Boolean Functions and Artificial Neural Networks*

Martin Anthony
Department of Mathematics
and Centre for Discrete and Applicable Mathematics
The London School of Economics and Political Science
London WC2A 2AE, UK
m.anthony@lse.ac.uk

CDAM Research Report LSE-CDAM-2003-01
January 2003

Abstract

This report surveys some connections between Boolean functions and artificial neural networks. The focus is on cases in which the individual neurons are linear threshold neurons, sigmoid neurons, polynomial threshold neurons, or spiking neurons. We explore the relationships between types of artificial neural network and classes of Boolean function. In particular, we investigate the type of Boolean functions a given type of network can compute, and how extensive or expressive the set of functions so computable is. A version of this is to appear as a chapter in a book on Boolean functions, but the report itself is relatively self-contained.

*A version of this is to appear as a chapter in *Boolean Functions: Volume II*, edited by Yves Crama and Peter Hammer

1 Introduction

There has recently been much interest in 'artificial neural networks', machines (or models of computation) based loosely on the ways in which the brain is believed to work. Neurobiologists are interested in using these machines as a means of modeling biological brains, but much of the impetus comes from their applications. For example, engineers wish to create machines that can perform 'cognitive' tasks, such as speech recognition, and economists are interested in financial time series prediction using such machines.

In this report we shall focus on individual 'artificial neurons' and feed-forward artificial neural networks. We shall be particularly interested in cases where the neurons are linear threshold neurons, sigmoid neurons, polynomial threshold neurons, and spiking neurons. We will investigate the relationships between types of artificial neural network and classes of Boolean function. In particular, we shall ask questions about the type of Boolean functions a given type of network can compute, and about how extensive or expressive the set of functions so computable is.

2 Artificial neural networks

2.1 Introduction

It appears that one reason why the human brain is so powerful is the sheer complexity of connections between neurons. In computer science parlance, the brain exhibits huge parallelism, with each neuron connected to many other neurons. This has been reflected in the design of artificial neural networks. One advantage of such parallelism is that the resulting network is *robust*: in a serial computer, a single fault can make computation impossible, whereas in a system with a high degree of parallelism and many computation paths, a small number of faults may be tolerated with little or no upset to the computation. There are many good general texts on neural networks, such as [7, 16]. Here we shall briefly describe the aspects of neural networks that we will be interested in from a Boolean functions point of view.

Generally speaking, we can say that an artificial neural network consists of a directed graph with *computation units* (or *neurons*) situated at the vertices. One or more of these computation

units are specified as *output units*. These are the units with zero out-degree in the directed graph. We shall consider networks in which there is only one output unit. Additionally, the network has *input units*, which receive signals from the outside world. Each unit produces an output, which is transmitted to other units along the arcs of the directed graph. The outputs of the input units are simply the input signals that have been applied to them. The computation units have *activation functions* determining their outputs. The degree to which the output of one computation unit influences those of its neighbors is determined by the weights assigned to the network. This description is quite abstract at this stage, but we shall concretize it shortly by focusing on particular types of network.

2.2 Neurons

The building blocks of feed-forward networks are *computation units* (or *neurons*). In isolation, a computation unit has some number, k, of *inputs*, and is capable of taking on a number of *states*, each described by a vector $w = (w_0, w_1, \ldots, w_p) \in \mathbb{R}^p$ of p real numbers, known as *weights* or *parameters*. Here, p, the number of parameters of the unit, will depend on k. If the unit is a *linear threshold* unit or *sigmoid unit*, then $p = k + 1$ and, in these cases, it is useful to think of the weights $w_1, w_2, \ldots, w_k$ as being assigned to each of the k inputs. For *spiking neurons* and *polynomial threshold units*, the number of parameters will be greater than $k + 1$. The different types of neurons we consider are best described by defining how they process their inputs.

Generally, when in the state described by $w \in \mathbb{R}^p$, and on receiving input $x = (x_1, x_2, \ldots, x_k)$, the computation unit produces as output an *activation* $g(w, x)$, where $g : \mathbb{R}^p \times \mathbb{R}^k \to \mathbb{R}$ is a fixed function. We may regard the unit as a parameterized function class. That is, we may write $g(w, x) = g_w(x)$, where, for each state w, $g_w : \mathbb{R}^k \to \mathbb{R}$ is the function computed by the unit on the inputs x.

Linear threshold units

For a linear threshold unit, the function g takes a particularly simple form:

$$g(w, x) = \operatorname{sgn}(w_0 + w_1 x_1 + \cdots + w_k x_k),$$

where sgn is the sign function, given by

$$\mathrm{sgn}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0, \end{cases}$$

Thus, when the state of the unit is given by $w = (w_0, w_1, \ldots, w_k)$, the output is either 1 or 0, and it is 1 precisely when

$$w_0 + w_1 x_1 + \cdots + w_k x_k \geq 0,$$

which may be written as

$$w_1 x_1 + \cdots + w_k x_k \geq \theta,$$

where $\theta = -w_0$ is known as the *threshold*. In other words, the computation unit gives output 1 (in biological parlance, it *fires*) if and only if the weighted sum of its inputs is at least the threshold θ. If the inputs to the threshold unit are restricted to $\{0,1\}^n$, then the set of Boolean functions it computes is precisely the *(Boolean) threshold functions*.

Sigmoid units

For a (standard) sigmoid unit, we have

$$g(w, x) = \sigma\left(w_0 + w_1 x_1 + \cdots + w_k x_k\right),$$

where the 'activation function' $\sigma(z) = 1/(1 + e^{-z})$ is the *standard sigmoid function*. Writing $\theta = -w_0$, as we did above for the linear threshold unit, we see that the output of the sigmoid unit is $\sigma\left(\sum_{i=1}^k w_i x_i - \theta\right)$. If the weighted sum $\sum_{i=1}^k w_i x_k$ is much larger than the threshold, then the output is close to 1; if it is much less than the threshold, the output is close to 0; and if it is very close to the threshold, then the output is close to $1/2$. In fact, the sigmoid function can be thought of as a 'smoothed' version of the sign function, sgn, since σ maps from $\mathbb{R}$ into the interval $(0,1)$, is differentiable, and satisfies

$$\lim_{z \to -\infty} \sigma(z) = 0, \quad \lim_{z \to \infty} \sigma(z) = 1.$$

Note that, whereas the linear threshold unit has output in $\{0,1\}$, the output of a sigmoid unit lies in the interval $(0,1)$ of real numbers.

Polynomial threshold units

The linear threshold and sigmoid units both work with $w_1x_1 + \cdots + w_kx_k$, a linear combination of the inputs to the unit, but we can generalize from this and consider instead units which use a non-linear combination of the x_i. For example, when $k = 3$, imagine a unit which computes the quadratic expression

$$w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1^2 + w_5x_2^2 + w_6x_3^2 + w_7x_1x_2 + w_8x_1x_3 + w_9x_2x_3,$$

for some contants $w_i, (1 \leq i \leq 9)$, and then compares this with a threshold value θ. Such a unit is a *polynomial threshold unit* of degree 2. We now set up a description of this generalization of linear threshold units. We shall denote by $[n]^m$ the set of all selections, in which repetition is allowed, of at most m objects from the set $[n] = \{1, 2, \ldots, n\}$. Thus, $[n]^m$ is a collection of 'multi-sets'. For example, $[3]^2$ consists of the multi-sets

$$\emptyset, \{1\}, \{1, 1\}, \{2\}, \{2, 2\}, \{3\}, \{3, 3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}.$$

A *polynomial threshold unit of degree* m (also termed a *sigma-pi unit* [37, 44, 48]) has $p = \binom{n+m}{m}$ parameters w_S, one for each multi-set $S \in [n]^m$. For $S \in [n]^m$ and $x = x_1x_2 \ldots, x_n \in \mathbb{R}^n$, let x_S denote the product of the x_i for $i \in S$ (with repetitions as required). For example, $x_{\{1,2,3\}} = x_1x_2x_3$ and $x_{\{1,1,2\}} = x_1^2x_2$. When $S = \emptyset$, the empty set, we interpret x_S as the constant 1. The output of the unit is given by

$$g_w(x) = g(w, x) = \operatorname{sgn}\left(\sum_{S \in [n]^m} w_S x_S\right).$$

Of course, when $m = 1$ we obtain a linear threshold unit. But for $m > 1$, a polynomial threshold unit can compute functions that a linear threshold unit is incapable of computing. Furthermore (and this will prove useful later), note that if we restrict the inputs x_i to belong to $\{0, 1\}$ then we do not need terms of the form $w_S x_S$ where the multi-set S contains repeated members: this is simply because if $x_i \in \{0, 1\}$ then $x_i^r = x_i$ for all $r > 1$.

