Neural Nets with Superlinear VC-Dimension

Wolfgang Maass

Institute for Theoretical Computer Science
Technische Universitaet Graz
Klosterwiesgasse 32/2
A-8010 Graz, Austria
e-mail: maass@igi.tu-graz.ac.at

Abstract

It has been known for quite a while that the Vapnik-Chervonenkis dimension (VC-dimension) of a feedforward neural net with linear threshold gates is at most $O(w \cdot \log w)$, where w is the total number of weights in the neural net. We show in this paper that this bound is in fact asymptotically optimal. More precisely, we construct for arbitrarily large $w \in \mathbf{N}$ neural nets $\mathcal{N}_w$ of depth 3 (i.e. with 2 layers of hidden units) that have VC-dimension $\Omega(w \cdot \log w)$. The construction exhibits a method that allows us to encode more "program-bits" in the weights of a neural net than previously thought possible.

The Vapnik-Chervonenkis-dimension (abbreviated: VC-dimension) of a neural net $\mathcal{N}$ is an important measure of the expressiveness of $\mathcal{N}$, i.e. for the variety of functions that can be computed by $\mathcal{N}$ with different choices for its weights. In particular it has been shown in [BEHW] and [EHKV] that the VC-dimension of $\mathcal{N}$ essentially determines the number of training examples that are needed to train $\mathcal{N}$ in Valiant's model [V] for probably approximately correct learning ("PAC-learning").

It has been known for quite a while that the VC-dimension of a neural net with linear threshold gates and w edges (respectively w weights) is at most $O(w \cdot \log w)$. This was first shown by Cover in [C 64] (see also [C 88]), and later by Baum and Haussler [BH]. It has frequently been conjectured that the "true" upper bound is $O(w)$. This conjecture is quite plausible, since a single linear threshold gate with w edges has VC-dimension $w + 1$. Furthermore it is hard to imagine that the VC-dimension of a network of linear threshold gates can be larger than the sum of the VC-dimensions of the individual linear threshold gates in the network.

We disprove this popular conjecture by constructing neural nets $\mathcal{N}_w$ whose VC-dimension is superlinear in the number w of edges in $\mathcal{N}_w$. More precisely, we construct for arbitrarily large $w \in \mathbf{N}$ neural nets $\mathcal{N}_w$ of depth 3 (i.e. with 2 hidden layers) with w weights that have VC-dimension $\Omega(w \cdot \log w)$. This shows that the quoted upper bound of $O(w \log w)$ is in fact asymptotically optimal.

The result of this paper may also be viewed as mathematical evidence for a certain type of "connectionism thesis": that a network of neuron-like elements is more than just the sum of its elements. We show that in a large neural net a single edge may add more than a constant to the VC-dimension of the neural net. Futhermore its contribution may increase with the total size of the neural net.

Although we consider in this paper only neural nets with linear threshold gates, it is obvious that the same lower bound can also be derived for neural nets with other activation functions such as $\sigma(y) = \frac{1}{1+e^{-y}}$ (see [RM]) or piecewise linear (resp. polynomial) functions of a similar type (see [S], [MSS], [M 93]).

This paper improves our earlier results from [M 92] (see [M 93] for an extended abstract), where we had exhibited neural nets of depth 4 with superlinear VC-dimension. Both our preceding results and the proof of our new result employ classical circuit construction methods due to Neciporuk [N] and Lupanov [L].

The *neural nets* which are considered in this paper are feedforward neural nets with linear threshold gates (or simpler: threshold gates), i.e. gates which apply the heaviside activation function to the weighted sum $\sum_{i=1}^{m} \alpha_i y_i + \alpha_0$ of their inputs $y_1, \ldots, y_m$. The parameters $\alpha_1, \ldots, \alpha_m$ and α_0 are the *weights* of such gate. We will consider in this paper only neural nets with boolean inputs and one boolean output.

The *depth* of the neural net is the length of the longest path from an input node to the output node (= output gate). The depth of a gate in a neural net is the

length of the longest path from an input node to that gate. We refer to all gates of depth d as "level d" of a neural net.

