In: *Neural Computation* **14**(2), pages 241–301, 2002.

On the Complexity of Computing and Learning with Multiplicative Neural Networks

Michael Schmitt

Lehrstuhl Mathematik und Informatik, Fakultät für Mathematik
Ruhr-Universität Bochum, D–44780 Bochum, Germany
`http://www.ruhr-uni-bochum.de/lmi/mschmitt/`
`mschmitt@lmi.ruhr-uni-bochum.de`

Abstract

In a great variety of neuron models neural inputs are combined using the summing operation. We introduce the concept of multiplicative neural networks that contain units which multiply their inputs instead of summing them and, thus, allow inputs to interact nonlinearly. The class of multiplicative neural networks comprises such widely known and well studied network types as higher-order networks and product unit networks.

We investigate the complexity of computing and learning for multiplicative neural networks. In particular, we derive upper and lower bounds on the Vapnik-Chervonenkis (VC) dimension and the pseudo dimension for various types of networks with multiplicative units. As the most general case, we consider feedforward networks consisting of product and sigmoidal units, showing that their pseudo dimension is bounded from above by a polynomial with the same order of magnitude as the currently best known bound for purely sigmoidal networks. Moreover, we show that this bound holds even in the case when the unit type, product or sigmoidal, may be learned. Crucial for these results are calculations of solution set components bounds for new network classes. As to lower bounds we construct product unit networks of fixed depth with superlinear VC dimension.

For sigmoidal networks of higher order we establish polynomial bounds that, in contrast to previous results, do not involve any restriction of the network order. We further consider various classes of higher-order units, also known as sigma-pi units, that are characterized by connectivity constraints. In terms of these we derive some asymptotically tight bounds.

Multiplication plays an important role both in neural modeling of biological behavior and in computing and learning with artificial neural networks. We briefly survey research in biology and in applications where multiplication is considered an essential computational element. The results we present here provide new tools for assessing the impact of multiplication on the computational power and the learning capabilities of neural networks.

1 Introduction

Neurons compute by receiving signals from a large number of other neurons and processing them in a complex way to yield output signals sent to other neurons again. A major issue in the formal description of single neuron computation is how the input signals interact and jointly affect the processing that takes place further. In a great many neuron models this combination of inputs is specified using a linear summation. The McCulloch-Pitts model and the sigmoidal neuron are both examples of these summing neurons which are very popular in applications of artificial neural networks. Neural network researchers widely agree that there is only a minor correspondence between these neuron models and the behavior of real biological neurons. In particular, the interaction of synaptic inputs is known to be essentially nonlinear (see, e.g., Koch, 1999). In search for biologically closer models of neural interactions, neurobiologists have found that multiplicative-like operations play an important role in single neuron computations (see also Koch and Poggio, 1992; Mel, 1994). For instance, multiplication models nonlinearities of dendritic processing and shows how complex behavior can emerge in simple networks. In recent years also evidence has accumulated that specific neurons in the nervous system of several animals compute in a multiplicative way (Andersen et al., 1985; Suga, 1990; Hatsopoulos et al., 1995; Gabbiani et al., 1999; Anzai et al., 1999a,b).

That multiplication increases the computational power and storage capacities of neural networks is well known from extensions of artificial neural networks where this operation appears in the form of higher-order units (see, e.g., Giles and Maxwell, 1987). A more general type of multiplicative neuron model is the product unit introduced by Durbin and Rumelhart (1989) where inputs are multiplied after they have been raised to some power specified by an adjustable weight. We subsume networks containing units that multiply their inputs instead of summing them under the general concept of multiplicative neural networks and investigate the impact that multiplication has on their computational and learning capabilities. A theoretical tool that quantifies the complexity of computing and learning with function classes in general, and neural networks in particular, is the Vapnik-Chervonenkis (VC) dimension. In this article we provide a theoretical study of the complexity of computing and learning with multiplicative neural networks in terms of this dimension.

The VC dimension and related notions, such as the pseudo dimension and the fat-shattering dimension, are well known to yield estimates for the number of examples required by learning algorithms for neural networks and other hypothesis classes such that training results in a low generalization error. Using these dimensions, bounds on this sample complexity of learning can be obtained not only for the model of probably approximately correct (pac) learning due to Valiant (1984) (see also Blumer et al., 1989), but also for the more general model of agnostic learning, that is, in the case when the training examples are generated by some arbitrary probability distribution (see, e.g. Haussler, 1992; Maass, 1995a,b; Anthony and Bartlett, 1999). Furthermore, in terms of the VC dimension bounds have been established also for the sample complexity of on-line learning (Maass and Turán, 1992) and Bayesian learning (Haussler et al., 1994). The VC dimension, however, is not only useful in the analysis of learning but has also proven to be a successful tool for studying the complexity of computing over, in particular, the real numbers. Koiran (1996) and Maass (1997) employed the VC dimension to establish lower bounds on the size of sigmoidal neural networks for the computation of functions. Further, using the VC dimension, limitations of the universal approximation capabilities of sigmoidal neural networks have been exhibited by the derivation of lower bounds on the size of networks that approximate continuous functions (Schmitt, 2000). Thus, in particular for neural networks the VC dimension has acquired a wide spectrum of applications in analyzing the complexity of analog computing and learning.

