Computing on Analog Neural Nets with Arbitrary Real Weights

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

1 Introduction

We examine in this chapter the computational power of high order analog feedforward neural nets $\mathcal{N}$, i.e. of circuits with analog computational elements in which certain parameters are treated as programmable parameters. We focus on neural nets $\mathcal{N}$ of bounded depth in which each gate g computes a function from $\mathbf{R}^m$ into $\mathbf{R}$ of the form $< y_1, \dots, y_m > \; \mapsto \gamma^g(Q^g(y_1, \dots, y_m))$. We assume that for each gate g , γ^g is some fixed piecewise polynomial activation function (also called response function). This function is applied to some polynomial $Q^g(y_1, \dots, y_m)$ of bounded degree with arbitrary real coefficients, where $y_1, \dots, y_m$ are the real valued inputs to gate g. One usually refers to the degree of the polynomial Q^g as the "order" of the gate g. The coefficients ("weights") of Q^g are the programmable variables of $\mathcal{N}$, whose values may arise from some learning process.

We are primarily interested in the case where the neural net $\mathcal{N}$ computes (respectively learns) a boolean valued function. For that purpose we assume that the real valued output of the output gate g_{out} of $\mathcal{N}$ is "rounded off". More precisely, we assume that there is an "outer threshold" T_{out} (which belongs to the programmable parameters of $\mathcal{N}$) such that the output of $\mathcal{N}$ is "1" whenever the real valued output z of g_{out} satisfies $z \geq T_{out}$, and "0" if $z < T_{out}$. In some results of this chapter we also assume that the input $< x_1, \dots, x_n >$ of $\mathcal{N}$ is boolean-valued. It should be noted, that this does not affect the capacity of $\mathcal{N}$ to carry out on its intermediate levels (i.e. in its "hidden units") computation over reals, whose real-valued results are then transmitted to the next layer of gates.

Circuits of this type have rarely been considered in computational complexity theory, and they give rise to the principal question whether these intermediate *analog* computational elements will allow the circuit to compute more complex boolean functions than a circuit with a similar layout but *digital* computational elements. Note that circuits with analog computational elements have an extra source of potentially unlimited parallelism at their disposal, since they can execute operations on numbers of arbitrary bit-length in one step, and they can transmit numbers of arbitrary bit-length from one gate to the next.

One already knows quite a bit about the special case of such neural nets $\mathcal{N}$ where each gate g is a "linear threshold gate". In this case each polynomial $Q^g(y_1, \ldots, y_m)$ is of degree ≤ 1 (i.e. a weighted sum), and each activation function γ^g in $\mathcal{N}$ is the "heaviside function" (also called "hard limiter") sgn defined by

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

(e.g. see [R], [Ni], [Mu], [MP], [PS], [HMPST], [GHR], [SR], [SBKH], [BH], [A], [L]). The "analog versus digital" issue does not arise in this case, since the output of each gate is a single bit. Still, it requires some work to bound the potential power of arbitrary weights (in the weighted sums) for the computation of boolean functions on such circuit. Since there are only finitely many boolean circuit inputs, it is obvious that only rational weights have to be considered. The key result for the analysis of these circuits was the discovery of Muroga et. al. (see [Mu]) that it is sufficient to consider for a linear threshold gate with m boolean inputs only weights $\alpha_1, \ldots, \alpha_m$ and a bias α_0 that are integers of size $2^{O(m \log m)}$ (this upper bound is optimal according to a recent result of Hastad [Has]). With the help of this a-priori-bound on the relevant bit-length of weights it is easy to show that the same arrays $(F_n)_{n \in \mathrm{N}}$ of boolean functions $F_n : \{0,1\}^n \to \{0,1\}$ are computable by arrays $(\mathcal{N}_n)_{n \in \mathrm{N}}$ of neural nets of depth $O(1)$ and size $O(n^{O(1)})$ with linear threshold gates, no matter whether one uses as weights arbitrary reals, rationals, integers, or elements of $\{-1, 0, 1\}$; see [Mu], [CSV], [HMPST], [GHR], [MT]. The resulting class of arrays $(F_n)_{n \in \mathrm{N}}$ of boolean functions is called (nonuniform-) TC^0 ([HMPST], [J]).

In comparison, very little is known about upper bounds for the computational power and the learning complexity of feedforward neural nets whose gates g employ more general types of activation functions γ^g. This holds in spite of the fact that "real neurons and real physical devices have continuous input-output relations" (Hopfield [Ho]). In the analysis of information processing in natural neural systems, one usually views the firing rate of a neuron as its current output. Such firing rates are known to change between a few and several hundred spikes per second (see ch. 20 in [MR]). Hence the activation function γ^g of a gate g that models such a neuron should have a "graded response". It should also be noted that the customary learning algorithms for artificial neural nets (such as backwards propagation [RM]) are based on gradient descent methods, which *require* that all gates g employ *smooth* activation functions γ^g.

In addition, it has frequently been pointed out that it is both biologically plausible and computationally relevant to consider gates g that pass to γ^g instead of a weighted sum $\sum\limits_{i=1}^{m} \alpha_i y_i + \alpha_0$ some polynomial $Q^g(y_1, \ldots, y_m)$ of bounded degree, where $y_1, \ldots, y_m$ are circuit inputs or outputs of the immediate predecessors of g. Such gates are called sigma-pi units or high order gates in the literature (see p. 73 and ch. 10 in [RM], also [DR], [H], [PG], [MD]). From the point of view of approximation theory there has been particular interest in the case where $Q^g(y_1, \ldots, y_m) = \sum\limits_{i=1}^{m} \alpha_i (y_i - c_i)^2$ measures a "distance" of its input $< y_1, \ldots, y_m >$ from some "center" $< c_1, \ldots, c_m >$ (the latter may be determined through a learning process).

Theorem 3.1 of this chapter provides the first upper bound for the computational power of high order feedforward neural nets with non-boolean activation functions and arbitrary real weights. The power of feedforward neural nets with other activation functions besides sgn has previously been investigated in [RM] (ch.10), [S1], [S2], [H], [MSS], [DS], [SS]. It was shown in [MSS] for a very general class of activation functions γ^g that neural nets $(\mathcal{N}_n)_{n \in \mathrm{N}}$ of constant depth and size $O(n^{O(1)})$ with real weights of size $O(n^{O(1)})$ and output-separation $\Omega(1/n^{O(1)})$ (between the unrounded circuit-outputs for rejected and accepted inputs) can compute only boolean functions in TC^0. It follows from a result of Sontag [S2] that the assumptions on the weight-size and separation are essential for this upper bound: he constructed an arbitrarily smooth monotone function Θ (which can be made to satisfy the conditions on γ^g in the quoted result of [MSS]) and neural nets $\mathcal{N}_n$ of size 2 (!) with activation function Θ such that $\mathcal{N}_n$ can compute with sufficiently large weights *any* boolean function $F_n : \{0,1\}^n \to \{0,1\}$ (hence $\mathcal{N}_n$ has VC-dimension 2^n).