The *VC-dimension* VC-dimension $(\mathcal{N})$ of a neural net $\mathcal{N}$ with n inputs is defined as the size of the largest set $S \subseteq \{0,1\}^n$ which is "shattered" by $\mathcal{N}$ in the sense that every function $F : S \to \{0,1\}$ can be computed by $\mathcal{N}$ with some assignment of real numbers to its weights.

In the proof of the subsequent theorem we write $\mathrm{bin}(i)$ for the binary string that represents the natural number $i-1$. Furthermore we write $(x)_j$ for the j-th coordinate x_j of some vector $x = \langle x_1, \ldots, x_m \rangle \in \mathbf{R}^m$.

Theorem: One can construct for arbitrarily large $w \in \mathbf{N}$ neural nets $\mathcal{N}_w$ of depth 3 with $\leq 33w$ edges such that VC-dimension $(\mathcal{N}_w) \geq w \cdot \log w$.

Proof: One may view the weights of a neural net $\mathcal{N}$ collectively as the "program" of $\mathcal{N}$. It is obviously no problem to employ large weights in a neural net. But the question is how many bits of the weights are actual "program-bits" in the sense that they contribute to the VC-dimension of the neural net. We will exhibit in this proof a method for "efficient programming" of a neural net that allows us to encode an unusually large number of "program-bits" in the weights. More precisely we show that on average each of the w weights can be used to store $\Omega(\log w)$ "program-bits". Furthermore our construction requires only integer weights whose absolute value is polynomial in w. Hence all weights have bit-length $O(\log w)$, and our construction shows that a constant fraction of all weight-bits can be used to store the "program" of $\mathcal{N}$.

Before we describe the precise construction of the desired neural net $\mathcal{N}_w$, we first illustrate this method for "efficient programming" of a neural net in a simpler example. This "example" will turn out to be an essential building block for the construction of $\mathcal{N}_w$. We construct a neural net $\mathcal{M}$ with n integer weights of size polynomial in n that can be programmed to compute any $\sqrt{n}$-tuple of permutations of $\left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}$ (where $\underline{e}_i \in \{0,1\}^{\sqrt{n}}$ is the i-th unit vector, assume n is chosen such that $\sqrt{n} \in \mathbf{N}$). Since $(\sqrt{n}!)^{\sqrt{n}} = 2^{\Theta(n \log n)}$ different $\sqrt{n}$-tuples of permutations of $\left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}$ exist, this construction of $\mathcal{M}$ may be viewed as an example for "efficient programming" of a neural net.

The neural net $\mathcal{M}$ will have $2 \cdot \sqrt{n}$ inputs and $\sqrt{n}$ outputs. For the construction of $\mathcal{M}$ we assume that h is an arbitrary given function from $\left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}^2$ into $\left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}$ so that for any $j \in \{1, \ldots, \sqrt{n}\}$ the function $h\left(\cdot, \underline{e}_j\right)$ is a permutation of $\left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}$. Obviously any $\sqrt{n}$-tuple of permutations of $\left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}$ can be represented by such a function h. We define for $i, j \in \{1, \ldots, \sqrt{n}\}$ weights $w_{i,j}$ by $w_{i,j} = r \Leftrightarrow h\left(\underline{e}_r, \underline{e}_j\right) = \underline{e}_i$. This implies that for any $p, q \in \{1, \ldots, \sqrt{n}\}$ one has

$h\left(\underline{e}_p, \underline{e}_q\right) = \underline{e}_i \Leftrightarrow w_{i,q} = p \Leftrightarrow \sum_{j=1}^{\sqrt{n}} w_{i,j} \cdot \left(\underline{e}_q\right)_j = p$. Hence one may take as the i-th output gate of $\mathcal{M}$ a gate that checks (for example via the AND of two threshold gates) whether $\sum_{j=1}^{\sqrt{n}} w_{i,j} \cdot \left(\underline{e}_q\right)_j = \sum_{r=1}^{\sqrt{n}} r \cdot \left(\underline{e}_p\right)_r$ (i.e. $w_{i,q} = p$). It is easy to verify that this neural net $\mathcal{M}$ computes the given function h.