There are some known bounds on the VC dimension for neural networks that also partly include multiplicative units. These results are concerned with networks of higher order where the linearly weighted sum of a classical summing unit is replaced by polynomials of a certain degree. All bounds determined thus far, however, are given in terms of the maximum order of the units or require the order to be fixed. This is a severe restriction since it not only imposes a bound on the exponent of each individual input but also limits the degree of interaction among different inputs. A network with units computing piecewise polynomial functions with order at most d is known to have VC dimension $O(W^2 d)$ where W is the number of network parameters (Goldberg and Jerrum, 1995; Ben-David and Lindenbaum, 1998). For sigmoidal networks of higher-order Karpinski and Macintyre (1997) established the bound $O(W^2 k^2 \log d)$ where k is the number of network nodes. There are some further results for other network types of which we give a more complete account in a later section. They all consider the degree of higher-order units to be restricted.

In this article we derive bounds on the VC dimension and the pseudo dimension of product unit networks. In these networks the exponents are no longer fixed but treated as variable weights. In addition, no restriction is imposed on their order. We show that a feedforward network consisting of product and sigmoidal units has pseudo dimension at most $O(W^2 k^2)$ where W is the number of parameters and k the number of nodes. Hence, the same bound that is known for higher-order sigmoidal networks of restricted degree also holds when product units are used instead of monomials. Moreover, the bound is valid not only when the types of the nodes are fixed but also when they can vary between a sigmoidal unit or a product unit. This may be the case, for instance, when a learning algorithm decides which unit type to assign to which node. The results are based on the method of solution set components bounds (see Anthony and Bartlett, 1999) and we derive new such bounds. We use this method also for showing that a network with k higher-order sigmoidal units and W parameters has pseudo dimension at most $O(W^4k^2)$, a bound that also includes the exponents as adjustable parameters. Thus, the VC dimension and the pseudo dimension of these networks cannot grow indefinitely in terms of the order, and the exponents can be treated as weights analogous to the coefficients of the polynomials. These results indicate that from the theoretical viewpoint multiplication can be considered as an alternative to summation without significantly increasing the complexity of computing and learning with higher-order networks.

We further derive bounds for specific classes of higher-order units. Considering a higher-order unit as a network with one hidden layer, we define these classes in terms of constraints on the network connectivity. On the one hand, we restrict the number of connections outgoing from the input nodes, on the other hand we put a limit on the number of hidden nodes. We derive various VC dimension bounds for these classes, order-dependent as well as independent, and show for some of them that they are asymptotically tight. It might certainly be possible to embed a class of networks entirely into a single network. The results show, however, that smaller bounds arise when the specific properties of the class are taken into account. We also establish a lower bound for product unit networks stating that networks with two hidden layers of product and summing units have a superlinear VC dimension. Finally, we show that the pseudo dimension of the single product unit and of the class of monomials is equal to the number of input variables.

We focus on feedforward networks with a single output node. This means that at least two possible directions are not pursued here further: recurrent networks and networks with multiple output nodes. VC dimension bounds for recurrent networks consisting of summing units, including threshold, sigmoidal, and higher-order units with fixed degree, have been established by Koiran and Sontag (1998). Shawe-Taylor and Anthony (1991) give bounds on the sample complexity for networks of threshold units with more than one output node. Since both these works build on previous bounds for feedforward networks or single output networks, respectively, the ideas presented here may be helpful for deriving similar results for recurrent or multiple-output networks containing multiplicative units.

The article is organized as follows. In Section 2 we first introduce the necessary terminology and, in particular, demarcate multiplicative from summing units. We then report on some research in neurobiology that resulted in the use of multiplication for the modeling of biological neural systems. Further, we give a brief review of some learning applications where multiplication, mainly in the form of higher order, has been employed to increase the capabilities of artificial neural networks. In Section 3 we introduce the definitions of the VC dimension and the pseudo dimension, and exhibit some close relationships between these two combinatorial characterizations of function classes. In this section we also survey previous results where bounds on these dimensions have been obtained for neural networks. The new results are to follow in the two subsequent sections: Section 4 contains the calculations of the upper bounds, whereas lower bounds are derived in Section 5. Finally, in Section 6 we give a summary of the results in form of a table and conclude with some remarks and open questions. Proofs of some technically more involved results, which are needed in Section 4, can be found in the appendix.

2 Neural Networks with Multiplicative Units

Terminology in neural network literature varies and is occasionally used inconsistently. In the following we introduce the terms and concepts that we shall adhere to throughout this article. Further, we present some of the biological and application-specific motivations that led researchers to the use of multiplicative units in neural networks.

2.1 Neural Network Terminology

The connectivity of a neural network is given in terms of an *architecture*, which is a graph with directed edges, or *connections*, between the nodes. Nodes with no incoming edges are called *input nodes*, nodes with no outgoing edges are *output nodes*. All remaining ones are *hidden nodes*. The *computation nodes* of an architecture are its output and hidden nodes. Input nodes serve as input variables of the network. The *fan-in* of a node is the number of connections entering the node; correspondingly, its *fan-out* is the number of connections leaving it. The architecture of a *feedforward network* has no cycles. We focus on feedforward networks with one output node such as shown in Fig. 1 that are suitable for computing functions with a scalar, that is, one-dimensional, output range. Some specific architectures are said to be *layered*. In this case all nodes with equal distance from the input nodes constitute one layer and edges exist only between subsequent layers. An architecture becomes a neural network when edges and nodes are labeled with weights and thresholds, respectively, as the network parameters. To further specify which function is to be computed by the network, variables must be assigned to the input nodes and units be selected for the computation nodes. The types of units studied in this article will be defined