These results leave open the question about the computational power and learning complexity of feedforward neural nets with arbitrary weights that employ "natural" analog activation functions γ^g. For example there has previously been no upper bound for the set of boolean functions computable by analog neural nets with the very simple piecewise linear function π defined by

$$\pi(y) = \begin{cases} 0 \ , & \text{if } y \leq 0 \\ y \ , & \text{if } 0 \leq y \leq 1 \\ 1 \ , & \text{if } y \geq 1 \end{cases}$$

([L] refers to a gate g with $\gamma^g = \pi$ as a "threshold logic element"). On the other hand there exist results which suggest that such upper bound would be non-trivial. It has already been shown in [MSS] that constant size neural nets of depth 2 with activation function π and small integer weights can compute *more* boolean functions than constant size neural nets of depth 2 with linear threshold gates (and arbitrary weights). [DS] exhibits an even stronger increase in computational power for the case of quadratic activation functions.

Hence even *simple* non-boolean activation functions provide more computational power to a neural net than the heaviside-function. However it has been open by *how much* they can increase the computational power (in the presence of arbitrary weights). E. Sontag has pointed out that known methods do not even suffice to show for a constant depth neural net $\mathcal{N}_n$ of size $O(n^{O(1)})$ with n inputs and activation function π, that there is *any* boolean function $F_n : \{0,1\}^n \rightarrow \{0,1\}$ that can *not* be computed on $\mathcal{N}_n$ with a suitable weight-assignment. Correspondingly no better upper bound than the trivial 2^n could be given for the VC-dimension of such $\mathcal{N}_n$ (with n boolean inputs). From the technical point of view, this inability was caused by the lack of an upper bound on the amount of information that can be encoded in such neural net by the assignment of weights. For the case of neural nets with heaviside gates this upper bound on the information-capacity of weights is provided by the quoted result of Muroga et. al. [Mu]. However this problem is substantially more difficult for neural nets with piecewise linear activation functions. For this model it is no longer sufficient to analyze a single gate with boolean inputs and outputs. Even if the inputs and outputs of the neural net are boolean valued, the "signals" that are transmitted between the hidden units are real valued. Furthermore one can give no a-priori bound on the precision required for such analog signals between hidden units, since one has no control over the maximal size of weights in the neural net. Obviously a large weight will magnify any imprecision. Note also that a computation on a multi-layer neural net of the here considered type involves *products* of weights from subsequent levels. Hence, if some of the weights are arbitrarily large, one needs arbitrarily high precision for the other weights.

The main technical contribution of this chapter are two new methods for reducing nonlinear problems about weights in multi-layer neural nets to linear problems for a transformed set of parameters. These two methods are presented in the sections 2 and 3 of this chapter. We introduce in section 2 of this chapter a method that allows us to prove an upper bound for the information-capacity of weights for neural nets with piecewise linear activation functions (hence in particular for π). It is shown that for the computation of boolean functions on neural nets $\mathcal{N}_n$ of constant depth and polynomially in n many gates (where n is the number of input variables) it is sufficient to use as weights rational numbers with polynomially in n many bits. As a consequence one can simulate any such analog neural net by a digital neural net of constant depth and polynomial size with the heaviside activation function (i.e. linear threshold gates) and binary weights (i.e. weights from $\{0,1\}$). This result also implies that the VC-dimension of $\mathcal{N}_n$ can be bounded above by a polynomial in n.

In section 3 we introduce another proof-technique, that allows us to derive the same two consequences also for neural nets with piecewise *polynomial* activation functions and nonlinear gate-inputs $Q^g(y_1, \ldots, y_m)$ of bounded degree. These results show that in spite of the previously quoted evidence for the superiority of non-boolean activation functions in neural nets, there is some limit to their computational power as long as the activation functions are piecewise polynomial. On the other hand the polynomial upper bound on the VC-dimension of such neural nets may be interpreted as *good news*: It shows that neural nets of this type can in principle be trained with a sequence of examples that is not too long.

The "linearization" of the requirements on the weights that is carried out in sections 2 and 3 has also implications for PAC-learning on analog neural nets (see [M 93c], or Theorem 4.7 in our later chapter on learning in this volume).

The results of this chapter were first announced in [M 92], and an extended abstract of these results appeared in [M 93a]. Another result of [M 93a], the construction of neural nets whose VC-dimension is superlinear in the number of weights, has subsequently been improved to apply also for depth 3. A full version of that proof appears in [M 93b].

Definition 1.1 *A network architecture (or "neural net") $\mathcal{N}$ of order k is a labelled acyclic directed graph $\langle V, E\rangle$. Its nodes of fan-in 0 are labelled by the input variables $x_1, \ldots, x_n$. Each node g of fan-in $m > 0$ is called a computation node (or gate), and is labelled by some activation function $\gamma^g : \mathbf{R} \to \mathbf{R}$ and some polynomial $Q^g(y_1, \ldots, y_m)$ of degree $\leq k$. Furthermore $\mathcal{N}$ has a unique node of fan-out 0, which is called the output node of $\mathcal{N}$ and which carries as an additional label a certain real number T_{out} (called "the outer threshold of $\mathcal{N}$").*

The coefficients of all polynomials $Q^g(y_1, \ldots, y_m)$ for gates g in $\mathcal{N}$ and the outer threshold T_{out} are called the programmable parameters of $\mathcal{N}$. Assume that $\mathcal{N}$ has w programmable parameters, and that some numbering of these has been fixed. Then each assignment $\underline{\alpha} \in \mathbf{R}^w$ of reals to the programmable parameters in $\mathcal{N}$ defines an analog circuit $\mathcal{N}^{\underline{\alpha}}$, which computes a function $\underline{x} \mapsto \mathcal{N}^{\underline{\alpha}}(\underline{x})$ from $\mathbf{R}^n$ into $\{0,1\}$ in the following way: Assume that some input $\underline{x} \in \mathbf{R}^n$ has been assigned to the input nodes of $\mathcal{N}$. If a gate g in $\mathcal{N}$ has m immediate predecessors in $\langle V, E\rangle$ which output $y_1, \ldots, y_m \in \mathbf{R}$, then g outputs $\gamma^g(Q^g(y_1, \ldots, y_m))$. Finally, if g_{out} is the output gate of $\mathcal{N}$ and g_{out} gives the real valued output z (according to the preceding inductive definition) we define

$$\mathcal{N}^{\underline{\alpha}}(\underline{x}) := \begin{cases} 1 \ , & \text{if } z \geq T_{out} \\ 0 \ , & \text{if } z < T_{out}, \end{cases}$$

where T_{out} is the outer threshold that has been assigned by $\underline{\alpha}$ to g_{out}.

Any parameters that occur in the definitions of the activation functions γ^g of $\mathcal{N}$ are referred to as architectural parameters of $\mathcal{N}$.