The above method for "efficient programming" of a neural net $\mathcal{M}$ cannot be used directly in order to show that a neural net has a large VC-dimension, because for that we have to be able to compute many $0-1$ valued rather than vector-valued functions F on a neural net. However the subsequent construction shows that any $0-1$ valued function F (with a suitable domain S) can be encoded by 4 functions $g_1, \ldots, g_4$ which are each $\sqrt{n}$-tuples of permutations of $\left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}$. Hence we can employ in the construction of $\mathcal{N}_w$ the preceding method for "efficient programming" of $\sqrt{n}$-tuples of permutations in a neural net. The only difference is that in $\mathcal{N}_w$ the output of each function g_k will be considered in *binary* representation (i.e. g_k outputs $\mathrm{bin}(i) \in \{0,1\}^{\log\sqrt{n}}$ instead of $\underline{e}_i \in \{0,1\}^{\sqrt{n}}$). This will be necessary since each bit of $\mathrm{bin}(i)$ will be used to encode the value of F for a different input.

We will now describe the precise construction of a neural net $\mathcal{N}_w$ with $O(w)$ edges and VC-dimension $(\mathcal{N}_w) \geq w \cdot \log w$.

For simplicity we assume that $n := 2w$ is of the form $2^{2n'}$ for some $n' \in \mathbf{N}$. This implies that $\sqrt{n}$ and $\log\sqrt{n}$ are natural numbers, and $\sqrt{n}$ is even. We construct a neural net $\mathcal{N}_w$ with $2\sqrt{n} + \log\sqrt{n}$ binary inputs and $O(n)$ weights that shatters the following set $S \subseteq \{0,1\}^{2\sqrt{n}+\log\sqrt{n}}$ of size $\frac{1}{2} n \log n$:

$$S := \left\{\underline{e}_p \underline{e}_q \tilde{\underline{e}}_m : p, q \in \left\{1, \ldots, \sqrt{n}\right\} \text{ and } m \in \left\{1, \ldots, \log\sqrt{n}\right\}\right\},$$

where $\underline{e}_r \in \{0,1\}^{\sqrt{n}}$ denotes the r-th unit vector of length $\sqrt{n}$ $(r = 1, \ldots, \sqrt{n})$, and $\tilde{\underline{e}}_m \in \{0,1\}^{\log\sqrt{n}}$ denotes the m-th unit vector of length $\log\sqrt{n}$ $(m = 1, \ldots, \log\sqrt{n})$. Thus each $u \in S$ contains exactly three 1's.

Fix any map $F : S \to \{0,1\}$. One can encode F by a function $g : \left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}^2 \to \{0,1\}^{\log\sqrt{n}}$ where the m-th output bit of $g\left(\underline{e}_p, \underline{e}_q\right)$ equals 1 if and only if $F\left(\underline{e}_p \underline{e}_q \tilde{\underline{e}}_m\right) = 1$. It is straightforward to show that for any function $g : \left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}^2 \to \{0,1\}^{\log\sqrt{n}}$ there exist for $k = 1, \ldots, 4$ functions $g_k : \left\{\underline{e}_1, \ldots, \underline{e}_{\sqrt{n}}\right\}^2 \to \{0,1\}^{\log\sqrt{n}}$ such that $g_k\left(\cdot, \underline{e}_q\right)$ is $1-1$ for every fixed $q \in \{1, \ldots, \sqrt{n}\}$, and such that for all $p, q \in \{1, \ldots, \sqrt{n}\}$:

$$g\left(\underline{e}_p, \underline{e}_q\right) = \begin{cases} g_1\left(\underline{e}_p, \underline{e}_q\right) \oplus g_2\left(\underline{e}_p, \underline{e}_q\right), \text{ if } p \leq \frac{\sqrt{n}}{2} \\ g_3\left(\underline{e}_p, \underline{e}_q\right) \oplus g_4\left(\underline{e}_p, \underline{e}_q\right), \text{ if } p > \frac{\sqrt{n}}{2}. \end{cases}$$