Consider, for example, the case $n = m = 2$ and suppose we take

$$w_\emptyset = -\frac{1}{2},\ w_{\{1\}} = w_{\{2\}} = 1,\ w_{\{1,2\}} = -2,$$

with the remaining weights $w_{\{1,1\}}$ and $w_{\{2,2\}}$ equal to 0. Then

$$g_w(x) = \operatorname{sgn}\left(-\frac{1}{2} + x_1 + x_2 - 2x_1x_2\right).$$

It is easy to verify that, as a Boolean function on $\{0,1\}^2$, g is the exclusive-or function, which is not computable by a linear threshold unit.

Spiking neurons

A very interesting class of artificial neurons are the *spiking neurons*. A number of results on the capabilities of these neurons and networks of them have been obtained by Maass and Schmitt [28, 25, 26, 41]. In this report we present some results from [41, 28] concerning spiking neurons of a simplified type. The type of neuron considered is a 'Type A' spiking neuron with 'binary encoding' [28]. For biological motivation for this model, see [28] and the references cited there. The key difference between this type of neuron and the ones considered so far is the introduction of a time variable. In the three types of neuron discussed so far, a weighted sum is immediately computed and the output of the neuron depends directly on that weighted sum. Here, however, *delays* in the inputs to the neuron are modeled by assuming not only that to each input there is associated a weight w_i, but also a *delay* d_i. It is assumed that the weighted input corresponding to input unit i is only 'active' during the time interval $[d_i, d_i + 1)$. If, at any time, the sum of the currently active weighted inputs is at least the threshold value, then the neuron fires; otherwise it does not. Formally, with k inputs and in state

$$w = (w_0, w_1, w_2, \ldots, w_k, d_1, d_2, \ldots, d_k),$$

the output of the spiking neuron is given by

$$g(w, x) = \operatorname{sgn}\left(w_0 + \max_{t \geq 0} \sum_{i=1}^{k} w_i x_i \chi_{[d_i, d_i+1)}(t)\right),$$

where $\chi_{[d_i,d_i+1)}$, the characteristic function of the time interval $[d_i, d_i + 1)$, is given by

$$\chi_{[d_i,d_i+1)}(t) = \begin{cases} 1 & \text{if } d_i \leq t < d_i + 1 \\ 0 & \text{otherwise,} \end{cases}$$

Observe that if all delays d_i are fixed at 0, then the spiking neuron behaves just like the linear threshold neuron with weights $(w_0, w_1, \ldots, w_k)$.

2.3 Networks

As mentioned in the general description above, a neural network is formed when we place units at the vertices of a directed graph, with the arcs of the digraph representing the flows of signals between units. Some of the units are termed *input units*: these receive signals not from other units, but instead they take their signals from the outside environment. Units that do not transmit signals to other units are termed *output units*. The network is said to be a *feed-forward network* if the underlying directed graph is acyclic (that is, it has no directed cycles). This feed-forward condition means that the units can be labeled with integers in such a way that if there is a connection from the computation unit labeled i to the computation unit labeled j then $i < j$. We will often be interested in *multi-layer* networks. In such networks, the units may be grouped into *layers*, labeled $0, 1, 2, \ldots, \ell$, in such a way that the input units form layer 0, these feed into the computation units, and if there is a connection from a computation unit in layer r to a computation unit in layer s, then we must have $s > r$. Note, in particular, that there are no connections between any two units in a given layer. We call such a network an ℓ-layer network. (Strictly speaking, it has $\ell + 1$ layers, but one of these consists entirely of input units, and it is the number of layers of computation units that is usually important.) Any feed-forward network is a multi-layer network (since we could just take the layers to consist of single computation units), but we shall often be interested in feed-forward networks with a small number of layers. It is easy to see that the smallest ℓ for which such a layering is possible is the *depth* of the network, defined as the length of the largest directed path in the underlying directed graph.

We shall primarily be interested in single polynomial threshold units and spiking neurons, and in one-output feed-forward networks in which the computation units are linear threshold units or sigmoid units. A threshold or sigmoid network with n input units is capable of computing a number of functions from $\mathbb{R}^n$ to $\mathbb{R}$, or (simply restricting the input signals to be $\{0, 1\}$-valued) from $\{0, 1\}^n \to \mathbb{R}$. The precise function computed depends on the state of each computation unit. Recall that for the threshold and sigmoid neurons, if a unit has k inputs then the state is a vector of $k + 1$ real numbers: one of these numbers (w_0 or its negative, the threshold θ in the description above) can be thought of as being attached to the unit itself, and the other k can be thought of as describing the weight attached to each of the k arcs feeding into the unit. Suppose that the network has N computation units, labeled $1, 2, \ldots, N$, and that computation unit i has k_i inputs. Then the total number of weights in the network is

$$\sum_{i=1}^{N} (k_i + 1) = N + \sum_{i=1}^{N} k_i = N + E,$$

where E denotes the total number of arcs in the digraph. We may therefore say that the *state of the network* as a whole is described by a vector w of $W = N + E$ real numbers. When there are n input units and one output unit, the network computes, for each state w, a function $h_w : \mathbb{R}^n \to \mathbb{R}$. The set of functions *computable* by the network when the weight vector can be chosen from a subset Ω of $\mathbb{R}^W$ is $\{h_w : w \in \Omega\}$. (Often, Ω will simply be $\mathbb{R}^W$, but one may want, for example, to restrict the sizes of the allowable weights, in which case Ω will be a strict subset of $\mathbb{R}^W$.)

Linear threshold networks have long been studied, and were the subject of much work in 'threshold logic' in the 1960's; see the books by Muroga [32] and Hu [17], and the papers cited there. A single linear threshold unit may be regarded as a linear threshold network, and this simplest of all neural networks is often called the *perceptron*, though that term is also used more generally [30]. Questions concerning the type of function computable by a polynomial threshold unit have been worked on by a number of researchers, and were considered in [30, 9, 34]. For more recent results, see the survey article by Saks [38]: this provides an excellent overview of much of the theoretical work on functions computable by threshold and polynomial threshold units and related areas (some of which will be touched on later in this report). See also [47].

In the rest of this report, we concentrate on two main issues. First, how many and what type of Boolean functions can be computed by neural networks of particular types? Secondly, what is the expressive power (as measured by the VC-dimension, an important parameter in quantifying the complexity of learning [46, 2]).