Definition 1.2 *A function $\gamma : \mathbf{R} \to \mathbf{R}$ is called piecewise polynomial if there are thresholds $t_1, \ldots, t_k \in \mathbf{R}$ and polynomials $P_0, \ldots, P_k$ such that $t_1 < \ldots < t_k$ and for each $i \in \{0, \ldots, k\} : t_i \leq x < t_{i+1} \Rightarrow \gamma(x) = P_i(x)$ (we set $t_0 := -\infty$ and $t_{k+1} := \infty$).*

If k is chosen minimal for γ, we refer to k as the number of polynomial pieces of γ, to $P_0, \ldots, P_k$ as the polynomial pieces of γ, and to $t_1, \ldots, t_k$ as the thresholds of γ. Furthermore we refer to $t_1, \ldots, t_k$ together with all coefficients in the polynomials $P_0, \ldots, P_k$ as the parameters of γ. The maximal degree of $P_0, \ldots, P_k$ is called the degree of γ. If the degree of γ is ≤ 1 then we call γ piecewise linear, and we refer to $P_0, \ldots, P_k$ as the linear pieces of γ.

If γ occurs as activation function γ^g of some network architecture $\mathcal{N}$, then one refers to the parameters of γ as architectural parameters of $\mathcal{N}$.

Note that we do not require that γ is continuous (or monotone).

Definition 1.3 *Assume that $\mathcal{N}$ is an arbitrary network architecture with n inputs and w programmable parameters, and $S \subseteq \mathbf{R}^n$ is an arbitrary set. Then one defines the VC-dimension of $\mathcal{N}$ over S in the following way:*

$$\begin{aligned} \textit{VC-dimension}(\mathcal{N}, S) \ := \ & \textit{max}\{|S'| \, \big| S' \subseteq S \textit{ has the property that for every function} \\ & F : S' \to \{0,1\} \textit{ there exists a parameter assignment} \\ & \underline{\alpha} \in \mathbf{R}^w \textit{ such that } \forall \, \underline{x} \in S'(\mathcal{N}^{\underline{\alpha}}(\underline{x}) = F(\underline{x}))\}. \end{aligned}$$

Remark 1.4 "VC-dimension" is an abbreviation for "Vapnik-Chervonenkis dimension". It has been shown in [BEHW] (see also [BH], [A]) that the VC-dimension of a neural net $\mathcal{N}$ essentially determines the number of examples that are needed to train $\mathcal{N}$ (in Valiant's model for probably approximately correct learning [V]). Sontag [S2] has shown that the VC-dimension of a neural net can be drastically increased by using activation functions with non-boolean output instead of the heaviside function sgn. We refer to our later chapter in this volume about learning on neural nets for further results about the VC-dimension of neural nets.

2 A Bound for the Information - Capacity of Weights in Neural Nets with Piecewise Linear Activation Functions

We consider for arbitrary $a \in \mathbf{N}$ the following set of rationals with up to a bits before and after the comma:

$$\mathbf{Q}_a := \Big\{ r \in \mathbf{Q} \ \Big| \quad r = s \cdot \sum_{i=-a}^{a-1} b_i \cdot 2^i \quad \text{for } b_i \in \{0,1\}, \ i = -a, \ldots, a-1 \text{ and } s \in \{-1, 1\} \Big\}.$$

Note that for any $r \in \mathbf{Q}_a : \ |r| \le 2^a \le 2^{2a} \cdot \min\{|r'| \mid r' \in \mathbf{Q}_a \text{ and } r' \neq 0\}$.

Theorem 2.1 *Consider an arbitrary network architecture $\mathcal{N}$ of order 1 over a graph $\langle V, E\rangle$ with n input nodes, in which every computation node has fan-out ≤ 1. Assume that each activation function γ^g in $\mathcal{N}$ is piecewise linear with parameters from $\mathbf{Q}_a$. Let $w := |V| + |E| + 1$ be the number of programmable parameters in $\mathcal{N}$.*

Then for every $\underline{\alpha} \in \mathbf{R}^w$ there exists a vector $\underline{\alpha}' = < \frac{s_1}{t}, \ldots, \frac{s_w}{t} > \in \mathbf{Q}^w$ with integers $s_1, \ldots, s_w, t$ of absolute value $\le (2w+1)! \, 2^{2a(2w+1)}$ such that $\forall \underline{x} \in \mathbf{Q}_a^n \big(\mathcal{N}^{\underline{\alpha}}(\underline{x}) = \mathcal{N}^{\underline{\alpha}'}(\underline{x})\big)$. In particular $\mathcal{N}^{\underline{\alpha}'}$ computes the same boolean function as $\mathcal{N}^{\underline{\alpha}}$.

Remark 2.2 The condition of Theorem 2.1 that all computation nodes in $\mathcal{N}$ have fan-out ≤ 1 is automatically satisfied for $d \le 2$. For larger d one can simulate any network architecture $\mathcal{N}$ of depth d with s nodes by a network architecture $\mathcal{N}'$ with $\leq \frac{s}{s-1} \cdot s^{d-1} \leq \frac{3}{2}s^{d-1}$ nodes and depth d that satisfies this condition. Hence this condition is not too restrictive for network architectures of a constant depth d.

It should also be pointed out that there is in the assumption of Theorem 2.1 no explicit bound on the number of linear pieces of γ^g (apart from the requirement that its thresholds are from $\mathbf{Q}_a$). For example these activation functions may consist of 2^a linear pieces (with discontinuous jumps in between). Furthermore γ^g is not required to be monotone.

Finally it should be mentioned that a corresponding version of Theorem 2.1 also holds for rational numbers that do not have a finite binary representation, i.e. for all rationals from $\mathbf{Q}'_a := \{r \in \mathbf{Q} : r \text{ is the quotient of integers of bit-length} \leq a\}$ instead of $\mathbf{Q}_a$.

Remark 2.3 Previously one had *no* upper bound for the computational power (or for the VC-dimension) of multi-layer neural nets $\mathcal{N}$ with arbitrary weights and analog computational elements (i.e. activation functions with non-boolean output). Theorem 2.1 implies that any $\mathcal{N}$ of the considered type can compute with the help of arbitrary parameter assignments $\underline{\alpha} \in \mathbf{R}^w$ at most $2^{O(aw^2 \log w)}$ different functions from $\mathbf{Q}_a^n$ into $\{0,1\}$, hence VC-dimension $(\mathcal{N}, \mathbf{Q}_a^n) = O(w^2(a + \log w))$.

Furthermore Theorem 2.1 implies that one can *replace* all *analog computations inside* $\mathcal{N}$ by *digital arithmetical operations on not too large integers* (the proof gives an upper bound of $O(wa + w \log w)$ for their bit-length). It is well-known that each of these digital arithmetical operations (multiple addition, multiplication, division) can be carried out on a circuit of small constant depth with $O(a^{O(1)} \cdot w^{O(1)})$ MAJORITY-gates, hence also on a network architecture of depth $O(1)$ and size $O(a^{O(1)} \cdot w^{O(1)})$ with linear threshold gates and weights from $\{-1, 0, 1\}$ ([CSV], [PS], [HMPST], [GHR], [SR], [SBKH]). Thus one can simulate for inputs from $\{0,1\}^n$ any depth d network architecture $\mathcal{N}$ as in Theorem 2.1 with arbitrary parameter assignments $\underline{\alpha} \in \mathbf{R}^w$ by a network architecture of depth $O(d)$ and size $O(a^{O(1)} \cdot w^{O(1)})$ with linear threshold gates and weights from $\{-1, 0, 1\}$. The same holds for inputs from $\mathbf{Q}_a^n$ if they are given to $\mathcal{N}$ in digital form.