The symbol $\oplus$ denotes here the bit-wise exclusive OR (i.e. parity) on bit-strings of length $\log\sqrt{n}$. For each fixed $\underline{e}_q$ one chooses simultaneously for $k = 1, 2$ values for

$g_k\left(\underline{e}_1, \underline{e}_q\right)$, $g_k\left(\underline{e}_2, \underline{e}_q\right)$, ... in such a way that earlier assigned values for $g_k\left(\cdot, \underline{e}_q\right)$ are avoided. After $g_1(\underline{e}_p, \underline{e}_q)$ and $g_2(\underline{e}_p, \underline{e}_q)$ have been defined for $p = 1, \ldots, l < \frac{\sqrt{n}}{2}$, one can choose for $g_1(\underline{e}_{l+1}, \underline{e}_q)$ any string in $\{0,1\}^{\log \sqrt{n}}$ that is not in the set

$$\{g_1(\underline{e}_p, \underline{e}_q) : p = 1, \ldots, l\} \cup \{g(\underline{e}_{l+1}, \underline{e}_q) \oplus g_2(\underline{e}_p, \underline{e}_q) : p = 1, \ldots, l\}.$$

Then one sets $g_2(\underline{e}_{l+1}, \underline{e}_q) := g(\underline{e}_{l+1}, \underline{e}_q) \oplus g_1(\underline{e}_{l+1}, \underline{e}_q)$. One continues the definition of $g_1(\underline{e}_p, \underline{e}_q)$ and $g_2(\underline{e}_p, \underline{e}_q)$ for $p > \frac{\sqrt{n}}{2}$ in an arbitrary manner so that $g_1(\cdot, \underline{e}_q)$ and $g_2(\cdot, \underline{e}_q)$ become $1-1$. The definition of $g_3(\cdot, \underline{e}_q)$ and $g_4(\cdot, \underline{e}_q)$ is analogous.

The neural net $\mathcal{N}_w$ computes F in the following way. The output gate on level 3 is an OR of $4 \log \sqrt{n}$ threshold gates. These threshold gates consist of $\log \sqrt{n}$ blocks of 4 threshold gates, such that for any $b \in \{1, \ldots, \log \sqrt{n}\}$ some threshold gate in the b-th block outputs 1 for network input $\underline{e}_p \underline{e}_q \tilde{\underline{e}}_m$ if and only if $m = b$ and $\left(g\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 1$ (i.e. $F\left(\underline{e}_p \underline{e}_q \tilde{\underline{e}}_m\right) = 1$). More precisely, the a-th threshold gate in block b outputs 1 if and only if the a-th one of the following 4 conditions is satisfied:

$$(1) \quad m = b \quad \wedge \quad p \leq \tfrac{\sqrt{n}}{2} \quad \wedge \quad \left(g_1\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 1 \quad \wedge \quad \left(g_2\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 0$$

$$(2) \quad m = b \quad \wedge \quad p \leq \tfrac{\sqrt{n}}{2} \quad \wedge \quad \left(g_1\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 0 \quad \wedge \quad \left(g_2\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 1$$

$$(3) \quad m = b \quad \wedge \quad p > \tfrac{\sqrt{n}}{2} \quad \wedge \quad \left(g_3\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 1 \quad \wedge \quad \left(g_4\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 0$$

$$(4) \quad m = b \quad \wedge \quad p > \tfrac{\sqrt{n}}{2} \quad \wedge \quad \left(g_3\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 0 \quad \wedge \quad \left(g_4\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 1.$$