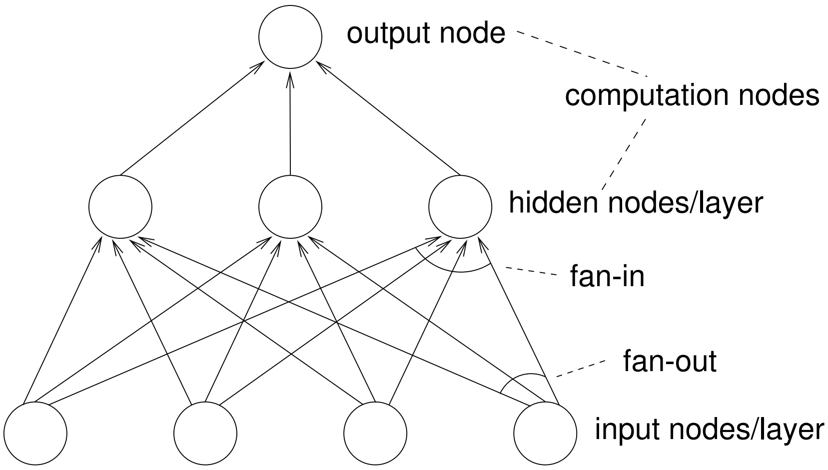

Figure 1: Neural network terminology. An architecture is shown with 4 input nodes, one hidden layer consisting of 3 hidden nodes, and one output node. Hidden and output nodes are computation nodes. All input nodes have fan-out 3, the hidden nodes have fan-in 4. We also consider architectures with fan-out restrictions. In this case subsequent layers need not be fully connected. When parameters, that is, weights and thresholds, are assigned and node functions are specified by choosing units, the architecture becomes a neural network.

in the following section. Finally, when values, that is, in general real numbers, are specified for the network parameters, the network computes a unique function defined by functional composition of its units.

2.2 Neuron Models that Sum or Multiply

The networks we are considering consist of multiplicative as well as standard summing units. First, we briefly recall the definitions of the latter.

2.2.1 Summing Units

The three most widely used summing units are the threshold unit, the sigmoidal unit and the linear unit. Each of them is parameterized by a set of weights $w_1, \ldots, w_n \in \mathbb{R}$, where n is the number of input variables, and a threshold $t \in \mathbb{R}$. They compute their output in the form

$$f(w_1x_1 + w_2x_2 + \cdots + w_nx_n - t),$$

where $x_1, \ldots, x_n$ are the input variables with domain $\mathbb{R}$ and f is a nonlinear function, referred to as the activation function of the unit. The particular choice of the activation function characterizes what kind of unit is supposed. The *threshold* *unit*, also known as McCulloch-Pitts neuron (after McCulloch and Pitts, 1943) or Perceptron[1], uses for f the sign or Heaviside function $\mathrm{sgn}(y) = 1$ if $y \geq 0$, and $\mathrm{sgn}(y) = 0$ otherwise. The logistic function $\sigma(y) = 1/(1 + e^{-y})$ is the activation function employed by the *sigmoidal unit*[2]. Finally, we have a *linear* unit if the activation function is chosen to be the identity, that is, the linear unit plainly outputs the weighted sum $w_1x_1 + w_2x_2 + \cdots + w_nx_n - t$. In other words, the linear unit computes an affine function.

The activation function also defines the output ranges of the units which are $\{0, 1\}$, $(0, 1)$, and $\mathbb{R}$ for the threshold, sigmoidal, and linear unit respectively. Linear units as computation nodes in neural networks are in general redundant if they feed their outputs only to summing units. For instance, a network consisting solely of linear units is equivalent to a single linear unit. They are mainly used as output nodes of networks that compute or approximate real-valued functions (see, e.g., the survey by Pinkus, 1999). However, linear units as hidden nodes can help to save network connections.

2.2.2 Multiplicative Units

The simplest type of a multiplicative unit is a *monomial*, that is, a product

$$x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n},$$

where $x_1, x_2, \ldots, x_n$ are input variables. Each d_i is a nonnegative integer referred to as the *degree* or *exponent* of the variable x_i. The value $d_1 + \cdots + d_n$ is called the *order* of the monomial. Since monomials restricted to the Boolean domain $\{0, 1\}$, where all non-zero exponents are 1 without loss of generality, compute the logical AND function, they are often considered as computing the continuous analog of a logical conjunction. A Boolean conjunction, however, does not necessarily require the use of multiplication because the logical AND can also be computed by a threshold unit. The same holds for the neural behavior known as shunting inhibition which is sometimes referred to as a continuous AND-NOT operation. In this article we do not consider conjunction as a multiplicative operation and use monomials as computational models for genuine multiplicative behavior with real-valued input and output.

[1]Perceptron is nowadays a widely used synonym for the threshold unit, although in the original definition by Rosenblatt (1958) the Perceptron is introduced as a network of threshold units with one hidden layer (see also Minsky and Papert, 1988).

[2]There is a large class of sigmoidal functions of which the logistic function is only one particular instance. The sigmoidal unit as it is defined here is also referred to as the *standard sigmoidal unit*.