Proof of Theorem 2.1: In the special case where $\gamma^g = \text{sgn}$ for all gates in $\mathcal{N}$ this result is well known ([Mu]). It follows by applying separately to each gate in $\mathcal{N}$ the following result.

Lemma 2.4 (folklore; see [MT] for a proof) *Consider a system $A\underline{x} \leq \underline{b}$ of some arbitrary finite number of linear inequalities in l variables. Assume that all entries in A and $\underline{b}$ are integers of absolute value $\leq K$.*

If this system has any solution in $\mathbf{R}^l$, then it has a solution of the form $\langle \frac{s_1}{t}, \ldots \frac{s_l}{t} \rangle$, where $s_1, \ldots, s_l, t$ are integers of absolute value $\leq (2l+1)!\, K^{2l+1}$.

Sketch of the proof for Lemma 2.4: Let k be the number of inequalities in $A\underline{x} \leq \underline{b}$. One writes each variable in $\underline{x}$ as a difference of 2 nonnegative variables, and one adds to each inequality a "slack variable". In this way one gets an equivalent system

$$(1) \qquad A'\underline{x}' = \underline{b} \quad , \quad \underline{x}' \geq \underline{0}$$

over $l' := 2l + k$ variables, for some $k \times l'$ matrix A'. The k columns of A' for the k slack-variables in $\underline{x}'$ form an identity matrix. Hence A' has rank k.

The assumption of the Lemma implies that (1) has a solution over $\mathbf{R}$. Hence by Caratheodory's Theorem (Corollary 7.1i in [Sch]) one can conclude that there is also a solution over $\mathbf{R}$ of a system

$$(2) \qquad A''\underline{x}'' = \underline{b} \quad , \quad \underline{x}'' \geq \underline{0}$$

where A'' consists of k linearly independent columns of A'. Since A'' has full rank, (2) has in fact a unique solution that is given by Cramer's rule: $x''_j = \det(A''_j)/\det A''$ for $j = 1, \ldots, k$, where A''_j results form A'' by replacing its j^{th} column by $\underline{b}$. Since all except up to $2l$ columns of A'' contain exactly one 1 and else only 0's, we can bring each of the matrices A'', A''_j by permutations of rows and columns into a form

$$B = \begin{pmatrix} C & 0 \\ D & I \end{pmatrix}$$

where C is a square matrix with $2l + 1$ rows. Hence the determinant of B is an integer of absolute value $\leq (2l + 1)!\, K^{2l+1}$. ∎

The difficulty of the proof of Theorem 2.1 lies in the fact that with *analog* computational elements one can no longer treat each gate separately, since intermediate values are no longer integers. Furthermore the total computation of $\mathcal{N}$ can in general *not* be described by a system of *linear* inequalities, where the w variable parameters of $\mathcal{N}$ are the variables in the inequalities (and the fixed parameters of $\mathcal{N}$ are the constants). This becomes obvious if one just considers the composition of two very simple analog gates g_1 and g_2 on levels 1 and 2 of $\mathcal{N}$, whose activation functions γ_1, γ_2 satisfy $\gamma_1(y) = \gamma_2(y) = y$. Assume $x = \sum_{i=1}^{n} \alpha_i x_i + \alpha_0$ is the input to gate g_1, and g_2 receives as input $\sum_{j=1}^{m} \alpha'_j y_j + \alpha'_0$ where $y_1 = \gamma_1(x) = x$ is the output of gate g_1. Then g_2 outputs $\alpha'_1 \cdot \left(\sum_{i=1}^{n} \alpha_i x_i + \alpha_0\right) + \sum_{j=2}^{m} \alpha'_j y_j + \alpha'_0$. Obviously this term is not linear in the weights $\alpha'_1, \alpha_1, \ldots, \alpha_n$. Hence if the output of gate g_2 is compared with a fixed threshold at the next gate, the resulting inequality is not linear in the weights of the gates in $\mathcal{N}$.

If the activation functions of all gates in $\mathcal{N}$ were linear (as in the example for g_1 and g_2), then there would be no problem because a composition of linear functions is linear. However for *piecewise* linear activation functions it is not sufficient to consider their composition, since intermediate results have to be compared with boundaries between linear pieces of the next gate.

We introduce in this chapter a new method in order to handle this difficulty. We simulate $\mathcal{N}^{\underline{\alpha}}$ by another neural net $\hat{\mathcal{N}}[\underline{c}]^{\underline{\beta}}$ (which one may view as a "normal form" for $\mathcal{N}^{\underline{\alpha}}$) that uses the same graph $\langle V, E\rangle$ as $\mathcal{N}$, but different activation functions and different values $\underline{\beta}$ for its variable parameters. The activation functions of $\hat{\mathcal{N}}[\underline{c}]$ depend on $|V|$ new parameters $\underline{c} \in \mathbf{R}^{|V|}$, which we call *scaling parameters* in the following. Although this new neural net has the *disadvantage* that it requires $|V|$ additional parameters $\underline{c}$, it has the *advantage* that we can choose in $\hat{\mathcal{N}}[\underline{c}]$ all weights on edges between computation nodes to be from $\{-1, 0, 1\}$. Since these weights from $\{-1, 0, 1\}$ are already of the desired bit-length, we can treat them as constants in the system of inequalities that describes computations of $\hat{\mathcal{N}}[\underline{c}]$. Thereby we can achieve that all variables that appear in the inqualities that describe computations of $\hat{\mathcal{N}}[\underline{c}]$ (the variables for weights of gates on level 1, the variables for the biases of gates on all levels, the variable for the outer threshold, *and the new variables for the scaling parameters* $\underline{c}$) appear only *linearly* in those inqualities. Hence we can apply Lemma 2.4 to the system of inequalities that describes the computations of $\hat{\mathcal{N}}$ for inputs from $\mathbf{Q}_a^n$, and thereby get a "nice" solution $\underline{\beta}', \underline{c}'$ for all variable parameters in $\hat{\mathcal{N}}$. Finally we observe that we can transform $\hat{\mathcal{N}}[\underline{c}']^{\underline{\beta}'}$ back into the original neural net $\mathcal{N}$ with an assignment of small "numbers" $\underline{\alpha}'$ to all variable parameters in $\mathcal{N}$.

We will now fill in some of the missing details. Consider the gate function γ of an arbitrary gate g in $\mathcal{N}$. Since γ is piecewise linear, there are fixed parameters $t_1 < \cdots < t_k$, $a_0, \ldots, a_k$, $b_0, \ldots, b_k$ in $\mathbf{Q}_a$ (which may be different for different gates g) such that with $t_0 := -\infty$ and $t_{k+1} := +\infty$ one has $\gamma(x) = a_i x + b_i$ for $x \in \mathbf{R}$ with $t_i \leq x < t_{i+1}$; $i = 0, \ldots, k$. For an arbitrary scaling parameter $c \in \mathbf{R}^+$ we associate with γ the following piecewise linear activation function γ^c: the thresholds of γ^c are $c \cdot t_1, \cdots, c \cdot t_k$ and its output is $\gamma^c(x) = a_i x + c \cdot b_i$ for $x \in \mathbf{R}$ with $c \cdot t_i \leq x < c \cdot t_{i+1}$; $i = 0, \ldots, k$ (set $c \cdot t_0 := -\infty$, $c \cdot t_{k+1} := +\infty$). Thus for all reals $c > 0$ the function γ^c is related to γ through the equality: $\forall x \in \mathbf{R}\ (\gamma^c(c \cdot x) = c \cdot \gamma(x))$.