3 Computing Boolean functions by neural networks

3.1 Linear threshold units

We have noted that the Boolean functions computed by the single linear threshold unit are precisely the Boolean threshold functions. Recall that f is a (Boolean) threshold defined on $\{0,1\}^n$ if there are $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ such that

$$f(x) = \begin{cases} 1 & \text{if } \langle w, x \rangle \geq \theta \\ 0 & \text{if } \langle w, x \rangle < \theta, \end{cases}$$

where $\langle w, x \rangle = w^T x$ is the standard inner product of w and x. Given such w and θ, we say that f is represented by $[w, \theta]$ and we write $f \leftarrow [w, \theta]$. The vector w is known as the *weight-vector*, and θ is known as the *threshold*. We denote the class of threshold functions on $\{0,1\}^n$ by T_n. Note that any $f \in T_n$ will satisfy $f \leftarrow [w, \theta]$ for ranges of w and θ.

Asummability and linear separability

Properties and characterizations of (Boolean) threshold functions have been much-explored, and we discuss only a few aspects here. Geometrically, a Boolean function f is a threshold function if the true and false points are separable by a hyperplane; that is, f is *linearly separable*. Such functions can also be characterized by the asummability property, as follows.

Theorem 3.1 *The Boolean function f is a threshold function if and only if it is* asummable*, meaning that for any $k \in \mathbb{N}$, for any sequence $x_1, x_2, \ldots, x_k$ of (not necessarily distinct) true points of f and any sequence $y_1, y_2, \ldots, y_k$ of (not necessarily distinct) false points of f,*

$$\sum_{i=1}^{k} x_i \neq \sum_{i=1}^{k} y_i.$$

Asummability can be seen to be equivalent to the non-intersection of the convex hulls of the sets true points and false points of f. (It can be seen quite directly to be equivalent to the assertion that there is no point that is simultaneously a rational convex combination of true points and a rational convex combination of false points. This, in turn, is equivalent to the non-intersection of the convex hulls.) By the Separating Hyperplanes Theorem, asummability is therefore equivalent to linear separability.

Number of functions computed

A classical result, which dates back to work by Schläfli in the last century [40] and which also appears in [9], is that the maximum number of connected regions into which $\mathbb{R}^d$ can be

partitioned by N hyperplanes passing through the origin is bounded above by

$$C(N,d) = 2\sum_{k=0}^{d-1} \binom{N-1}{k}.$$

(Here, we apply the usual convention that $\binom{a}{b} = 0$ if $b > a$, and $\binom{0}{b} = 1$.) From this, it is possible to obtain the following result [9].

Theorem 3.2 *Suppose that $S \subseteq \mathbb{R}^n$ is finite. Then the number of different functions $f : S \to \{0,1\}$ computable by a linear threshold unit on domain S is at most*

$$2\sum_{k=0}^{n} \binom{|S|-1}{k}.$$

Taking $S = \{0,1\}^n$, the set of computable functions is just T_n, the set of Boolean threshold functions, so we obtain

$$|T_n| \leq 2\sum_{k=0}^{n} \binom{2^n-1}{k} \leq 2^{n^2}.$$

It is clear that T_n is a vanishingly small fraction of all Boolean functions on $\{0,1\}^n$, as might be expected. Since the 1960's (see Muroga's book [32]), a lower bound on $|T_n|$ of the form $2^{(n^2/2)(1+o(1))}$ has been known. More recently, Zuev [49] showed that, for sufficiently large n, $\log_2 |T_n| > n^2\left(1 - 10/\ln n\right)$. So the upper bound is asymptotically of the right order.

Sizes of weights

A weight-vector and threshold are said to be *integral* if the threshold and each entry of the weight-vector are integers. Any Boolean threshold function can be represented by an integral weight-vector and threshold. To see this, note first that, by the discreteness of $\{0,1\}^n$, any Boolean threshold function can be represented by a rational threshold and weight-vector. Scaling these by a suitably large integer yields integral threshold and weight-vector representing the same function. A natural question is how large the integer weights (including the threshold) have to be. An upper bound is as follows [32].

Theorem 3.3 *For any Boolean threshold function f on $\{0,1\}^n$, there is an integral weight-vector w and an integral threshold θ such that $t \leftarrow [w, \theta]$ and such that*

$$\max\{|\theta|, |w_1|, \ldots, |w_n|\} \leq (n+1)n^{n/2}.$$

It is easy to show that exponential-sized integer weights are sometimes necessary just by a simple counting argument. A result of Muroga [32] alluded to above says that there are at least $2^{n(n-1)/2}$ threshold functions on $\{0,1\}^n$. For $B \in \mathbb{N}$, the number of pairs (w, θ) of integer weight-vector and threshold which satisfy $|w_i| \leq B$ for $i = 1, 2, \ldots, n$, and $|\theta| \leq B$, is at most B^{n+1}. So, for example, the number of threshold functions representable with integer weights and threshold bounded in magnitude by $2^{n/6}$ is no more than $2^{n(n+1)/6}$. But this is less than $2^{n(n-1)/2}$ for $n \geq 2$, so there must be some threshold functions in which, using integer weights, we would need weights greater than $2^{n/6}$ in magnitude. This simple argument given above establishes the need for large weights, but it does not provide a concrete example of a threshold function requiring such large weights. Specific examples of such functions have long been known (see [32, 31]). We now present an example function which, although it is not the simplest possible, will be useful later and has been of much interest in analysing the performance of the perceptron learning algorithm [14, 5].

Consider, for n even, the Boolean function f_n on variables with formula

$$f_n = u_n \wedge (u_{n-1} \vee (u_{n-2} \wedge (u_{n-3} \vee (\ldots (u_2 \wedge u_1)) \ldots)).$$

Thus, for example,

$$f_6 = u_6 \wedge (u_5 \vee (u_4 \wedge (u_3 \vee (u_2 \wedge u_1)))).$$

It can be shown (see [35, 1], for example) that if w is any integral weight-vector in a threshold representation of f_n, then $w_i \geq F_i$ for $1 \leq i \leq n$, where F_i is the ith Fibonacci number. Since

$$F_n \geq \frac{\sqrt{5}}{6}\left(\frac{1+\sqrt{5}}{2}\right)^n,$$

for all n this function requires integer weights exponential in n.

The general upper bound on integral weights given in Theorem 3.3 is $(n+1)n^{n/2}$, whereas the specific lower bound exhibited by the function f_n is (merely) exponential in n. The question arises as to whether the general upper bound is loose and could potentially be considerably improved. In fact, however, the upper bound is quite tight. Specifically, Håstad [15] has proved

that there are constants $k > 0$ and $c > 1$ such that, for n a power of 2, there is a threshold function f on $\{0,1\}^n$ such that any integral weight-vector representing f has a weight at least $kc^{-n}n^{n/2}$.

Test sets for linear threshold functions

For $f \in T_n$, we say that a set $S \subseteq \{0,1\}^n$ is a *test set* for f if when $h \in T_n$ and h classifies the inputs in S in the same way as f does, then h is necessarily equal to f, among all threshold functions. In other words, S is a test set for f if the inputs in S serve to specify uniquely the function f. Denote by $\sigma(f)$ the cardinality of the smallest test set for t. This parameter is useful in considering the complexity of 'teaching' linear threshold functions; see [11, 4]. The following result was obtained in [4].

Theorem 3.4 *Suppose $f \in T_n$ and suppose that $k \geq 1$ is such that any weight-vector realizing f has at least k non-zero weights and that there is a weight-vector realizing f which has exactly k non-zero weights. Then*

$$2^{n-k}(k+1) \leq \sigma(f) \leq 2^{n-k}\binom{k+1}{\lfloor \frac{k+1}{2} \rfloor},$$

and equality is possible in both of these inequalities.

Despite the fact that the testing number can be exponential, it can be shown [4] that the average, or expected, testing number of a function in T_n is at most n^2.