If $M_1, \ldots, M_k$ are monomials, a *higher-order unit* is a polynomial

$$w_1 M_1 + w_2 M_2 + \cdots + w_k M_k - t$$

with real-valued weights $w_1, \ldots, w_k$ and threshold t. It is also well known under the name *sigma-pi unit* (Rumelhart et al., 1986a; Williams, 1986). As in the case of a summing unit, weights and threshold are parameters, but to uniquely identify a higher-order unit one has also to specify the structural parameter of the unit, that is, the set of monomials $\{M_1, \ldots, M_k\}$. We call this set the *structure* of a higher-order unit. A higher-order unit can be viewed as a network with one hidden layer of monomials and a linear unit as output node. For this reason, a higher-order unit is often referred to in the literature as a higher-order network. In such a network only the connections leading from hidden nodes to the output node have weights whereas the connections between input and hidden layer are weightless. Clearly, assigning multiplicative weights to input variables does not increase the power of a higher-order unit since due to the multiplicative nature of the hidden nodes all such weights can be moved forward to the output node. For higher-order units it is also common, depending on the type of application, to use a threshold or a sigmoidal unit instead of a linear unit as output node. This yields a higher-order unit that computes

$$f(w_1 M_1 + w_2 M_2 + \cdots + w_k M_k - t)$$

with nonlinear activation function f. In case that f is the sign function we have a *higher-order threshold unit*, or *polynomial threshold unit*. If the activation function is sigmoidal we refer to it as a *higher-order sigmoidal unit*.

Finally, we introduce the most general type of multiplicative unit, the *product unit*. It has the form

$$x_1^{w_1} x_2^{w_2} \cdots x_p^{w_p}$$

with variables $x_1, \ldots, x_p$ and weights $w_1, \ldots, w_p$. The number p of variables is called the *order* of the product unit. In contrast to the monomial, the product unit does not have fixed integers as exponents but variable ones that may even take on any real number. Thus, a product unit is computationally at least as powerful as a monomial. Moreover, ratios of variables, and thus division, can be expressed using negative weights. This is one reason the product unit was introduced by Durbin and Rumelhart (1989). Another advantage these and other authors explore is that the exponents of product units can be adjusted automatically, for instance, by gradient-based and other learning methods (Durbin and Rumelhart, 1989; Leerink et al., 1995a,b).

We mentioned above the well-known fact that networks of linear units are equivalent to single units. The same holds for networks consisting solely of product units. Unless there is a restriction on the order, such a network can be replaced by a single unit. Therefore, product units are mainly used in networks where they occur together with other types of units, such as threshold or sigmoidal units. For instance, the standard neural network containing product units is an architecure with one hidden layer such as shown in Fig. 1 where the hidden nodes are product units and the output node is a sigmoidal unit.

A legitimate question is whether multiplicative units are actually needed in neural networks and whether their task cannot be done by some reasonably sized network of summing units. Indeed, for multiplication and exponentiation of integers in binary representation networks of threshold units with a small number of layers have been constructed that perform these and other arithmetic operations (for surveys see, e.g., Hofmeister, 1994; Siu et al., 1995). However, the drawback with these networks is that they grow in size, albeit polynomially, with the number of bits required for the representation of the numbers to be multiplied. Such a quality is certainly disastrous when one would like to process real numbers with possibly infinite precision. Therefore, networks of summing units do not seem to be an adequate substitute for genuine multiplicative units.

2.2.3 Units vs. Neurons

We have introduced all neuron models that will be considered in this article as "units". As common in the neural network literature we could have also called them neurons or gates synonymously, so that we could speak of a sigmoidal neuron or a product gate. In what follows we shall always use the term unit for the computational element of a neural network, thereby viewing it as an abstraction for the functionality it represents. In an artificial neural network a unit may be implemented by connecting several model neurons together, such as monomials and a linear unit form a higher-order unit. Likewise, a unit may also serve as a model for information processing in some part of a single biological neuron such as monomials or product units are used to model nonlinear interactions of synaptic inputs in dendritic trees.

2.3 Multiplication in Biological Neural Networks

There are several reasons why neurobiologists study multiplication as a computational mechanism underlying the behavior of neural systems. First, it can be used to model the nonlinearities involved in dendritic processing of synaptic inputs. Second, it is shown to arise in the output activity of individual model neurons or in neural populations. Third, it can be employed to explain how simple model networks can achieve complex behavior with biologically plausible methods. And finally, multiplicative neurons are actually found in real neural networks. In the following we mention some of the research that has been done into each of these directions. Further references regarding the fundamental role and the evidence of multiplication-like operations on all levels of neural information processing can be found in the article of Koch and Poggio (1992) and the book of Koch (1999).

2.3.1 Dendritic Multiplication and Division

In a large quantity of neuron models interaction of synaptic inputs is modeled as a linear operation. Thus, synaptic inputs are combined by adding them. This is evident for the summing units introduced above but it holds also for the more complex and biologically closer model of the leaky integrate-and-fire neuron[3] (see, e.g., Softky and Koch, 1995; Koch, 1999, for surveys of single neuron models). Linearity is believed to be sufficient for capturing the passive, or cable, properties of the dendritic membrane where synaptic inputs are currents that add.