Assume that $\underline{\alpha} \in \mathbf{R}^w$ is some arbitrary given assignment to the variable parameters in $\mathcal{N}$. We transform $\mathcal{N}^{\underline{\alpha}}$ into a "normal form" $\hat{\mathcal{N}}[\underline{c}]^{\underline{\beta}}$ in which all weights on edges between computation nodes are from $\{-1, 0, 1\}$, such that $\forall \underline{x} \in \mathbf{R}^n \left(\mathcal{N}^{\underline{\alpha}}(\underline{x}) = \hat{\mathcal{N}}[\underline{c}]^{\underline{\beta}}(\underline{x})\right)$. We proceed inductively from the output level towards the input level. Assume that the output gate g_{out} of $\mathcal{N}^{\underline{\alpha}}$ receives as input $\sum_{i=1}^{m} \alpha_i y_i + \alpha_0$, where $\alpha_1, \ldots, \alpha_m, \alpha_0$ are the weights and the bias of g_{out} (under the assignment $\underline{\alpha}$) and $y_1, \ldots, y_m$ are the (real valued) outputs of the immediate predecessors $g_1, \ldots, g_m$ of g. For each $i \in \{1, \ldots, m\}$ with $\alpha_i \neq 0$ such that g_i is not an input node we replace the activation function γ_i of g_i by $\gamma_i^{|\alpha_i|}$, and we multiply the weights and the bias of gate g_i with $|\alpha_i|$. Finally we replace the weight α_i of gate g_{out} by 1, if $\alpha_i > 0$, and by -1 if $\alpha_i < 0$.

This operation has the effect that the multiplication with $|\alpha_i|$ is carried out *before* the gate g_i (rather than after g_i, as done in $\mathcal{N}^{\underline{\alpha}}$), but that the considered output gate g_{out} still receives the same input as before. The analogous operation is then inductivily carried out for the predecessors g_i of g_{out} (note however that the weights of g_i are no longer the original ones from $\mathcal{N}^{\underline{\alpha}}$, since they have been changed in the preceding step). We exploit here the assumption that each gate has fan-out ≤ 1.

Let $\underline{\beta}$ consist of the new weights on edges adjacent to input nodes, of the resulting biases of all gates in $\hat{\mathcal{N}}$, and of the (unchanged) outer threshold T_{out}. Let $\underline{c}$ consist of the resulting scaling factors at the gates of $\mathcal{N}$. Then we have $\forall \underline{x} \in \mathbf{R}^n \left(\mathcal{N}^{\underline{\alpha}}(\underline{x}) = \hat{\mathcal{N}}[\underline{c}]^{\underline{\beta}}(\underline{x})\right)$.

Finally we have to replace all *strict* inequalities of the form "$s_1 < s_2$" that are needed to describe the computation of $\hat{\mathcal{N}}[\underline{c}]^{\underline{\beta}}$ for some input $\underline{x} \in \mathbf{Q}_a^n$ by inequalities of the form "$s_1 + 1 \leq s_2$". This concerns inequalities of the form $s < c \cdot t_i$, where $c \cdot t_i$ is the threshold of some gate g in $\hat{\mathcal{N}}[\underline{c}]$ and s is its gate input, inequalities of the form $s < T_{out}$ where s is the output of g_{out}, and inequalities of the form $0 < c$ for each scaling parameter c. In order to achieve this stronger separation it is sufficient to multiply all parameters $\underline{\beta}$, $\underline{c}$ in $\hat{\mathcal{N}}$ by a sufficiently large constant K. For simplicity we write again $\underline{\beta}$, $\underline{c}$ for the resulting parameters. We now specify a system $\mathcal{A}\underline{z} \leq \underline{b}$ of linear inequalities in w variables $\underline{z}$ that play the role of the w parameters $\underline{\beta}$, $\underline{c}$ in the computations of $\hat{\mathcal{N}}[\underline{c}]^{\underline{\beta}}$ for all inputs $\underline{x}$ from $\mathbf{Q}_a^n$. The constants of these inequalities are the coordinates of all inputs $\underline{x} \in \mathbf{Q}_a^n$, the parameters of the activation functions γ in $\mathcal{N}$, the constants $-1, 1$ that occur in $\hat{\mathcal{N}}$ as weights of edges between computation nodes, and the constants 1 that arise from the replacement of strict inequalities "$s_1 < s_2$" by "$s_1 + 1 \leq s_2$".

For each fixed input $\underline{x} \in \mathbf{Q}_a^n$ one places into the system $\mathcal{A}\underline{z} \leq \underline{b}$ up to two linear inequalities for each gate g in $\mathcal{N}$. These inequalities are defined by induction on the depth of g. If g has depth 1, $t_1 < \cdots < t_k$ are the thresholds of its activation functions γ in $\mathcal{N}$, and its input $\sum_{i=1}^{n} \alpha_i x_i + \alpha_0$ in $\hat{\mathcal{N}}[\underline{c}]^{\underline{\beta}}$ satisfies $c \cdot t_j \leq \sum_{i=1}^{n} \alpha_i x_i + \alpha_0$ and $\sum_{i=1}^{n} \alpha_i x_i + \alpha_0 + 1 \leq c \cdot t_{j+1}$, then one adds these two inequalities to the system (more precisely: if $j = 0$ or $j = k$ then only one inequality is needed since the other one is automatically true).

If g' is a successor gate of g, it receives from g for some specific $j \in \{0, \ldots, k\}$ an output of the form $a_j \cdot (\sum_{i=1}^{n} \alpha_i x_i + \alpha_0) + c \cdot b_j$ (where c is the scaling factor of gate g). Note that this term is linear, since a_j, b_j are fixed parameters of gate g'. In this way one can express for circuit input $\underline{x}$ the input $\mathrm{I}(\underline{x})$ of gate g' as a linear term in the weights, biases and scaling factors of its preceding gates (we exploit here that in $\hat{\mathcal{N}}$ the weight on the edge between g' and each predecessor gate is a fixed parameter from $\{-1, 0, 1\}$, not a variable). If this input $\mathrm{I}(\underline{x})$ satisfies in $\hat{\mathcal{N}}[\underline{c}]^{\underline{\beta}}$ the inequalities $c' \cdot t'_{j'} \leq \mathrm{I}(\underline{x})$ and $\mathrm{I}(\underline{x}) + 1 \leq c' \cdot t'_{j'+1}$ (where $t'_1 < \ldots < t'_{k'}$ are the thresholds of g' in $\mathcal{N}$, and c' is the scaling factor of g' in $\hat{\mathcal{N}}$), then one adds these two inequalities to the system $\mathcal{A}\underline{z} \leq \underline{b}$ (respectively only one if $j' = 0$ or $j' = k'$). Note that all resulting inequalities are linear, in spite of the fact that it contains variables for the biases of *all* gates. It should also be pointed out that the definition of this system of inequalities is more involved than it may first appear, since the sum of terms $I(\underline{x})$

depends on the chosen inequalities for all predecessor gates (e.g. on j in the example above). Hence a precise definition has to be similar to that of the more detailed proof of Theorem 3.1 (see the Journal version of [M 93a]).