The subcondition "$m = b$" is satisfied if and only if $\left(\underline{e}_p \underline{e}_q \tilde{\underline{e}}_m\right)_{2\sqrt{n}+b} = 1$. The subcondition "$p \leq \frac{\sqrt{n}}{2}$" is satisfied if and only if $\sum_{r=1}^{\sqrt{n}} r \cdot \left(\underline{e}_p\right)_r \leq \frac{\sqrt{n}}{2}$, hence it can be tested by a threshold gate on level 1 (analogously for "$p > \frac{\sqrt{n}}{2}$"). The remaining subconditions are tested with the help of $8\sqrt{n}$ threshold gates on level 1 that involve weights $w_{k,i,j} \in \{1, \ldots, \sqrt{n}\}$, which are defined by the condition

$$w_{k,i,j} = r \Leftrightarrow g_k\left(\underline{e}_r, \underline{e}_j\right) = \mathrm{bin}(i).$$

Among these there are $4\sqrt{n}$ threshold gates $G^+_{k,i}\left(\underline{e}_p, \underline{e}_q\right)$ on level 1 that output 1 if and only if $\sum_{r=1}^{\sqrt{n}} r \cdot \left(\underline{e}_p\right)_r \geq \sum_{j=1}^{\sqrt{n}} w_{k,i,j} \cdot \left(\underline{e}_q\right)_j$ (i.e. $p \geq w_{k,i,q}$), and $4\sqrt{n}$ threshold gates $G^-_{k,i}\left(\underline{e}_p, \underline{e}_q\right)$ on level 1 that output 1 if and only if $\sum_{r=1}^{\sqrt{n}} r \cdot \left(\underline{e}_p\right)_r \leq \sum_{j=1}^{\sqrt{n}} w_{k,i,j} \cdot \left(\underline{e}_q\right)_j$ (i.e. $p \leq w_{k,i,q}$); for $k = 1, \ldots, 4$ and $i = 1, \ldots, \sqrt{n}$. These are the only weights in the neural net $\mathcal{N}_w$ which depend on the function $F : S \to \{0,1\}$. By definition one has that for each k, i at least one of the two gates $G^+_{k,i}\left(\underline{e}_p, \underline{e}_q\right)$, $G^-_{k,i}\left(\underline{e}_p, \underline{e}_q\right)$ outputs 1. Furthermore for any $k \in \{1, \ldots, 4\}$ and any $\underline{e}_p \underline{e}_q \tilde{\underline{e}}_m \in S$ there is exactly one

$i \in \{1, \ldots, \sqrt{n}\}$ such that both of these gates output 1. This index i is characterized by the equality $g_k\left(\underline{e}_p, \underline{e}_q\right) = \mathrm{bin}(i)$. Hence one can check whether $\left(g_k\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 1$ by testing whether

$$\sum_{\substack{i=1,\ldots,\sqrt{n} \\ \text{with } (\mathrm{bin}(i))_b=1}} G^+_{k,i}\left(\underline{e}_p, \underline{e}_q\right) + G^-_{k,i}\left(\underline{e}_p, \underline{e}_q\right) \geq \frac{\sqrt{n}}{2} + 1,$$

and one can check whether $\left(g_k\left(\underline{e}_p, \underline{e}_q\right)\right)_b = 0$ by testing whether

$$\sum_{\substack{i=1,\ldots,\sqrt{n} \\ \text{with } (\mathrm{bin}(i))_b=0}} G^+_{k,i}\left(\underline{e}_p, \underline{e}_q\right) + G^-_{k,i}\left(\underline{e}_p, \underline{e}_q\right) \geq \frac{\sqrt{n}}{2} + 1.$$

Furthermore the sums on the left hand side of both inequalities can only assume the values $\frac{\sqrt{n}}{2}$ or $\frac{\sqrt{n}}{2} + 1$. Therefore one can test the AND of two subconditions of this type and of the subconditions "$m = b$" and "$p \leq \frac{\sqrt{n}}{2}$" ("$p > \frac{\sqrt{n}}{2}$") by a *single* threshold gate on level 2 of $\mathcal{N}_w$. Hence one can test each of the conditions $(1), \ldots, (4)$ by a separate threshold gate in the b-th block on level 2.

Altogether the network $\mathcal{N}_w$ has $8\sqrt{n} + 2$ threshold gates on level 1 (the first hidden layer), which are connected by $16n + 2\sqrt{n}$ edges to level 0 (the input level). Among these are the $8n$ edges with the weights $w_{k,i,j}$ that depend on the function $F : S \to \{0, 1\}$ (all other weights and thresholds in $\mathcal{N}_w$ are independent of F).

