately from Theorems 6 and 7. \square

A function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is said to be *piecewise C^2* if there is a finite sequence $a_1 < a_2 < \dots < a_p$ such that on each interval I of the form $] -\infty, a_1[$, $]a_i, a_{i+1}[$ or $]a_p, +\infty[$, $\sigma|_I$ is C^2 .

(Note: our results hold even if it is only assumed that the second derivative exists in each of the above intervals; we do not use the continuity of these second derivatives.)

Theorem 8 *Let σ be a piecewise C^2 function. For every $n \geq 1$, there exists an architecture made of σ -gates, and with $O(n)$ weights, that can shatter a subset of \mathbb{R}^2 of cardinality n^2 , except perhaps in the following cases:*

1. σ is piecewise-constant, and in this case the VC dimension of any architecture of n weights is $O(n \log n)$;
2. σ is affine, and in this case the VC dimension of any architecture of n weights is at most n .
3. there are constants $a \neq 0$ and b such that $\sigma(x) = ax + b$ except at a finite nonempty set of points. In this case, the VC dimension of any architecture of n weights is $O(n^2)$, and there are architectures of VC dimension $\Omega(n \log n)$.

Note that the upper bound of the first special case is tight for threshold nets, and that of the second special case is tight for linear functions in \mathbb{R}^n .

The proof of Theorem 8 is broken down into several steps. The following result deals with the most interesting case.

Lemma 3 *Assume that σ is piecewise C^2 and that there exists a point $x_1 \notin \{a_1, \dots, a_p\}$ where $\sigma''(x_1) \neq 0$. For every $n \geq 1$, there exists an architecture of σ -gates with $O(n)$ weights that can shatter a subset of \mathbb{R}^2 of cardinality n^2 .*

Proof. The proof technique is similar to that of Theorem 5. We will show that an architecture consisting of linear and multiplication gates can be simulated on any finite set by an architecture of σ -gates with the same number of weights, up to a constant factor.

Let I be an open interval containing x_1 where σ is twice differentiable. There exists $x_0 \in I$ such that $\sigma'(x_0) \neq 0$: indeed, if $\sigma'(x) = 0$ for every $x \in I$, $\sigma''(x_1)$ would be equal to zero. Thus linear gates can be simulated using σ in a neighborhood of x_0 , just as in the proof of Theorem 5.

Product gates can first be replaced by s -gates, where $s(x) = x^2$ is the square function, since $xy = [(x+y)^2 - x^2 - y^2]/2$. An s -gate can be simulated by two σ -gates and a linear gate since for any $A > 0$,

$$\lim_{\epsilon \rightarrow 0} \frac{\sigma(x_1 + \epsilon x) + \sigma(x_1 - \epsilon x) - 2\sigma(x_1)}{\sigma''(x_1)\epsilon^2} = x^2$$

for all $x \in [-A, A]$ (in fact, uniformly on such intervals). Any such linear gate can be simulated by a σ -gate, as explained earlier. \square

If σ is piecewise C^2 but does not satisfy the hypotheses of Lemma 3, it is a piecewise-linear function. We conclude with a case-by-case analysis based on the slopes of its linear pieces. The first special case in Theorem 8 follows from the next trivial lemma and from the results of [3] for threshold nets.

Lemma 4 *Let σ be a piecewise-constant function. An architecture of σ -gates with n weights can be simulated by an architecture of threshold gates with $O(n)$ weights.*

Proof. Assume that $\sigma(x) = v_0$ for $x < a_1$ and $\sigma(x) = v_i$ for $x \in]a_i, a_{i+1}[$ (with the convention $a_{p+1} = +\infty$). A σ -gate can be replaced by a subnetwork made of a constant number of threshold gates by the following relation:

$$\sigma(x) = v_0 + \sum_{i=1}^p (v_i - v_{i-1})H(x - a_i) + \sum_{i=1}^p (\sigma(a_i) - v_i)(H(x - a_i) + H(-x + a_i) - 2).$$

Note that the role of the second sum is to compute the correct value of σ at the breakpoints a_1, \dots, a_p . \square

If σ is not piecewise-constant, there is a non-trivial (i.e., not reduced to one of the a_i 's) piece where $\sigma(x) = ax + b$ with $a \neq 0$. If this relation holds in fact over \mathbb{R} , σ is affine and we are in the second case of Theorem 8. In this case, the input-output mapping of the network is affine, so that the VC dimension is bounded by the number of weights (this observation goes back at least to 1965; see [6]).