It is clear that the resulting system $\mathcal{A}\underline{z} \leq \underline{b}$ has a solution in $\mathbf{R}^w$, since $\underline{z} := \langle \underline{\beta}, \underline{c} \rangle$ is a solution. Hence we can apply Lemma 2.4, which provides a solution $\underline{z}'$ of the form $\langle \frac{s_i}{t} \rangle_{i=1,\ldots,w}$ with integers $s_1, \ldots, s_w, t$ of absolute value $\leq (2w+1)!\ 2^{2a(2w+1)}$. Let $\hat{\mathcal{N}}[\underline{c}']^{\underline{\beta}'}$ be the neural net $\hat{\mathcal{N}}$ with this new assignment $\langle \underline{\beta}', \underline{c}' \rangle := \underline{z}'$ of "small" parameters. By definition we have $\forall \underline{x} \in \mathbf{Q}_a^n (\mathcal{N}^{\underline{\alpha}}(\underline{x}) = \hat{\mathcal{N}}[\underline{c}']^{\underline{\beta}'})$. We show that one can transform this neural net $\hat{\mathcal{N}}[\underline{c}']^{\underline{\beta}'}$ into a net $\mathcal{N}^{\underline{\beta}'}$ with the *same activation functions* as $\mathcal{N}^{\underline{\alpha}}$ but a new assignment $\underline{\alpha}'$ of "small" parameters (that can easily be computed from $\underline{\beta}', \underline{c}'$). This transformation proceeds inductively from the input level towards the output level. Consider some gate g on level 1 in $\hat{\mathcal{N}}$ that uses (for the new parameter assignment $\underline{c}'$) the scaling factor $c > 0$ for its activation function γ^c. Then we replace the weights $\alpha_1, \ldots, \alpha_n$ and bias α_0 of gate g in $\hat{\mathcal{N}}[\underline{c}']^{\underline{\beta}'}$ by $\frac{\alpha_1}{c}, \ldots, \frac{\alpha_n}{c}, \frac{\alpha_0}{c}$, and γ^c by γ. Furthermore if $r \in \{-1, 0, 1\}$ was in $\hat{\mathcal{N}}$ the weight on the edge between g and its successor gate g, we assign to this edge the weight $c \cdot r$. Note that g' receives in this way from g the same input as in $\hat{\mathcal{N}}[\underline{c}']^{\underline{\beta}'}$ (for every circuit input). Assume now that $\alpha_1', \ldots, \alpha_m'$ are the weights that the incoming edges of g' get assigned in this way, that α_0' is the bias of g' in the assignment $\underline{z}' = \langle \underline{\beta}', \underline{c}' \rangle$, that $c' > 0$ is the scaling factor of g' in $\hat{\mathcal{N}}[\underline{c}']^{\underline{\beta}'}$. Then we assign the new weights $\frac{\alpha_1'}{c'}, \ldots, \frac{\alpha_m'}{c'}$ and the new bias $\frac{\alpha_0'}{c'}$ to g', and we multiply the weight on the outgoing edge from g' by c'.

By construction we have that $\forall \underline{x} \in \mathbf{R}^n\ (\mathcal{N}^{\underline{\alpha}'}(\underline{x}) = \hat{\mathcal{N}}[\underline{c}']^{\underline{\beta}'}(\underline{x}))$, hence
$\forall \underline{x} \in \mathbf{Q}_a^n\ \ (\mathcal{N}^{\underline{\alpha}'}(\underline{x}) = \mathcal{N}^{\underline{\alpha}}(\underline{x}))$. ∎

3 Upper Bounds for Neural Nets with Piecewise Polynomial Activation Functions

Theorem 3.1 *Consider an arbitrary array $(\mathcal{N}_n)_{n \in \mathrm{N}}$ of high order network architectures $\mathcal{N}_n$ of depth $O(1)$ with n inputs and $O(n^{O(1)})$ gates, in which the gate function γ^g of each gate g is piecewise polynomial of degree $O(1)$ with $O(n^{O(1)})$ polynomial pieces, with arbitrary reals as architectural parameters.*

Then there exists an array $(\tilde{\mathcal{N}}_n)_{n \in \mathrm{N}}$ of first order network architectures $\tilde{\mathcal{N}}_n$ of depth $O(1)$ with n inputs and $O(n^{O(1)})$ gates such that each gate g in $\tilde{\mathcal{N}}_n$ uses as its activation function the heaviside function sgn *(i.e. g is a linear threshold gate), and such that for each assignment $\underline{\alpha}$ of arbitrary reals to the programmable parameters in $\mathcal{N}_n$ there is an assignment $\underline{\tilde{\alpha}}$ of $O(n^{O(1)})$ numbers from $\{-1, 0, 1\}$ to the programmable parameters in $\tilde{\mathcal{N}}_n$ such that $\forall \underline{x} \in \{0,1\}^n\ \ (\mathcal{N}_n^{\underline{\alpha}}(\underline{x}) = \tilde{\mathcal{N}}_n^{\underline{\tilde{\alpha}}}(\underline{x}))$.*

Hence for any assignment $(\underline{\alpha}_n)_{n \in \mathrm{N}}$ of real valued parameters the boolean functions that are computed by $(\mathcal{N}_n^{\underline{\alpha}_n})_{n \in \mathrm{N}}$ are in TC^0. In particular VC-dimension

$(\mathcal{N}_n, \{0,1\}^n) = O(n^{O(1)})$.

Remark 3.2 Theorem 3.1 yields no bound for the computational power of neural nets with the activation function $\sigma(y) = 1/(1+e^{-y})$. However it provides bounds for the case where the activation functions are spline approximations to σ of arbitrarily high degree d, provided that $d \in \mathbf{N}$ is fixed.

Idea of the proof of Theorem 3.1 This proof is quite long and involved, even for the simplest nonlinear case where the activation functions consist of 2 polynomial pieces of degree 2. Note that in contrast to the model in [SS] the magnitude of the given weights in $\mathcal{N}_n$ may grow arbitrarily fast as a function of n.

We first note that one can eliminate all nonlinear polynomials $\mathbf{Q}^g$ as arguments of activation functions by introducing intermediate gates with linear gate inputs and quadratic activation functions. One exploits here the obvious fact that $y \cdot z = \frac{1}{2}\Big((y+z)^2 - y^2 - z^2\Big)$. In this way one can transform the given network architectures into *first order* network architectures which still satisfy the assumptions of Theorem 3.1.