On level 2 (the second hidden layer) the network $\mathcal{N}_w$ has $4 \log \sqrt{n}$ threshold gates that test the conditions $(1), \ldots, (4)$ as described before. Each of these gates is connected by $1 + 2\sqrt{n}$ edges to gates on level 1, and by one edge to some input node. On level 3 the network $\mathcal{N}_w$ has an OR that is connected by $4 \log \sqrt{n}$ edges to the gates on level 2.

Thus the constructed network $\mathcal{N}_w$ consists of $2\sqrt{n} + \log \sqrt{n}$ input nodes, $8\sqrt{n} + 2 \log n + 3$ computation nodes, and $16n + (4 \log n + 2)\sqrt{n} + 6 \log n$ edges. Obviously the nodes and edges of $\mathcal{N}_w$ are independent of the given function $F : S \to \{0, 1\}$, and only the weights on $8n$ of the edges depend on F (these weights range over $\{1, \ldots, \sqrt{n}\}$). Since the function $F : S \to \{0, 1\}$ was choosen arbitrarily, the construction implies that the set S is shattered by $\mathcal{N}_w$. Hence VC-dimension$(\mathcal{N}_w) \geq |S| = \frac{1}{2} n \log n \geq w \log w$ (since $n := 2w$). ∎

Acknowledgement

We would like to thank György Turán for inspiring discussions.

References

[BH] E. B. Baum, D. Haussler, "What size net gives valid generalization?", *Neural Computation*, vol. 1, 1989, 151 - 160

[BEHW] A. Blumer, A. Ehrenfeucht, D. Haussler, M. K. Warmuth, "Learnability and the Vapnik-Chervonenkis dimension", *J. of the ACM*, vol. 36(4), 1989, 929 - 965

[C 64] T. M. Cover, "Geometrical and statistical properties of linear threshold devices", Stanford PH. D. Thesis 1964, *Stanford SEL Technical Report No. 6107-1*, May 1964

[C 88] T. M. Cover, "Capacity problems for linear machines", in: *Pattern Recognition*, L. Kanal ed., *Thompson Book Co.*, 1988, 283 - 289

[EHKV] A. Ehrenfeucht, D. Haussler, M. Kearns, L. Valiant, "A general lower bound on the number of examples needed for learning", *Information and Computation*, vol. 82, 1989, 247 - 261

[L] O. B. Lupanov, "On circuits of threshold elements", *Dokl. Akad. Nauk* SSSR, vol. 202, 1288 - 1291; engl. translation in: *Sov. Phys. Dokl.*, vol. 17, 1972, 91 - 93

[MSS] W. Maass, G. Schnitger, E. D. Sontag, "On the computational power of sigmoid versus boolean threshold circuits", *Proc. of the 32nd Annual IEEE Symp. on Foundations of Computer Science*, 1991, 767 - 776

[M 92] W. Maass, "Bounds for the computational power and learning complexity of analog neural nets", *IIG-Report 349 of the Technische Universität Graz*, (October 1992).

[M 93] W. Maass, "Bounds for the computational power and learning complexity of analog neural nets (Extended Abstract)", *Proc. of the 25th Annual ACM Symposium on the Theory of Computing*, 1993, 335 - 344

[N] E. I. Neciporuk, "The synthesis of networks from threshold elements", *Probl. Kibern.* No. 11, 1964, 49 - 62; engl. translation in: *Autom. Expr.*, vol. 7, No. 1, 1964, 35 - 39

[RM] D. E. Rumelhart, J. L. McClelland, "Parallel Distributed Processing", vol. 1, *MIT Press* (Cambridge, 1986)

[S] E. D. Sontag, "Feedforward nets for interpolation and classification", *J. Comp. Syst. Sci.*, vol. 45, 1992, 20 - 48

[V] L. G. Valiant, "A theory of the learnable", *Comm. of the ACM*, vol. 27, 1984, 1134 - 1142