From numerous studies using recording experiments or computer simulations sufficient evidence has arisen that synaptic inputs can interact nonlinearly when the synapses are co-localized on patches of dendritic membrane with specific properties. Thus, the spatial grouping of synapses on the dendritic tree is reflected in the computations performed at local branches. The summing operation, due to its associativity merely representing the dendrite as an amorphous device, does not take hold of this. Consequently, these local computations must be nonlinear, thereby enriching the computational power of the neuron by nonlinear computations that take place in the dendritic tree prior to the central nonlinearity of the neuron, the threshold. It has been convincingly argued that these dendritic nonlinearities should be modeled by multiplication. For instance, performing extensive computer experiments Mel (1992b, 1993) has found that excitatory voltage-dependent membrane mechanisms, such as NMDA receptor channels, could form a basis for multiplicative interactions among neighboring synapses. The exhibited so-called cluster sensitivity can give rise to complex pattern recognition tasks performed by single neurons. This is also shown in a learning experiment where a biologically plausible, Hebbian-like learning rule is introduced to manipulate the spatial ordering of synaptic connections onto the dendritic tree (Mel, 1992a).

Multiplicative-like operations in dendritic trees are also shown to take place in the form of division, an operation that is not performed by monomials and higher-order units but is available in the product units of Durbin and Rumelhart (1989) by using negative exponents (see Section 2.2.2). Neurobiological results

[3]The linearity referred to here concerns the way how *several* inputs interact, and not the neural response to a single synaptic input. Leaky integrate-and-fire models treat single synaptic inputs as nonlinearitites by describing the course of the postsynaptic response as a nonlinear function in time. Here, we do not take detailed temporal effects into account but consider synaptic inputs as represented by weighted analog variables.

exhibiting division arise mainly in investigations concerned with shunting inhibitory synapses. For instance, in a mathematical analysis of a simple model neuron Blomfield (1974) derives sufficient conditions for inhibitory synapses to perform division. The division operation can be seen as a continuous analog of the logical AND-NOT, also referred to as veto mechanism, a form in which it is studied by Koch et al. (1983). Using computer simulations they show that inhibitory synapses are able to perform such an analog veto operation. The capability of computing division is also found to be essential in a model constructed by Carandini and Heeger (1994) for the responses of cells in the visual cortex. The divisive operation of the cell membrane of the model neuron explains how nonlinear operations required for selectivity to various visual input stimuli, such as position, orientation, and motion direction, can be achieved by single neurons. These authors also show that their theoretical results compare well with physiological data from the monkey primary visual cortex. Perhaps one of the earliest modeling studies involving nonlinear dendritic processing, albeit not explicitly multiplicative, is due to Feldman and Ballard (1982) who demonstrate how complex cognitive tasks can be implemented in a biologically plausible way using their model neurons. For a detailed account of further ideas and results about dendritic computation we refer to the review by Mel (1994).

2.3.2 Model Neurons and Networks that Multiply

Beyond using multiplication for modeling dendritic computation, investigations have been aimed at revealing that entire neurons can function as multipliers. Already in the early days of cybernetics research it was known that under certain conditions a coincidence detecting neuron performs a multiplication by transforming the frequencies of input spikes arriving at two synaptic sites into an output spike frequency that is proportional to the product of the input frequencies (Küpfmüller and Jenik, 1961). Similar studies based on slightly different neuron models have been carried out by Srinivasan and Bernard (1976), Bugmann (1991), and by Rebotier and Droulez (1994) with comparable results. Bugmann (1991) argues that proximal synapses lead to a multiplicative behavior while with distal synapses the neuron operates in a summation mode. A further model of Bugmann (1992) includes time-dependent synaptic weights for the compensation of a problem caused by irregularities in input spike trains, which can deteriorate multiplicative behavior.

Tal and Schwartz (1997) show that even the summing operation can be used to compute multiplication. They find sufficient conditions for a leaky integrate-and-fire neuron to compute a nonlinearity close to the logarithm. In this way, the logarithm of a product can be obtained by summing the outputs of leaky integrate-and-fire neurons. By means of the ln-exp transform

$$xy = e^{\ln x + \ln y}$$

the logarithm is also a key ingredient for the biophysical implementation of multiplication proposed by Koch and Poggio (1992) (see also Koch, 1999). As such the transform is also suggested by Durbin and Rumelhart (1989) for their product units.

Multiplication is not only shown to arise in single elements, but also in an ensemble of neurons where it emerges as a property of the network. Salinas and Abbott (1996) show by computer simulations that population effects in a recurrent network can lead to multiplicative neural responses even when the individual neurons are not capable of computing a product.

2.3.3 Complex Neural Behavior through Multiplication

Assuming that multiplicative operations can be carried out in the nervous system, several researchers have established that model neural networks relying on multiplication are capable of solving complex tasks in a biologically plausible way. Poggio (1990) proposes a general framework for the approximate representation of high-dimensional mappings using radial basis functions. In particular, he shows that multiplication offers a computationally powerful and biologically realistic possibility of synthesizing high-dimensional Gaussian radial basis functions from low dimensions (see also Poggio and Girosi, 1990a,b). Building on this theory Pouget and Sejnowski (1997) demonstrate that basis functions could be used by neurons in the parietal cortex to encode sensory inputs in a format suitable for generating motor commands. The hidden units of their model network, which is trained using gradient descent, compute the product of a Gaussian function of retinal location with a sigmoidal function of eye position. Learning in multiplicative basis function networks is also investigated by Mel and Koch (1990) using a Hebbian method.