Subsequently we transform each given neural net $\mathcal{N}_n^{\underline{\alpha}_n}$ into a normal form $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$ of constant depth and size $O(n^{O(1)})$ in which all gates g have fan-out ≤ 1, and all gates g use as activation functions γ^g piecewise polynomial functions of the following special type: γ^g consists of up to 3 pieces, of which at most one is not identically 0, and in which the nontrivial piece outputs a constant, or computes a power $y \mapsto y^k$ (where $k \in \mathbf{N}$ satisfies $k = O(1)$). We can choose $\underline{\beta}_n$ such that one has "$s_1 + 1 \leq s_2$" for all strict inequalities "$s_1 < s_2$" that arise in $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$ for inputs from $\{0,1\}^n$ when one compares some intermediate term s_1 with the threshold s_2 of some gate, or with the outer threshold (analogously as in the proof of Theorem 2.1). This transformation can be done in such a way that $\forall \underline{x} \in \{0,1\}^n (\mathcal{N}_n^{\underline{\alpha}_n}(\underline{x}) = \hat{\mathcal{N}}_n^{\underline{\beta}_n}(\underline{x}))$.

It would also be possible to push all nontrivial weights to the gates on level 1, in correspondence to the construction in the proof of Theorem 2.1. However in the present context this additional operation does not eliminate non-linear conditions on the weights. Assume for example that g is a gate on level 1 with input $\alpha_1 x_1 + \alpha_2 x_2$ and activation function $\gamma^g(y) = y^2$. Then this gate g outputs $\alpha_1^2 x_1^2 + 2\alpha_1\alpha_2 x_1 x_2 + \alpha_2^2 x_2^2$. Hence the variables α_1, α_2 will not occur linearly in an inequality which describes the comparison of the output of g with some threshold of a gate at the next level.

Although it does not eliminate non-linear conditions on the weights if one pushes all weights towards level 1, the resulting network provides some notational advantage because all weights between computation nodes can be treated as constants (with three possible values). Therefore this approach has been chosen in [M 92] and [M 93a]. However this approach is disadvantageous if one wants to apply the method of this proof in the context of agnostic PAC-learning on analog neural nets ([M 93c]).

In this application one has to be able to control the bit-length of the (rational) weights. Therefore one cannot afford to push all weights towards level 1, since this may increase the bit-length of weights in an unbounded manner. For example if one pushes the weight 2 through a gate g with activation function $\gamma^g(y) = y^2$, then this weight is changed to $\sqrt{2}$ (since $2\gamma^g(y) = \gamma^g(\sqrt{2} \cdot y)$).

Since the non-linearity of the conditions on the weights cannot be eliminated in the same way as for Theorem 2.1, we have to introduce an alternative method. We fix an arbitrary assignment $\underline{\beta}_n$ of real numbers to the parameters of $\hat{\mathcal{N}}_n$. We introduce for the system of inequalities $L(\hat{\mathcal{N}}_n^{\underline{\beta}_n}, \{0,1\}^n)$ (that describes the computations of $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$ for all inputs $\underline{x} \in \{0,1\}^n$) new variables v for all nontrivial parameters in $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$ (i.e. for the weights and bias of each gate g, for the outer threshold T_{out} *and for the thresholds* t_1^g, t_2^g *of each gate* g). In addition we introduce new variables for all *products* of such parameters that arise in the computation of $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$. We have to keep the inequalities linear in order to apply Lemma 2.4. Hence we cannot demand in these inequalities that the value of the variable $v_{v_1^g, v_2^g}$ (that represents the product of α_1^g and α_2^g) is the product of the values of the variables v_1^g and v_2^g (that represent the weights α_1^g respectively α_2^g). We solve this problem by describing in detail in the linear inequalities $L(\hat{\mathcal{N}}_n^{\underline{\beta}_n}, \{0,1\}^n)$ which *role* the product of α_1^g and α_2^g plays in the computations of $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$ for inputs from $\{0,1\}^n$. It turns out that this can be done in such a way that *it does not matter* whether a solution $\mathcal{A}$ of $L(\hat{\mathcal{N}}_n^{\underline{\beta}_n}, \{0,1\}^n)$ assigns to the variable $v_{v_1^g, v_2^g}$ a value $\mathcal{A}(v_{v_1^g, v_2^g})$ that is equal to the product of the values $\mathcal{A}(v_1^g)$ and $\mathcal{A}(v_2^g)$ (that are assigned by $\mathcal{A}$ to the variables v_1^g and v_2^g). In any case $\mathcal{A}(v_{v_1^g, v_2^g})$ is forced to *behave like* the product of $\mathcal{A}(v_1^g)$ and $\mathcal{A}(v_2^g)$ in the computations of $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$.

We would like to emphasize that the parameters $\underline{\beta}_n$ do *not* occur as constants in the system $L(\hat{\mathcal{N}}_n^{\underline{\beta}_n}, \{0,1\}^n)$ of inequalities. They are also replaced by variables. The reason why the real valued parameters $\underline{\beta}_n$ occur nevertheless in our notation $L(\hat{\mathcal{N}}_n^{\underline{\beta}_n}, \{0,1\}^n)$ of inequalities is the following. These inequalities consist of conditions which demand that for any input $\underline{x} \in \{0,1\}^n$ the computation on the neural net proceeds exactly as for the parameter assignment $\underline{\beta}_n$ (i.e. the same inequalities with thresholds of the piecewise polynomial activation functions are satisfied and the same pieces of the activation functions are used at each gate as in the computation with parameter assignment $\underline{\beta}_n$).

In more abstract terms, one may view any solution $\mathcal{A}$ of $L(\hat{\mathcal{N}}_n^{\underline{\beta}_n}, \{0,1\}^n)$ as a model of a certain "linear fragment" $L(\hat{\mathcal{N}}_n^{\underline{\beta}_n}, \{0,1\}^n)$ of the theory of the role of the parameters $\underline{\beta}_n$ in the computations of $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$ on inputs from $\{0,1\}^n$. Such model $\mathcal{A}$ (which will be given by Lemma 2.4) is some type of "nonstandard model" of the theory of computations of $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$, since it replaces products of weights by "nonstandard products". Such nonstandard model $\mathcal{A}$ does not provide a new assignment of (small) weights to the network architecture $\hat{\mathcal{N}}_n$, only to a "nonstandard version" $\mathcal{M}_n^{\mathcal{A}}$ of the neural net $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$. However the linear fragment $L(\hat{\mathcal{N}}_n^{\underline{\beta}_n}, \{0,1\}^n)$ can be chosen in such a way that $\mathcal{M}_n^{\mathcal{A}}$ computes the same boolean function as $\hat{\mathcal{N}}_n^{\underline{\beta}_n}$. Furthermore, if $\mathcal{A}$ consists of a solution with "small" values as given by Lemma 2.4, then $\mathcal{M}_n^{\mathcal{A}}$ can be simulated by a constant-depth polynomial-size boolean circuit whose gates g are all MAJORITY-gates (i.e. $g(y_1, \ldots, y_m) = 1$ if $\sum\limits_{i=1}^{m} y_i \geq m/2$, otherwise $g(y_1, \ldots, y_m) = 0$). This implies that the boolean functions that are computed by $(\mathcal{M}_n^{\mathcal{A}})_{n \in \mathbf{N}}$ are in TC^0. However by construction these are the same boolean functions that are computed by $(\mathcal{N}_n^{\underline{\alpha}_n})_{n \in \mathbf{N}}$.