If the relation $\sigma(x) = ax + b$ ($a \neq 0$) holds everywhere except at a nonempty finite number of points, we are in special case 3. The VC dimension of any architecture of n weights is $O(n^2)$ by [7] (that paper actually deals with arbitrary piecewise-polynomial gate functions). The lower bound is established in Lemmas 6 and 7.

The only remaining case is that in which there exist at least two non-trivial pieces, and in at least one σ is not constant. This leads again to quadratic VC dimension, as shown in the next lemma.

Lemma 5 *Let σ be a piecewise-linear function such that $\sigma(x) = \alpha x + \beta$ on a non-trivial interval I and $\sigma(x) = \alpha' x + \beta'$ on another non-trivial interval J , with $(\alpha, \beta) \neq (\alpha', \beta')$ and $(\alpha, \alpha') \neq (0, 0)$.*

For every $n \geq 1$, there exists an architecture of σ -gates with $O(n)$ weights that can shatter a subset of \mathbb{R}^2 of cardinality n^2 .

Proof. As in Lemma 3, the proof technique is similar to that of Theorem 5. We will show that an architecture consisting of linear and multiplication gates can be simulated on any finite set by an architecture of σ -gates with the same number of weights, up to a constant factor.

Assume for instance that $\alpha \neq 0$. Linear gates can be simulated using σ on I , just as in the proof of Theorem 5. The two intervals I and J can be taken adjacent, i.e., I can be assumed to be of the form $I =]a, b[$ and J of the form $J =]b, c[$. In order to simulate threshold gates, two cases can be distinguished.

Assume first that σ has two distinct limits at b , i.e., $\sigma(b^-) \neq \sigma(b^+)$. In this case, for any $A > 0$, $H(x) = \lim_{\epsilon \rightarrow 0^+} [\sigma(b + \epsilon x) - \sigma(b^-)] / [\sigma(b^+) - \sigma(b^-)]$ for all $x \in [-A, A] \setminus \{0\}$ (in fact, uniformly on this set). Thus we can simulate threshold gates as well.

Assume now that, instead, $\sigma(b^-) = \sigma(b^+)$. This implies that $\alpha \neq \alpha'$ since $(\alpha, \beta) \neq (\alpha', \beta')$. In this case, for any $A > 0$ and any $\delta > 0$,

$$H(x) = \lim_{\epsilon \rightarrow 0^+} \frac{\sigma(b + \epsilon x) - \sigma(b + \epsilon x - \epsilon^2) - \alpha \epsilon^2}{(\alpha' - \alpha) \epsilon^2}$$

(uniformly) for $x \in [-A, A] \setminus (0, \delta)$. Thus we can also simulate threshold gates in this case. \square

In order to deal with the last exceptional case in Theorem 8, we find it useful to introduce another auxiliary computation model, based on circuits of linear and equality gates. An equality gate has fan-in one and outputs $E(x) = 1$ when its input x is equal to 0; it outputs $E(x) = 0$ otherwise.

Lemma 6 *For every $n \geq 1$, there is a network architecture \mathcal{A} with inputs in \mathbb{R}^2 and $O(n)$ weights that can shatter a set of size $N = n \lfloor \log_2 n \rfloor$. This architecture is made only of linear and equality gates.*

Proof. The construction is similar to that of Proposition 1, and the reader is advised to start with that result. The shattered set is $S = \{1, \dots, k\} \times \{1, \dots, n\}$ where $k = \lfloor \log_2 n \rfloor$, instead of $\{1, \dots, n\}^2$. The output $f_W(x, y)$ of the network is still required to be the x -th bit of W_y . Hence we work with parameters W_1, \dots, W_n of the form $0.w_1 \dots w_k$ ($w_i \in \{0, 1\}$). The general structure of the network remains the same. In particular, the maps f_W^1 and f^3 can be computed with the following obvious modifications: $f_W^1(y) = \sum_{z=1}^n W_z E(y - z)$ and $f^3(x, w) = \sum_{z=1}^k w_z E(x - z)$ (products can be rewritten as follows: $uv = E(u + v - 2)$).