Olshausen et al. (1993) suggest a mechanism how the visual system could focus attention and achieve pattern recognition that is invariant to position and scale by dynamic routing of information. The neural circuit contains control units that make multiplicative contacts in order to dynamically modulate the strengths of synaptic connections. The already mentioned multiplicative network of Salinas and Abbott (1996) is conceived for solving a similar task. They show that it can transform visual information from retinal coordinates to coordinates that represent object locations with respect to the body. That nonlinear dendritic computation can be used in visual cortical cells for translation-invariant tuning is shown by the computer simulations of Mel et al. (1998).

A neuron model proposed by Maass (1998) includes input variables for the representation of firing correlations among presynaptic neurons, in addition to the common input variables that represent their firing rates. In the computation of the output the correlation variables are multiplicatively combined with monomials of rate variables. The model accounts for various phenomena believed to be relevant for complex information processing in biological networks such as synchronization and binding. Theoretical results show that the model neurons and networks are computationally more powerful than models that compute only in terms of firing rates.

2.3.4 Biological Multiplicative Neurons

Neurons that perform multiplicative-like computations have actually been identified in several biological nervous systems. Recordings performed by Andersen et al. (1985) from single neurons in the visual cortex of monkeys show that the selectivity of their receptive fields changes with the angle of gaze. Moreover, the interaction of the visual stimulus and the eye position in these neurons is found to be multiplicative. Thus they could contribute to the encoding of spatial locations independent of eye position.

Suga et al. (1990) describe arrays of neural filters in the auditory system of the bat that provide a means for processing complex sound signals by operating as multipliers. As such they are involved in a cross-correlation analysis of distance information conveyed by echo delays (see also Suga, 1990).

Investigating the visual system of the locust, Hatsopoulos et al. (1995) show that a single, motion-sensitive neuron, known as the lobula giant motion detector, performs a multiplication of two independent input signals. From experimental data they derive an algorithm that could be used in the visual system to anticipate the time of collision with approaching objects. The results reveal multiplication to be an elementary building block underlying motion detection in insects. In the work of Gabbiani et al. (1999) these investigations are continued resulting in a confirmation and generalization of the model.

In an experimental study of the visual system of the cat, Anzai et al. (1999a,b) find that neural mechanisms underlying binocular interaction are based on multiplication. They show that the well-studied simple and complex cells in the visual cortex perform multiplicative operations analogous to those that are used in an algorithm for the computation of interocular cross-correlation. These results provide a possible explanation of how the visual system could solve the stereo correspondence problem.

2.4 Learning in Artificial Multiplicative Networks

Early on in the history of artificial neural networks and machine learning, higher-order neurons have been used as natural extensions of the wide-spread linearly weighted neuron models. Perhaps one of the first machine learning algorithms using higher-order terms might be the checker playing program developed by Samuel (1959). Its decisions are based on computing a weighted sum including second-order Boolean conjunctions. In the sixties higher-order units were very common in pattern recognition where they appear in the form of polynomial discriminant functions (Cover, 1965; Nilsson, 1965; Duda and Hart, 1973). There is an obvious reason for this popularity: Higher-order units are computationally more powerful than single neurons since the higher-order terms act as hidden units. Furthermore, since these hidden units have no weights, there is no need to backpropagate errors when using gradient descent for training. Therefore, considering a higher-order unit as a linear unit in an augmented input space, all available learning methods for linear units are applicable. Later the delta rule, deleveloped within the framework of parallel distributed processing, was easily generalized to networks having hidden units of higher-order, resulting in a learning method for backpropagating errors in networks of higher-order sigmoidal units (Rumelhart et al., 1986a,b).

Soon, however, higher-order networks were caught up by what is known as the curse of dimensionality. Due to the fact that the number of monomials in a higher-order unit can be exponential in the input dimension, the complete representation of a higher-order unit succumbs to a combinatorial explosion. Even if the order of the monomials is restricted by some fixed value, the number of higher-order terms, and thus the number of free parameters to manipulate, is exponential in this bound—too large for many practical applications. Therefore, networks of higher-order neurons have gained importance in cases where the number of parameters can be kept low. One area of application for these sparse higher-order networks is the recognition and classification of patterns that underlie various geometric transformations such as scale, translation, and rotation. Maxwell et al. (1986) describe a method for incorporating invariance properties into a network of higher-order units that leads to a significant reduction of the number of parameters (see also Giles and Maxwell, 1987; Bishop, 1995, for an explanation). Thus, also faster training can be achieved by encoding into the network a priori knowledge that need not be learned. Several studies using invariant recognition and classification experiments show that higher-order networks can be superior to standard neural networks and other methods with respect to training time and generalization capabilities (see, e.g., Giles and Maxwell, 1987; Perantonis and Lisboa, 1992; Schmidt and Davis, 1993; Spirkovska and Reid, 1994). Invariance properties are also established for the so-called pi-sigma networks introduced by Ghosh and Shin (1992). A pi-sigma network consists of one hidden layer of summing units and has a monomial as output unit. Compared to a sigma-pi unit, it has the advantage of further reducing the number of adjustable parameters. The dependence of the number of weights on the order is linear for a pi-sigma network, whereas it can be exponential for a higher-order unit.