Fixing attention for the moment on the case $k = n$ above, it has been shown [4] that there is a large family of threshold functions — the *nested* functions — each having minimum possible testing number. Let us recursively define a Boolean function to be *canonically nested* by: both functions of 1 variable are canonically nested, and t_n, a function of n variables, is canonically nested if $t_n = u_n \star t_{n-1}$ or $t_n = \bar{u}_n \star t_{n-1}$ where $\star$ is $\vee$ (the OR connective) or $\wedge$ (the AND connective) and t_{n-1} is a canonically nested function of $n-1$ variables. (Here, we mean that t_{n-1} acts on the first $n-1$ entries of its argument.) We say that a function f is *nested* if, by permuting (or re-labeling) the variables, we obtain a canonically nested function. One may relate nested functions to particular types of *decision lists* (as defined by [36]). It is straightforward to

see that any nested function can be realized as a 1-decision list of length n in which, for each i between 1 and n, precisely one term of the form (u_i, b) or $(\bar{u}_i, b)$ occurs (for some $b \in \{0,1\}$) (and *vice versa*). It is easily seen that any nested function is a threshold function. Examples of nested functions include the functions f_n described above. It turns out [4] that all nested functions (regarded as threshold functions) have the smallest possible testing numbers, since each has testing number $n+1$.

3.2 Polynomial threshold units

We now consider the Boolean functions computable by a polynomial threshold unit. For $n \in \mathbb{N}$ and $d \leq n$, let $[n]^{(d)}$ denote all subsets of $[n] = \{1, 2, \ldots, n\}$ of cardinality at most d. A Boolean function f defined on $\{0,1\}^n$ is a *polynomial threshold function* of *degree* d if there are real numbers w_S, one for each $S \in [n]^{(d)}$, such that

$$f(x) = \operatorname{sgn}\left(\sum_{S \in [n]^{(d)}} w_S x_S\right),$$

where the notation is as defined earlier. The set of polynomial threshold functions on $\{0,1\}^n$ of degree d will be denoted by $\mathcal{P}(n,d)$. The class $\mathcal{P}(n,1)$ is, of course, simply as the set of threshold functions T_n on $\{0,1\}^n$. (Note that we have used the earlier observation that, for Boolean inputs to the polynomial threshold unit, no powers of x_i other than 0 or 1 are needed; so S ranges over subsets rather than multi-subsets of $[n]$.)

Asummability and polynomial separability

We have already observed that a function is a linear threshold function if and only if the true points can be separated from the false points by a hyperplane. For a polynomial threshold function of degree m, we have the corresponding geometrical characterization that the true points can be separated from the false points by a surface whose equation is a polynomial of degree m.

It is possible to relate such polynomial separation to linear separation in a higher-dimensional space [47, 9]. For $x \in \{0,1\}^n$, we define the *m-augment*, $x^{(m)}$, of x to be the $\{0,1\}$-vector

of length $\sum_{i=1}^{m} \binom{n}{i}$ whose entries are x_S for $\emptyset \neq S \in [n]^{(m)}$ in some prescribed order. To be precise, we shall suppose the entries are in order of increasing degree and that terms of the same order are listed in *lexicographic (dictionary) order*. Thus, for example, when $n = 5$ and $m = 2$,

$$x^{(5)} = (x_1, x_2, x_3, x_4, x_5, x_1x_2, x_1x_3, x_1x_4, x_1x_5, x_2x_3, x_2x_4, x_2x_5, x_3x_4, x_3x_5, x_4x_5).$$

We observe that a Boolean function f is a polynomial threshold function of degree m if and only if there is some linear threshold function h_f, defined on $\{0, 1\}$ vectors of length $r = \sum_{i=1}^{m} \binom{n}{i}$, such that

$$f(x) = 1 \Longleftrightarrow h_f(x^{(m)}) = 1;$$

that is, if and only if the m-augments of the true points of f and the m-augments of the false points of f can be separated by a hyperplane in the higher-dimensional space $\mathbb{R}^r$, where $r = \sum_{i=1}^{m} \binom{n}{i}$.

The m-augments can be used to provide an asummability criterion similar to Theorem 3.1.

We say that f is m-asummable if for any $k \in \mathbb{N}$, for any sequence $x_1, x_2, \ldots, x_k$ of (not necessarily distinct) true points of f and any sequence $y_1, y_2, \ldots, y_k$ of (not necessarily distinct) false points of f,

$$\sum_{i=1}^{k} x_i^{(m)} \neq \sum_{i=1}^{k} y_i^{(m)}.$$

Note that if f is m-asummable then f is m'-asummable for any $m' > m$. The following result holds [47].

Theorem 3.5 *The Boolean function f is a threshold function of degree m if and only if f is m-asummable.*

Number of polynomial threshold functions

We can obtain an upper bound on the number of polynomial threshold functions of a given degree by using Theorem 3.2, together with the fact that a Boolean function is a polynomial threshold function of degree m if and only if the m-augments of true points and the m-augments of the false points are linearly separable.

Theorem 3.6 *The number, $|\mathcal{P}(n,m)|$ of polynomial threshold functions of degree m on $\{0,1\}^n$ satisfies*

$$|\mathcal{P}(n,m)| \leq 2^{\sum_{i=1}^{m}\binom{n}{i}} \sum_{k=0} \binom{2^n-1}{k}.$$

for all m, n with $1 \leq m \leq n$.

It is fairly easy to deduce from this that $\log_2 |T(n,m)|$ is at most $n\binom{n}{m} + O(n^m)$ as $n \to \infty$, with $m = o(n)$.

Saks [38] observed that $|\mathcal{P}(n,m)| \geq |\mathcal{P}(n-1,m)||\mathcal{P}(n-1,m-1)|$, for $2 \leq m \leq n-1$. From this, it follows [38, 1] that:

Theorem 3.7 *The number, $|\mathcal{P}(n,m)|$, of polynomial threshold functions of degree m on $\{0,1\}^n$ satisfies $|\mathcal{P}(n,m)| \geq 2^{\binom{n}{m+1}}$. for all m, n with $1 \leq m \leq n-1$.*

Note that this lower bound is not at all tight for $m > n/2$. However, for constant m it provides a good match for the upper bound of Theorem 3.6. Taken together, the results imply that, for fixed m, for some positive constants c, k, $\log_2 |T(n,m)|$ is, between cn^{m+1} and kn^{m+1}.

Threshold order

A Boolean function is said to be a k-DNF function if it has a DNF formula in which each term is of degree at most k. It is easy to see that any k-DNF f on $\{0,1\}^n$ is in $\mathcal{P}(n,k)$, as follows. Given a term $T_j = u_{i_1}u_{i_2}\ldots u_{i_r}\bar{u}_{j_1}\bar{u}_{j_2}\ldots\bar{u}_{j_s}$ of the DNF, we form the expression

$$A_j = x_{i_1}x_{i_2}\ldots x_{i_r}(1-x_{j_1})(1-x_{j_2})\ldots(1-x_{j_s}).$$

We do this for each term $T_1, T_2, \ldots, T_l$ and expand the algebraic expression $A_1 + A_2 + \cdots + A_l$ according to the normal rules of algebra, until we obtain a linear combination of the form $\sum_{S\in[n]^{(k)}} w_S x_S$. Then, since $f(x) = 1$ if and only if $A_1 + A_2 + \cdots + A_l > 0$, it follows that

$$f(x) = \operatorname{sgn}\left(\sum_{S\in[n]^{(k)}} w_S x_S\right),$$

so $f \in \mathcal{P}(n,k)$. Thus, any k-DNF function is also a polynomial threshold function of degree at most k.

Generally, given a Boolean function f, the *threshold order* [47, 30] of f is the least k such that $f \in \mathcal{P}(n,k)$. We mention that there are always (exactly) two functions with threshold order n, namely the parity function PARITY_n (defined by $\mathrm{PARITY}_n(x) = 1$ if and only if x has an odd number of entries equal to 1) and its complement; see [47].