The extraction of the bits w_1, \dots, w_k is more interesting. Assume again by induction that we have a net \mathcal{N}_i^2 that maps w to $(w_1, \dots, w_i, 0.w_{i+1} \dots w_k)$. The next bit can be computed by exhaustive search as follows:

$$w_{i+1} = \sum_{z_{i+2}, \dots, z_k=0}^1 E(0.w_{i+1} \dots w_k - 0.1z_{i+2} \dots z_k).$$

Then of course $0.w_{i+2} \dots w_n = 2 \times 0.w_{i+1} \dots w_n - w_{i+1}$. The number of equality gates in \mathcal{N}_k^2 is $\sum_{i=0}^{k-1} 2^i = 2^k - 1 < n$. Hence the number of weights in the whole network is $O(n)$. \square

The following lemma completes the proof of Theorem 8 (note that we only need the direct implication).

Lemma 7 *Let σ be a real function of the form $\sigma(x) = \alpha x + \beta$ (with $\alpha \neq 0$) except at a finite number of points.*

A network of linear and equality gates with n weights can be simulated by a network of σ -gates with $O(n)$ gates. Conversely, a network of σ -gates with n weights can be simulated by a network of linear and equality gates with $O(n)$ gates.

Proof. As usual, a linear gate can be simulated using σ on any of its linear pieces.

Assume that $\sigma(x) = \alpha x + \beta$ except for $x \in \{a_1, \dots, a_p\}$. For any $A > 0$,

$$E(x) = \lim_{\epsilon \rightarrow 0} \frac{\sigma(a_1 + \epsilon x) - [\alpha(a_1 + \epsilon x) + \beta]}{\sigma(a_1) - (\alpha a_1 + \beta)}$$

(uniformly) for $x \in [-A, A]$.

Conversely, a σ -gate can be simulated with a constant number of linear and equality gates by the following identity:

$$\sigma(x) = \alpha x + \beta + \sum_{i=1}^p [\sigma(a_i) - (\alpha a_i + \beta)] E(x - a_i).$$

□

5 Some Remarks on Threshold Nets with “Shared” Weights

In Section 2, we showed that we can get VC dimension n^2 when using a mixture of $O(n)$ threshold and linear gates, compared to the known upper bound of $O(n \log n)$ which would hold if only threshold gates were allowed. The gain involved in adding linear gates would seem to be counterintuitive, since it is obviously possible to rewrite a network made up of linear and threshold gates as a network made exclusively of threshold gates. The explanation of this apparent paradox is that, when rewriting in this manner, the number of weights becomes as high as $O(n^2)$. The resulting weights are “shared” among gates. But such a sharing arrangement is not allowed in our definitions, and indeed, cannot be exploited when using standard Cover-like counting arguments, as in [6, 3], which are dependent upon the fan-in of gates.

As a simple illustration of this process, consider a way of rewriting the network obtained in Proposition 1 as a network which only employs threshold gates. We sketch next how one eliminates linear gates in that case.

First of all, the function f_W^1 in Equation (1) does not need to be itself computed as a linear function. Instead, the $n - 1$ values $H(y - 2 + 1/2), H(y - 3 + 1/2), \dots$ are first computed by a layer of threshold gates. Write $\vec{H}(y)$ for the $n - 1$ -dimensional column vector that lists these values, and write $C = (W_2 - W_1, \dots, W_n - W_{n-1})$ and $d = W_1$. Thus, $f_W^1(y) = C\vec{H}(y) + d$, but we do not compute this value just yet. Then the coordinates of the composition $f^2(f_W^1(y))$ can be computed recursively, starting with $\vec{H}(y)$: the value for any given y is the vector (w_1, \dots, w_n) , where $w_1 = H(C\vec{H}(y) + d)$, $w_2 = H(C\vec{H}(y) - w_1/2 + (2d - 1/2)/2)$, and so forth. Finally, we can replace f^3 by its thresholded value. This construction does not involve any linear gates. Each entry of C occurs in the gates computing w_1, \dots, w_n , but these n instances can be considered as a single instance of a “shared weight”.