Further details of this proof can be found in the forthcoming journal-version of [M 93a]. ∎

4 Concluding Remarks

It is shown in this chapter that high order feedforward neural nets of constant depth with piecewise polynomial activation functions and arbitrary real weights can be simulated for boolean inputs and outputs by neural nets of a somewhat larger size and depth with heaviside gates and weights from $\{-1, 0, 1\}$. This provides the first known upper bound for the computational power of the former type of neural nets. It is also shown that in the case of first order nets with piecewise linear activation functions one can replace arbitrary real weights by rational numbers with polynomially many bits, without changing the boolean function that is computed by the neural net. In order to prove these results we have introduced two new methods for reducing nonlinear problems about weights in multi-layer neural nets to linear problems for a transformed set of parameters. These transformed parameters can be interpreted as weights in a somewhat larger neural net.

As another application of this proof technique one can show a positive result for PAC-learning (even for agnostic PAC-learning of real-valued functions) on analog neural nets of bounded size (see [M 93a], and Theorem 4.7 in our subsequent chapter on learning in this volume).

Acknowledgements

We would like to thank Eduardo D. Sontag for drawing our attention to the problem of finding upper bounds for neural nets with π-gates, and for his insightful comments. We thank Peter Auer, Franz Aurenhammer, Eric Baum, David Haussler, Philip M. Long, György Turan, and Gerhard Wöginger for various helpful discussions on this research.

References

[A] Y. S. Abu-Mostafa, "The Vapnik-Chervonenkis dimension: information versus complexity in learning", *Neural Computation*, vol. 1, 1989, 312 - 317

[B] P. L. Bartlett, "Lower bounds on the Vapnik-Chervonenkis dimension of multi-layer threshold networks", *Proc. of the 5th Annual ACM Conference on Computational Learning Theory*, 1993, ACM-Press, 144 - 150

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

[BR] A. Blum, R. L. Rivest, "Training a 3-node neural network is NP-complete", *Proc. of the 1988 Workshop on Computational Learning Theory,* Morgan Kaufmann (San Mateo, 1988), 9 - 18

[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

[CSV] A. K. Chandra, L. Stockmeyer, U. Vishkin, "Constant depth reducibility", *SIAM J. Computing*, vol. 13 (2), 1984, 423 - 439

[DS] B. DasGupta, G. Schnitger, "The power of approximating: a comparison of activation functions", in: *Advances in Neural Information Processing Systems*, vol. 5, Morgan Kaufmann (1993), 615 - 622

[DR] R. Durbin, D. E. Rumelhart, "Product units: a computationally powerful and biologically plausible extension to backpropagation networks", *Neural Computation*, vol. 1, 1989, 133 - 142

[GHR] M. Goldmann, J. Hastad, A. Razborov, "Majority gates vs. general weighted threshold gates", *Proc. of the* 7th *Structure in Complexity Theory Conference*, 1992, 2 - 13

[HMPST] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy and G. Turan, "Threshold circuits of bounded depth", *Proc. of the 28th Annual IEEE Symp. on Foundations of Computer Science*, 1987, 99 - 110. Full version in *J. Comp. System Sci.*, vol. 46, 1993, 129 - 154

[Has] J. Hastad, "On the size of weights for threshold gates", *preprint* (September 1992)

[H] D. Haussler, "Decision theoretic generalizations of the PAC model for neural nets and other learning applications", *Information and Computation*, vol. 100, 1992, 78 - 150

[Ho] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", *Proc. Nat. Acad. of Sciences USA*, 1984, 3088 - 3092

[J] D. S. Johnson, "A catalog of complexity classes", in: *Handbook of Theoretical Computer Science* vol. A, J. van Leeuwen ed., *MIT Press* (Cambridge, 1990)

[KV] M. Kearns, L. Valiant, "Cryptographic limitations on learning boolean formulae and finite automata", *Proc. of the 21st ACM Symposium on Theory of Computing*, 1989, 433 - 444

[L] R. P. Lippmann, "An introduction to computing with neural nets", *IEEE ASSP Magazine*, 1987, 4 - 22

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

[M 93b] W. Maass, "Neural nets with superlinear VC-dimension", *IIG-Report 366 of the Technische Universität Graz,* (June 1993); to appear in *Neural Computation*

[M 93c] W. Maass, "Agnostic PAC-learning of functions on analog neural nets", *IIG-Report 362 of the Technische Universität Graz,* (May 1993); to appear in *Neural Computation*

[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

[MT] W. Maass, G. Turan, "How fast can a threshold gate learn?", in: *Computational Learning Theory and Natural Learning Systems: Constraints and Prospects*, G. Drastal, S. J. Hanson and R. Rivest eds., *MIT Press*, to appear

[MR] J. L. McClelland, D. E. Rumelhart "Parallel Distributed Processing", vol. 2, *MIT Press* (Cambridge, 1986)

[Me] N. Megiddo, "Linear Programming in linear time when the dimension is fixed", *J. of the ACM*, vol. 31, 1984, 114 - 127

[MP] M. Minsky, S. Papert, "Perceptrons: An Introduction to Computational Geometry", Expanded Edition, *MIT Press* (Cambridge, 1988)

[MD] J. Moody, C. J. Darken, "Fast learning in networks of locally-tuned processing units", *Neural Computation*, vol. 1, 1989, 281 - 294

[Mu] S. Muroga, "Threshold Logic and its Applications", *Wiley* (New York, 1971)

[Ni] N. J. Nilsson, Learning Machines, McGraw-Hill (New York, 1971)

[PS] I. Parberry, G. Schnitger, “Parallel computation with threshold functions”, *Lecture Notes in Computer Science* vol. 223, Springer (Berlin, 1986), 272 - 290

[PG] T. Poggio, F. Girosi, “Networks for approximation and learning”, *Proc. of the IEEE,* vol. 78(9), 1990, 1481 - 1497

[R] F. Rosenblatt, “Principles of Neurodynamics”, *Spartan Books* (New York, 1988)

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

[Sch] A. Schrijver, “Theory of Linear and Integer Programming”, *Wiley* (New York, 1986)

[SS] H. T. Siegelmann, E. D. Sontag, “Neural networks with real weights: analog computational complexity”, *Report SYCON*-92-05, Rutgers Center for Systems and Control (Oct. 1992)

[SBKH] K. Y. Siu, J. Bruck, T. Kailath, T. Hofmeister, “Depth efficient neural networks for division and related problems”, to appear in *IEEE Transactions on Inf. Theory*

[SR] K. Y. Siu, V. Roychowdhury, “On optimal depth threshold circuits for multiplication and related problems”, *Tech. Report* ECE - 92-05, University of California, Irvine (March 1992)

[S1] E. D. Sontag, “Remarks on interpolation and recognition using neural nets”, in: *Advances in Neural Information Processing Systems* 3, R. P. Lippmann, J. Moody, D. S. Touretzky, eds., Morgan Kaufmann (San Mateo, 1991), 939 - 945

[S2] E. D. Sontag, “Feedforward nets for interpolation and classification”, *J. Comp. Syst. Sci.*, vol. 45, 1992, 20 - 48