A very precise behaviour of the 'expected' threshold order has been conjectured by Wang and Williams [47]. Roughly speaking, the conjecture says that, for large even numbers n, almost all the Boolean functions on $\{0,1\}^n$ have threshold order equal to $n/2$; and that for large odd n, almost every function has threshold order $(n-1)/2$ or $(n+1)/2$, with an equal split between these. To make this precise, we introduce some notation. Let $\sigma(n,k)$ denote the proportion of Boolean functions of n variables with threshold order k; thus,

$$\sigma(n,k) = \frac{|\mathcal{P}(n,k)| - |\mathcal{P}(n,k-1)|}{2^{2^n}}.$$

Wang and Williams conjectured that for even values of n, $\sigma(n,n/2) \to 1$ as $n \to \infty$ and that for odd values of n, $\sigma(n,(n-1)/2) \to 1/2$ and $\sigma(n,(n+1)/2) \to 1/2$ as $n \to \infty$. The following observation [3] provides a partial proof of this.

Theorem 3.8 *For $k = k(n) \leq \lfloor n/2 \rfloor - 1$, $\sigma(n,k(n)) \to 0$ as $n \to \infty$. Furthermore, for all odd n, $\sigma\big(n, \lfloor \frac{n}{2} \rfloor\big) \leq 1/2$.*

This result shows, among other things, that the representational power of $\mathcal{P}(n,k)$ is limited unless k is of the same order as n. In particular, it might be said that the 'typical' Boolean function has threshold order at least $\lfloor n/2 \rfloor$.

Some progress has been made on the remaining parts of the conjecture. As reported in [38], Alon, using a result of Gotsman [12] on the harmonic analysis of Boolean functions, showed that there is a fixed constant $\epsilon > 0$, such that almost all Boolean functions of n variables have threshold order less than $(1-\epsilon)n$; that is, $\sigma(n,(1-\epsilon)n) \to 0$ as $n \to \infty$. This has recently been improved upon by Samorodnitsky [39], who has shown (again, using harmonic analysis) that almost all Boolean functions have threshold order at most $\frac{n}{2} + O(\sqrt{n \log n})$.

3.3 Linear threshold networks

We now move on to consider the representation of Boolean functions by feed-forward linear threshold networks (which we will refer to as threshold networks for the sake of brevity). Single linear threshold units have very limited computational abilities, but we can easily see that any Boolean function can be represented by a threshold network with one hidden layer. It is natural to ask how small a threshold network can be used for particular functions or types of functions. Questions like this bring us into the realm of circuit complexity, (in which threshold networks are usually referred to as *threshold circuits*) a large area which we will only very briefly touch on here.

The existence of a DNF formula for every Boolean function can be used to show that any Boolean function can be computed by a two-layer feed-forward threshold network.

Theorem 3.9 *There is a 2-layer threshold network capable of computing any Boolean function.*

Proof: Suppose that $f : \{0,1\}^n$, and let ϕ be the DNF formula obtained as the disjunction of the prime implicants of f. Suppose $\phi = T_1 \vee T_2 \vee \cdots \vee T_k$, where each T_j is a term of the form $T_j = \left(\bigwedge_{i \in P_j} u_i\right) \bigwedge \left(\bigwedge_{j \in N_j} \bar{u}_j\right)$, for some disjoint subsets P_j, N_j of $\{1, 2, \ldots, n\}$. Suppose that the network has 2^n hidden units, and let us set the weights to and from all but the first k of these to equal 0, and the corresponding thresholds equal to 1 (so the effect is as if these units were absent). Then for each of the first k units, let the weight-vector $\alpha^{(j)}$ from the inputs to unit j correspond directly to T_j, in that $\alpha_i^{(j)} = 1$ if $i \in P_j$, $\alpha_i^{(j)} = -1$ if $i \in N_j$, and $\alpha_i^{(j)} = 0$ otherwise. We take the threshold on unit j to be $|P|$, the weight on the connection between the unit and the output unit to be 1, and the threshold on the output unit to be $1/2$. It is clear that unit j outputs 1 on input x precisely when x satisfies T_j, and that the output unit computes the 'or' of all the outputs of the hidden units. Thus, the output of the network is the disjunction of the terms T_j, and hence equals f.

A *universal network* for Boolean functions on $\{0,1\}^n$ is a threshold network which is capable of computing every Boolean function of n variables. Theorem 3.9 shows that the two-layer threshold network with n inputs, 2^n units in the hidden layer, and one output unit, is universal. The question arises as to whether there is a universal network with fewer threshold units. By an easy counting argument, one can obtain a lower bound on the size of *any* universal network, regardless of its structure. In particular (see [43, 33]), any universal network (regardless of how

many layers it has) must have at least $\Omega\left(2^{n/2}/\sqrt{n}\right)$ threshold units. Moreover, any two-layer universal network for Boolean functions must have at least $\Omega(2^n/n^2)$ threshold units.

Much work in circuit complexity has gone into consideration of the sizes of threshold network needed to compute particular Boolean functions. Of particular interest has been the parity function PARITY_n. Many sophisticated techniques have been used to produce lower bounds on the sizes of networks (with particular numbers of layers, for example) capable of computing parity; see [43]. One such result is that any two-layer threshold network capable of computing PARITY_n, must have $\Omega(\sqrt{n})$ units.

3.4 Spiking neurons

We have observed that if all delays on a spiking neuron are set to zero, then the neuron behaves exactly like a linear threshold unit. So the spiking neuron is at least as powerful as the linear threshold unit and the set S_n of Boolean functions it computes is at least as large as T_n. However, S_n is not significantly larger than T_n, for as shown by Maass and Schmitt [28], $\log_2 |S_n| \leq n^2 + O(\log n)$, whereas, as noted above, $\log_2 |T_n|$ is $n^2(1 + o(1))$.

By the way the neuron acts, the weighted signal from input i is 'active' (if at all) on the time interval $[d_i, d_i + 1)$ and the output of the neuron is 1 if and only if the sum of active weighted inputs exceeds the threshold, at some time. By partitioning the time axis into intervals on which the same weighted inputs are active, it can be seen [28] that there are at most $2n-1$ intervals on which the sum of active weighted inputs is constant. (For, there are at most $2n$ times at which the set of active weighted inputs can change.) Hence, the neuron fires if one of these $2n - 1$ sums exceeds the threshold. Thus, we obtain the result from [28] that any function in S_n can be expressed as a disjunction of at most $2n - 1$ threshold functions. Schmitt [41] improved this to $n-1$. Hammer *et al.* [13] defined the *threshold number* of a Boolean function to be the smallest number of threshold functions of which it is a conjunction (a number that is well-defined and at most 2^{n-1} by a result of Jeroslow [18]). Thus, this result may be re-phrased as saying that any function in S_n has threshold number at most $n - 1$. That there are functions in S_n quite different from threshold functions has been indicated by Schmitt [41], who showed that there is a function in S_n with threshold number at least $\lfloor n/2 \rfloor$ (whereas, of course, any function in T_n has threshold number 1). (He also shows, however, that there is some function of threshold number 2 that is not in S_n.)

Further differences between the spiking neuron and the threshold (and polynomial threshold) unit emerge when the threshold order of computable functions is considered [41]. Whereas the threshold order of any function in T_n is 1, there are functions in S_n with threshold order $n^{1/3}/4^{1/3}$. This shows, additionally, that the functions in S_n cannot be computed by a polynomial threshold unit of any fixed degree. (Schmitt also shows that some Boolean function of threshold order 2 is not in S_n.)