While invariance and pi-sigma networks are mostly used with restricted order, researchers have also striven to utilize the full computational power of higher-order networks, that is, without imposing any constraints on the degree of multiplicative interactions. In order for this to be accomplished, however, it is not sufficient to simply adopt learning methods from standard neural networks. Instead, a number of new learning algorithms have been developed that allow units of any order but enforce sparseness by incrementally building up higher orders and adding units only when necessary. Instances of such algorithms can be found in the work of Redding et al. (1993), Ring (1993), Kowalczyk and Ferrá (1994), Fahner and Eckmiller (1994), Heywood and Noakes (1995), and Roy and Mukhopadhyay (1997). A constructive algorithm for a class of pi-sigma networks, called ridge polynomial networks, is proposed by Shin and Ghosh (1995). A method that replaces units in a quadratic classifier with a limited number of terms is devised in the early work of Samuel (1959).

All these incremental algorithms, however, have the disadvantage, that they create higher order in a step-wise, discrete manner. Further, once a unit has its order it cannot change it, unless the unit is deleted and a new one added. This problem is overcome by the product units of Durbin and Rumelhart (1989) where high order appears in the form of real-valued, adjustable weights that are exponents of the input variables. Thus a product is able to learn any higher-order term. Moreover, its computational power is significantly larger compared to a monomial due to the fact that exponents can be non-integral and even negative. In further studies by Janson and Frenzel (1993), Leerink et al. (1995a,b), and Ismail and Engelbrecht (2000) product unit networks are found to be computationally more powerful than sigmoidal networks in many learning applications. In particular, it is shown that they can solve many well-studied problems using less neurons than networks with summing units. Besides the artificial neural network learning methods backpropagation and cascade correlation also more global optimization algorithms such as genetic algorithms, simulated annealing, and random search techniques are considered.

A considerable volume of research has been concerned with the encoding and learning of formal languages in higher-order recurrent networks. Especially second-order networks have proven to be useful for learning finite-state automata and recognizing regular languages (Giles et al., 1990; Pollack, 1991; Giles et al., 1992; Watrous and Kuhn, 1992; Omlin and Giles, 1996a,b). Higher-order units have also been employed as computational elements in Boltzmann machines and Hopfield networks in order to enlarge the capabilities and overcome the limitations of the first-order versions of these network models (Lee et al., 1986; Sejnowski, 1986; Psaltis et al., 1988; Venkatesh and Baldi, 1991a,b; Burshtein, 1998). Results on storage capacities and learning curves for single higher-order units are established by Yoon and Oh (1998) using an approach from statistical mechanics.

3 Vapnik-Chervonenkis and Pseudo Dimension

As the main subjects of study we now introduce the tools for assessing the computational and learning capabilities of multiplicative neural networks: the Vapnik-Chervonenkis (VC) dimension and the pseudo dimension. We give the definition of the VC dimension in Section 3.1 and exhibit as a helpful property that if the output node of a network is a summing unit then it does not matter for the VC dimension which one it is. We also postulate a condition for the parameter domain of product units that avoids the problem of having to deal with complex numbers.

In contrast to the VC dimension, the pseudo dimension takes into account that neural networks in general deliver output values that are not necessarily binary. Thus, the pseudo dimension appears suitable for networks that employ as output node a real-valued unit such as the linear or the sigmoidal unit, the monomial or the product unit. Nevertheless, after introducing the pseudo dimension in Section 3.2 we shall point out that the pseudo dimension of a neural network, being an upper bound on its VC dimension, can also almost serve as lower bound since it can be considered as the VC dimension of a slightly augmented network.

In Section 3.3 we give a brief review of VC dimension bounds for neural networks that have been previously established in the literature. The bounds concern networks with summing and multiplicative units and are the starting point for the results that will be derived in the subsequent sections.

3.1 Vapnik-Chervonenkis Dimension

Before we can give a formal definition of the VC dimension we require some additional notions. A partition of a set $S \subseteq \mathbb{R}^n$ into two disjoint subsets (S_0, S_1) is called a *dichotomy* of S. A function $f : \mathbb{R}^n \to \{0,1\}$ is said to *induce* the dichotomy (S_0, S_1) of S if f satisfies $f(S_0) \subseteq \{0\}$ and $f(S_1) \subseteq \{1\}$. More generally, if $\mathcal{F}$ is a class of functions mapping $\mathbb{R}^n$ to $\{0,1\}$ then $\mathcal{F}$ induces the dichotomy (S_0, S_1) if there is some $f \in \mathcal{F}$ that induces (S_0, S_1). Further, the class $\mathcal{F}$ *shatters* S if $\mathcal{F}$ induces all possible dichotomies of S.

Definition. *The* Vapnik-Chervonenkis (VC) dimension *of a class $\mathcal{F}$ of functions that map $\mathbb{R}^n$ to $\{0,1\}$, denoted* $\mathrm{VCdim}(\mathcal{F})$, *is the cardinality of the largest set shattered by $\mathcal{F}$. If $\mathcal{F}$ shatters arbitrarily large sets then the VC dimension of $\mathcal{F}$ is infinite.*

The definition applies to function classes but it is straightforward transferred to neural networks. Consider a network having connections and nodes labeled with programmable parameters, that is, weights and thresholds respectively, and having a certain number, say n, of input nodes and one output node. If the output node is a threshold unit, there is a a set of functions mapping $\mathbb{R}^n$ to $\{0,1\}$ naturally associated with this network, namely the set of functions obtained by assigning all possible values to the network parameters. Thus the VC dimension of a neural network is well defined if the output node is a threshold unit.