We conclude that it is possible to shatter the same set as in Proposition 1 by means of an architecture of threshold gates with $O(n^2)$ “non-programmable” (i.e., constant) weights and only $O(n)$ “programmable” shared weights (the entries of C and affine functions of d). Note that without weight sharing, the VC dimension of a threshold network with n programmable weights remains $O(n \log n)$ by the counting argument of [3].

A more restrictive type of weight-sharing has been studied in the neural network literature, and proved to be useful in invariant recognition tasks [9]. A formal model is studied in [13], and it is shown that the VC dimension remains $O(n \log n)$. In this model one assumes that there is an equivalence relation between weights; this is similar to our weight-sharing mechanism. However, one also assumes that there is an equivalence relation on nodes, and that this relation is compatible (in a precise sense) with the equivalence relation on weights. This makes the model more restrictive, explaining the smaller VC dimension.

References

- [1] N. ALON, S. BEN-DAVID, N. CESA-BIANCHI AND D. HAUSSLER, *Scale-sensitive dimensions, uniform convergence, and learnability*, in Proc. 34th IEEE Symp. on Foundations of Computer Science, 1993, pp. 292-301.
- [2] M. ANTHONY AND N.L. BIGGS, *Computational Learning Theory: An Introduction*, Cambridge U. Press, 1992.
- [3] E.B. BAUM AND D. HAUSSLER, *What size net gives valid generalization?*, Neural Computation, 1 (1989), pp. 151-160.
- [4] L. BLUM, M. SHUB AND S. SMALE, *On the theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines*, Bulletin of the AMS, 21 (1989), pp. 1-46.
- [5] A. BLUMER, A. EHRENFEUCHT, D. HAUSSLER, AND M. WARMUTH, *Learnability and the Vapnik-Chervonenkis dimension*, J. of the ACM, 36 (1989), pp. 929-965.
- [6] T.M. COVER, *Capacity problems for linear machines*, in: Pattern Recognition, L. Kanal ed., Thompson Book Co., 1988, pp. 283-289.
- [7] P. GOLDBERG AND M. JERRUM, *Bounding the Vapnik-Chervonenkis dimension of concept classes parametrized by real numbers*, Machine Learning, 18 (1995), pp. 131-148.
- [8] M. KARPINSKI AND A. MACINTYRE, *Polynomial bounds for VC dimension of sigmoidal neural networks*, in Proc. 27th ACM Symposium on Theory of Computing, 1995, pp. 200-208.
- [9] K. LANG AND G.E. HINTON, *The development of TDNN architecture for speech recognition*, Technical Report CMU-CS-88-152, Carnegie-Mellon University, 1988.
- [10] W. MAASS, *Bounds for the computational power and learning complexity of analog neural nets*, in Proc. 25th ACM Symposium on Theory of Computing, 1993, pp. 335-344.
- [11] W. MAASS, *Perspectives of current research about the complexity of learning in neural nets*, in *Theoretical Advances in Neural Computation and Learning*, V.P. Roychowdhury, K.Y. Siu, and A. Orlitsky, editors, Kluwer, Boston, 1994, pp. 295-336.
- [12] B.K. NATARAJAN, *Machine Learning : A Theoretical Approach*, Morgan Kaufmann Publishers, San Mateo, CA, 1991.
- [13] J. SHAWE-TAYLOR, *Threshold network learning in the presence of equivalences*, in Advances in Neural Information Processing Systems 4, Morgan Kaufmann Publishers, 1992, pp. 879-886.
- [14] E.D. SONTAG, *Sigmoids distinguish better than Heavisides*, Neural Computation, 1 (1989), pp. 470-472.
- [15] E.D. SONTAG, *Feedforward nets for interpolation and classification*, J. Comp. Syst. Sci, 45 (1992), pp. 20-48.
- [16] L.G. VALIANT, *A theory of the learnable*, Comm. of the ACM, 27, 1984, pp. 1134-1142
- [17] V.N. VAPNIK, *Estimation of Dependencies Based on Empirical Data*, Springer, Berlin, 1982.