A Boolean function is a μ-DNF function if it has a DNF formula in which each variable appears, either negated or not, at most once. Maass and Schmitt [28] showed that any μ-DNF function can be computed by a spiking neuron, and that, by contrast, there are μ-DNF functions that cannot be computed by a linear threshold unit.

4 Expressive power of neural networks

4.1 Growth function and VC-dimension

Definitions

The number of examples needed for valid learning in standard probabilistic models of learning can be quantified fairly precisely by the VC-dimension of the class of functions being used as hypotheses (that is, as the functions chosen to approximate to the training data); see [2, 8], for example . In this sense, the VC-dimension is a useful way of measuring the expressive power of a set of functions. In this section, we examine the growth functions and VC-dimensions of the sets of functions computable by certain types of neural networks.

We start by recalling what is meant by the growth function and VC-dimension. Suppose that H is a set of functions from a set X to $\{0, 1\}$. (So, when H is the set of functions computable by an n-input neural network, X will be $\mathbb{R}^n$ or — the case of most interest to us — $\{0, 1\}^n$.) For a finite subset S of X, $\Pi_H(S)$ denotes the cardinality of the set of functions $H|_S$, obtained by restricting H to domain S. For $m \in \mathbb{N}$, $\Pi_H(m)$ is defined to be the maximum of $\Pi_H(S)$ over all subsets of cardinality m. For all m, $\Pi_H(m) \leq 2^m$. The *Vapnik-Chervonenkis dimension* [46, 8] of H is defined as the maximum m (possibly infinite, in the case where the domain is $\mathbb{R}^n$) such that $\Pi_H(m) = 2^m$. We say that $S \subseteq X$ is *shattered* by H, or that H *shatters* S, if

$\Pi_H(S) = 2^{|S|}$; that is, if H gives all possible classifications of the points of S. Thus, S is shattered by H if for each subset R of S, there is some function f_R in H such that for $1 \leq i \leq m$, $f_R(x_i) = 1 \Longleftrightarrow x_i \in R$.

The neural networks considered in this report compute a class of $\{0,1\}$-valued functions. So we can define the VC-dimension of a neural network to be the VC-dimension of the set of functions computable by the network. For a network $\mathcal{N}$ with n inputs, we denote by $\mathrm{VCdim}(\mathcal{N}, \mathbb{R}^n)$ the VC-dimension of the class of functions from $\mathbb{R}^n \rightarrow \{0,1\}^n$ computed by $\mathcal{N}$ and $\mathrm{VCdim}(\mathcal{N}, \{0,1\}^n)$ will denote the VC-dimension of the corresponding class of Boolean functions. In this report, we shall be primarily interested in the VC-dimension of the set of Boolean functions computable by the network.

VC-dimension and linear dimension

There is a useful connection between linear (vector-space) dimension and the VC-dimension [10]. Suppose $\mathcal{V}$ is a set of real functions defined on some set X. For $f, g \in \mathcal{V}$ and $\lambda \in \mathbb{R}$, we can form the function $f + g : X \rightarrow \mathbb{R}$ by *pointwise addition* and the function $\lambda f : X \rightarrow \mathbb{R}$ by *pointwise scalar multiplication*, as follows:

$$(f+g)(x) = f(x) + g(x), \quad (\lambda f)(x) = \lambda f(x), \quad (x \in X).$$

If $\mathcal{V}$ is closed under these operations, then it is a vector space of functions. Then, in $\mathcal{V}$, we say that the set $\{f_1, f_2, \ldots, f_k\}$ of functions is *linearly dependent* if there are constants λ_i $(1 \leq i \leq k)$, not all zero, such that, *for all* $x \in X$,

$$\lambda_1 f_1(x) + \lambda_2 f_2(x) + \cdots + \lambda_k f_k(x) = 0;$$

that is, some non-trivial linear combination of the functions is the zero function on X. The vector space $\mathcal{V}$ is finite-dimensional, of linear dimension d, if the maximum cardinality of a linearly independent set of functions in $\mathcal{V}$ is d. We have the following result, due to Dudley [10].

Theorem 4.1 *Let $\mathcal{V}$ be a real vector space of real-valued functions defined on a set X. Suppose that $\mathcal{V}$ has linear dimension d. For any $f \in \mathcal{V}$, define the $\{0,1\}$-valued function f_+ on X by*

$$f_+(x) = \begin{cases} 1 & \text{if } f(x) \geq 0 \\ 0 & \text{if } f(x) < 0, \end{cases}$$

and let $\mathrm{sgn}(\mathcal{V}) = \{f_+ : f \in \mathcal{V}\}$. Then the VC-dimension of $\mathrm{sgn}(\mathcal{V})$ is d.

4.2 Linear threshold units

The VC-dimension of the single linear threshold unit can be bounded fairly directly using Theorem 4.1. For, the class of functions in question is precisely $\mathrm{sgn}(\mathcal{V})$ where $\mathcal{V}$ is the set of affine functions, of the form $x \mapsto w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$, for some constants $w_0, w_1, \ldots, w_n$. The set $\mathcal{V}$ is easily seen to be a vector space of linear dimension $n+1$, and hence has VC-dimension $n+1$. In fact, this is so even if we restrict the inputs to $\{0,1\}^n$:

Theorem 4.2 *The VC-dimension of T_n, the set of (Boolean) threshold functions, is $n+1$.*

Proof: We have already indicated why the VC-dimension of the set of functions computable by the threshold unit on $\mathbb{R}^n$ is $n+1$. Certainly, we must therefore have $\mathrm{VCdim}(T_n)$ no more than $n+1$, since T_n is a restriction to the Boolean domain, $\{0,1\}^n$, of this class. So the result will follow if we show that $\mathrm{VCdim}(T_n) \geq n+1$. We do this by proving that a particular subset of $\{0,1\}^n$ of cardinality $n+1$ is shattered by the T_n. Let $\mathbf{0}$ denote the all-0 vector and, for $1 \leq i \leq n$, let e_i be the point with a 1 in the ith coordinate and all other coordinates 0. We shall show that T_n shatters the set $S = \{\mathbf{0}, e_1, e_2, \ldots, e_n\}$. Suppose that R is any subset of S and, for $i = 1, 2, \ldots, n$, let

$$w_i = \begin{cases} 1, & \text{if } e_i \in R; \\ -1, & \text{if } e_i \notin R; \end{cases}$$

and let

$$\theta = \begin{cases} -1/2, & \text{if } \mathbf{0} \in R; \\ 1/2, & \text{if } \mathbf{0} \notin R. \end{cases}$$

Then it is straightforward to verify that if h_R is the function computed by the threshold unit when the weight-vector is $w = (w_1, w_2, \ldots, w_n)$ and the threshold is θ, then the set of positive examples of h_R in S is precisely R. Therefore S is shattered by T_n and, consequently, $\mathrm{VCdim}(T_n) \geq n+1$. □

Theorem 3.2 shows that

$$\Pi_H(m) \leq 2 \sum_{k=0}^{n} \binom{m-1}{k}.$$

This upper bound is easily seen to equal 2^m for $m \leq n+1$ and to be less than 2^m for $m > n+1$, from which it follows also that $\mathrm{VCdim}(T_n) \leq n+1$.

4.3 Polynomial threshold units

We now bound the VC-dimension of the class $\mathcal{P}(n,m)$ of (Boolean) polynomial threshold functions of degree m. Recall that such a function takes the form

$$f(x) = \operatorname{sgn}\left(\sum_{S\in[n]^m} w_S x_S\right),$$

for some $w_S \in \mathbb{R}$, where $[n]^{(m)}$ is the set of subsets of at most m elements from $\{1,2,\ldots,n\}$ and x_S denotes the product of the x_i for $i \in S$. For $m \leq n$, let $C(n,m) = \left\{x_S : S \in [n]^{(m)}\right\}$, regarded as a set of real functions on domain $\{0,1\}^n$.

Theorem 4.3 *For all n, m with $m \leq n$, $C(n,m)$ is a linearly independent set of real functions defined on $\{0,1\}^n$.*

Proof: Let $n \geq 1$ and suppose that for some constants c_S and for all $x \in \{0,1\}^n$,

$$A(x) = \sum_{S\in[n]^{(m)}} c_S x_S = 0.$$

Set x to be the all-0 vector to deduce that $c_\emptyset = 0$. Let $1 \leq k \leq m$ and assume, inductively, that $c_S = 0$ for all $S \subseteq [n]$ with $|S| < k$. Let $S \subseteq [n]$ with $|S| = k$. Setting $x_i = 1$ if $i \in S$ and $x_j = 0$ if $j \notin S$, we deduce that $A(x) = c_S = 0$. Thus for all S of cardinality k, $c_S = 0$. Hence $c_S = 0$ for all S, and the functions are linearly independent. □

It is therefore immediate, from Theorem 4.1, that for all n, m with $m \leq n$,

$$\operatorname{VCdim}(\mathcal{P}(n,m)) = \sum_{i=0}^{m} \binom{n}{i}.$$

A similar analysis will determine the VC-dimension of the set of functions from $\mathbb{R}^n$ to $\{0,1\}$ computable by the polynomial threshold unit. In this case, the set of functions of degree m is $\operatorname{sgn}(\mathcal{V})$, where $\mathcal{V}$ is the vector space with basis x_S for all $\binom{n+m}{m}$ multi-sets of at most m elements from $[n]$. So the VC-dimension in this case is $\binom{n+m}{m}$. To sum up, we have the following results.

Theorem 4.4 *Let $\mathcal{N}$ be a single n-input polynomial threshold unit of degree m. Then, for all $m, n \in \mathbb{N}$,*

$$\mathrm{VCdim}(\mathcal{N}, \mathbb{R}^n) = \binom{n+m}{m},$$

and for all n, m with $m \leq n$,

$$\mathrm{VCdim}(\mathcal{N}, \{0,1\}^n) = \sum_{i=0}^{m} \binom{n}{i}.$$

We have only considered single polynomial threshold units here, but clearly networks could be formed from such units. The VC-dimensions of the resulting networks (and of further generalizations of these types of network) have been bounded by Schmitt [42].

4.4 Linear threshold networks

We now provide a bound on the VC-dimension of feed-forward linear threshold networks. This is a slightly weaker version (with an easier proof, from [24]) of a bound due to Baum and Haussler [6].

Theorem 4.5 *Suppose that $\mathcal{N}$ is a feed-forward linear threshold network having a total of W variable weights and thresholds, and n inputs. Then*

$$\mathrm{VCdim}\left(\mathcal{N}, \{0,1\}^n\right) \leq \mathrm{VCdim}\left(\mathcal{N}, \mathbb{R}^n\right) < 6W \log_2 W.$$

Proof: Let $X = \mathbb{R}^n$ and suppose that $S \subseteq X$ is of cardinality m. Let H be the set of functions computable by $\mathcal{N}$. We bound the growth function of H by bounding $\Pi_H(S)$ independently of S. Denote by N the number of computation units (that is, the number of linear threshold neurons) in the network. Since the network is a feed-forward network, the computation units may be labeled with the integers $1, 2, \ldots, N$ so that if the output of threshold unit i feeds into unit j then $i < j$. Consider any particular threshold unit, i. Denote the in-degree of i by d_i. By Theorem 3.2, the number of different ways in which a set of m points can be classified by unit

i is at most $2\sum_{k=0}^{d_i}\binom{m-1}{k}$, which is certainly at most m^{d_i+2} for $m \geq d_i + 1$. It follows that, (if $m > \max_i d_i + 1$) the number of classifications $\Pi_H(S)$ of S by the network is bounded by

$$m^{d_1+2}m^{d_2+2}\ldots m^{d_N+2},$$

which, since $W = d_1 + d_2 + \ldots + d_N + N$, the total number of weights and thresholds, is at most m^{W+N}. Since $W \geq N$ (there being a threshold for each threshold unit), this is at most m^{2W}. Now, $m^{2W} < 2^m$ if $m = 6W\log_2 W$, from which it follows that the VC-dimension of the network is less than $6W\log_2 W$. □

With more careful bounding [6], the VC-dimension can be bounded above by $2W\log_2(eN)$. This upper bound is of order $W\ln N$ where W is the total number of weights and thresholds; that is, the total number of variable parameters determining the state of the network. We have already seen that the linear threshold unit on n inputs has VC-dimension $n+1$, which is exactly the number of variable parameters in this case. We have also seen that for polynomial threshold functions, the VC-dimension is precisely the number of variable parameters. The question therefore arises as to whether the $O(W\ln N)$ bound is of the best possible order or whether in this case, too, the VC-dimension is of order W. In fact, the $\ln N$ factor cannot, in general, be removed, as the following result of Maass [23] shows.

Theorem 4.6 *Let W be any positive integer greater than 32. Then there is a three-layer feed-forward linear threshold network $\mathcal{N}_W$ with at most W weights and thresholds, for which $\mathrm{VCdim}(\mathcal{N}_W, \{0,1\}^n) > (1/132)W\log_2(N/16)$, where N is the number of computation units.*

4.5 Sigmoid networks

Bounding the VC-dimension of sigmoid networks is rather more complicated than for threshold networks. Finiteness of the VC-dimension of sigmoid networks was established by Macintyre and Sontag [29], using deep results from logic. This in itself was a significant result, since it had previously been shown by Sontag [45] that for small networks with activation function other than the standard sigmoid, σ, the VC-dimension could be infinite. (Indeed, there are activation functions very similar to the standard sigmoid, for which the VC-dimension of a very small corresponding network is infinite; see [2].) The following result of Karpinksi and Macintyre [19, 20] provides concrete, polynomial, upper bounds on the VC-dimension of sigmoid networks. The proof, which is quite involved, brings together techniques from logic and algebraic geometry. (See also [2].)

Theorem 4.7 *Let $\mathcal{N}$ be a feed-forward sigmoid network. Suppose that the total number of adjustable weights and thresholds is W, that the number of inputs is n, and that there are N computation units. Then*

$$\mathrm{VCdim}(\mathcal{N}, \mathbb{R}^n) \leq (WN)^2 + 11WN \log_2(18WN^2).$$

Note that this bound, which is $O(W^4)$, is polynomial in the number of weights and thresholds. It has been shown by Koiran and Sontag [21, 22] that the VC-dimension of (unbounded depth) sigmoid nets can be as large as W^2. There is thus, generally, a strict separation between the VC-dimension of threshold networks (with VC-dimension $O(W \ln W)$) and sigmoid networks.

4.6 Spiking neurons

Recall that S_n denotes the set of Boolean functions computable by the n-input spiking neuron. Maass and Schmitt [28] obtained the following result on the VC-dimension of a single spiking neuron.

Theorem 4.8 *The VC-dimension of S_n, the set of functions computable by a spiking neuron on $\{0,1\}^n$, is $O(n \log n)$ and $\Omega(n \log n)$. Moreover, this lower bound is also true for a subclass of S_n in which the weights and threshold are kept fixed and only the delay parameters are varied.*

Thus, although, as noted earlier, there are not significantly many more Boolean functions computable by the spiking neuron than by the threshold unit, the spiking neuron is considerably more expressive. For, the VC-dimension of the linear threshold unit is $n + 1$, whereas the VC-dimension of the spiking neuron is $\Theta(n \log n)$.

The VC-dimension of feed-forward networks of spiking neurons has also been investigated. Maass and Schmitt [28] proved that for each n, there is a network of this type with $O(n)$ edges, for which, varying only the delays (and leaving weights and threshold fixed), the resulting class of functions defined on $\{0,1\}^n$ has VC-dimension $\Omega(n^2)$. Note that, here, only the delays are variable and there are $O(n)$ of these. Thus the VC-dimension is at least quadratic in the number of variable delays. Recall that any linear threshold network has VC-dimension $O(W \log W)$ where W is the number of weights and thresholds. Thus, the VC-dimension of a network of

spiking neurons with a given number of adjustable delays (and weights and threshold fixed) can be larger than the VC-dimension of a threshold network with the same number of adjustable weights and thresholds. Maass and Schmitt also showed that any such network has a VC-dimension (over inputs from $\mathbb{R}^n$) which is at most $O(E^2)$ where E is the number of edges in the underlying digraph (that is, the number of network connections). So the VC-dimension is at most quadratic in the number of variable weights, thresholds, and delays, and their lower bound is asymptotically tight.

References

[1] M. Anthony. *Discrete Mathematics of Neural Networks: Selected Topics*. SIAM Monographs on Discrete Mathematics and Applications DT08, SIAM: Philadelphia, 2001.

[2] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[3] M. Anthony. Classification by polynomial surfaces, *Discrete Applied Mathematics*, 61, 1996: 91–103.

[4] M. Anthony, G. Brightwell and J. Shawe-Taylor. On specifying Boolean functions by labelled examples. *Discrete Applied Mathematics*, 61, 1995: 1–25.

[5] M. Anthony and J. Shawe-Taylor", Using the perceptron algorithm to find consistent hypotheses. *Combinatorics, Probability and Computing*, 4(2), 1993: 385–387.

[6] E. Baum and D. Haussler. What Size Net Gives Valid Generalization?. *Neural Computation*, 1(1), 1989: 151–160.

[7] C. M. Bishop. *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 1995.

[8] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth: Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4), 1989: 929–965.

[9] T. M. Cover. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *IEEE Trans. on Electronic Computers*, EC-14, 1965: 326–334.

[10] R.M. Dudley, Central limit theorems for empirical measures, *Ann. Probability* 6, 1978: 899–929.

[11] S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1), 1995: 20–31.

[12] C. Gotsman, On Boolean functions, polynomials and algebraic threshold functions. Technical report TR-89-18, Department of Computer Science, Hebrew University, 1989.

[13] P. L Hammer, T. Ibaraki and U. N. Peled. Threshold numbers and threshold completions. *Annals of Discrete Mathematics* 11, 1981: 125–145.

[14] S.E. Hampson and D.J. Volper, Linear function neurons: structure and training. *Biological Cybernetics* 53, 1986: 203–217.

[15] J. Håstad. On the size of weights for threshold gates, *SIAM Journal on Discrete Mathematics*, 7(3), 1994: 484–492.

[16] J. Hertz, A. Krogh and R. G. Palmer. *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, California, 1991.

[17] S-T. Hu, *Threshold Logic*, University of California Press, Berkeley, 1965.

[18] R.G. Jeroslow. On defining sets of vertices of the hypercube by linear inequalities. *Discrete Mathematics*, 11, 1975: 119–124.

[19] M. Karpinski and A. J. Macintyre. Polynomial bounds for VC dimension of sigmoidal neural networks, in *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, ACM Press, New York, NY, 1995: 200-208.

[20] M. Karpinski and A. J. Macintyre. Polynomial Bounds for VC Dimension of Sigmoidal and General Pfaffian Neural Networks, *Journal of Computer and System Sciences*, 54, 1997: 169–176.

[21] P. Koiran and E. D. Sontag. Neural Networks with Quadratic VC Dimension, *Journal of Computer and System Sciences*, 54(1), 1997: 190–198.

[22] P. Koiran and E. D. Sontag. Neural Networks with Quadratic VC Dimension. In *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer and M. E. Hasselmo (eds). MIT Press, 1996: 197–203.

[23] W. Maass. Bounds for the computational power and learning complexity of analog neural nets, In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, ACM Press, New York, NY, 1993: 335–344.

[24] W. Maass. On the complexity of learning in feedforward neural nets. Manuscript, Institute for Theoretical Computer Science, Technische Universitaet Graz, 1993.

[25] W. Maass. On the relevance of time in neural computation and learning. In *Proceedings of the 8th International Workshop on Algorithmic Learning Theory, ALT'97* (M. Li and A. Maruoka, eds). Springer, Berlin, 1997.

[26] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks* (10), 1997: 1659–1671.

[27] W. Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation* 8, 1996: 1–40.

[28] W. Maass and M. Schmitt. On computing Boolean functions by a spiking neuron.

[29] A. Macintyre and E. D. Sontag. Finiteness results for sigmoidal "neural" networks. In *Proceedings 25th Annual ACM Symposium on the Theory of Computing*. ACM Press, New York, NY, 1993: 325–334.

[30] M. Minsky and S. Papert, *Perceptrons*. MIT Press, Cambridge, MA., 1969. (Expanded edition 1988.)

[31] S. Muroga. Lower bounds of the number of threshold functions and a maximum weight, *IEEE Transactions on Electronic Computers*, 14, 1965: 136–148.

[32] S. Muroga, *Threshold Logic and its Applications*, Wiley, New York, 1971.

[33] E. I Nechiporuk. The synthesis of networks from threshold elements, *Problemy Kibernetiki*, 11, 1964: 49–62.

[34] N.J. Nilsson, *Learning Machines*, McGraw-Hill, New York, 1965.

[35] I. Parberry. *Circuit Complexity and Neural Networks*, Foundations of Computing Series, MIT Press, 1994.

[36] R. Rivest. Learning Decision Lists. *Machine Learning* 2 (3), 1987: 229–246.

[37] D.E. Rumelhart, G. E. Hinton and J. L. McClelland. A general framework for parallel distributed processing. In Rumelhart, D.E. and McClelland, J.L. (eds), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* Volume 1. MIT Press, Cambridge, MA.

[38] M. Saks, Slicing the hypercube, in *Surveys in Combinatorics, 1993*, (ed. K. Walker), Cambridge University Press, 1993.

[39] A. Samorodnitsky. Unpublished (personal communication).

[40] L. Schläfli. *Gesammelte Mathematische Abhandlungen I*, Birkhäuser, Basel, 1950.

[41] M. Schmitt. On computing Boolean functions by a spiking neuron. *Annals of Mathematics and Artificial Intelligence* 24, 1998: 181–191.

[42] M. Schmitt. On the complexity of computing and learning with multiplicative neural networks. *Neural Computation* 14(2), 2002: 241–301.

[43] K-Y. Siu, V. Roychowdhury and T. Kailath. *Discrete Neural Computation: A Theoretical Foundation*, Prentice Hall Information and System Sciences Series. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[44] W. Softky and C. Koch. Single-cell models. In *The Handbook of Brain Theory and Neural Networks*, ed. M. A. Arbib. MIT Press, Cambridge, MA, 1995: 879–884.

[45] E. D. Sontag. Feedforward nets for interpolation and classification. *Journal of Computer and System Sciences*, 45, 1992: 20–48.

[46] V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2), 1971: 264–280.

[47] C. Wang and A.C. Williams, The threshold order of a Boolean function, *Discrete Applied Mathematics*, 31, 1991: 51–69.

[48] R. J. Williams. The logic of activation functions. In Rumelhart, D.E. and McClelland, J.L. (eds), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* Volume 1. MIT Press, Cambridge, MA.

[49] Y. A. Zuev. Asymptotics of the logarithm of the number of threshold functions of the algebra of logic. *Soviet Mathematics Doklady*, 39, 1989: 512–513.