If the output node is not a threshold unit, the network functions are made $\{0,1\}$-valued to comply with the definition of the VC dimension. A general convention is to compare the output of the network with some fixed threshold θ, for instance $\theta = 1/2$ in the case of a sigmoidal unit. This is also common in applications when networks with continuous output values are used for classification tasks. More specifically, the binary output value of the network is obtained by applying the function $y \mapsto \text{sgn}(y - \theta)$ to the output y of the real-valued network.

Thus, θ can be considered as an additional parameter of the network which may be chosen independently for every dichotomy. However, one can show that for a sigmoidal or a linear output node the VC dimension does not rely on the specific values of this threshold. Moreover, it is also independent of the particular summing unit that is chosen. The following result makes this statement more precise. We omit its proof since it is easily obtained using scalings of the output weights and adjustments of the thresholds.

Proposition 1. *The VC dimension of a neural network remains the same regardless which summing unit (threshold, sigmoidal, or linear) is chosen for the output node.*

According to this statement we may henceforth assume without loss of generality that if the the output node of a network is a summing unit then it is a linear unit. Linear output nodes are commonly used, for instance, in neural networks constructed with the aim of approximating continuous functions (Pinkus, 1999).

Beyond analyzing single networks we also investigate the VC dimension of particular classes of networks. In this case we refer to the VC dimension of a class of networks as the VC dimension of the class of functions obtained by taking the union of the function classes which are associated with each particular network.

A problem arises with networks containing product units that receive negative inputs and have weights that are not integers. A negative number raised to some non-integral power yields a complex number and has no meaning in the reals. Since neural networks with complex outputs are hardly ever used in applications, Durbin and Rumelhart (1989) suggest a method how to proceed in this case. The idea is to discard the imaginary part and to use only the real component for further processing. For Boolean inputs this implies that the product unit becomes a summing unit that uses the cosine activation function. Although there are no problems reported from applications so far, this manipulation would have disastrous consequences for the VC dimension if extended to real-valued inputs. It is known that a summing unit with the sine or cosine activation function can shatter finite subsets of $\mathbb{R}$ of arbitrarily large cardinality (Sontag, 1992; Anthony and Bartlett, 1999). Therefore, no finite VC dimension bounds can in general be derived for networks containing such units.

To avoid these problems arising from negative inputs in combination with fractional weights we shall require that the following condition on the parameter domain of product units is always satisfied in the networks we consider.

Condition. *If an input x_i of a product unit is negative, the corresponding weight w_i is an integer.*

This presupposition guarantees that there are no complex outputs resulting from product units, and it still permits to view the product unit as a generalization of the monomial and, hence, networks containing product units as generalizations of networks with higher-order units. One of the main results in this article will be that networks with product units, where inputs and parameters satisfy the above condition, have a VC and pseudo dimension that is a low-degree polynomial in the number of network parameters and the network size. This will then, because product units comprise monomials, also lead to a similar bound for higher-order networks.

3.2 Pseudo Dimension

Besides the VC dimension several other combinatorial measures have been considered in the literature for the characterization of the variety of, in particular, a real-valued function class. The most relevant is the pseudo dimension, which is also well-known for providing bounds on the number of training examples in models of learning (Anthony and Bartlett, 1999).

Definition. *Let $\mathcal{F}$ be a class of functions that map $\mathbb{R}^n$ to $\mathbb{R}$. The* pseudo dimension *of $\mathcal{F}$, denoted* $\mathrm{Pdim}(\mathcal{F})$*, is the VC dimension of the class* $\{g : \mathbb{R}^{n+1} \to \{0,1\} \mid$ *there is some $f \in \mathcal{F}$ such that for all $x \in \mathbb{R}^n$ and $y \in \mathbb{R}$:* $g(x,y) = \mathrm{sgn}(f(x) - y)\}$.

The pseudo dimension of a neural network is then defined as the pseudo dimension of the class of functions computed by this network. Clearly, the VC dimension of a network is not larger than its pseudo dimension. The pseudo dimension and the VC dimension of a network with a summing unit as output node are even more closely related together. In particular, if one has an upper bound on the VC dimension of such a network in terms of the number of weights and computation nodes then one can also obtain an upper bound on the pseudo dimension. We shall show this now in a statement that slightly improves Theorems 14.1 and 14.2 of Anthony and Bartlett (1999) if the output node is a summing unit. The latter results require two more connections and one more computation node whereas the following manages with the same number of computation nodes and only one additional connection.

Proposition 2. *Let $\mathcal{N}$ be a neural network having as output node a summing unit (threshold, sigmoidal, or linear). If the output node is a threshold unit then* $\mathrm{Pdim}(\mathcal{N}) = \mathrm{VCdim}(\mathcal{N})$. *If the output node is a sigmoidal or a linear unit then there is a network $\mathcal{N}'$ with the same computation nodes as $\mathcal{N}$, one more input node, and one more connection such that* $\mathrm{Pdim}(\mathcal{N}) = \mathrm{VCdim}(\mathcal{N}')$.

Proof. Suppose $\mathcal{N}$ has n input nodes and let the function class $\mathcal{G}$ be defined by

$$\mathcal{G} = \{g : \mathbb{R}^{n+1} \to \{0,1\} \mid \text{there is some } f \text{ computed by } \mathcal{N} \text{ such that}$$
$$\text{for all } x \in \mathbb{R}^n \text{ and } y \in \mathbb{R} : g(x,y) = \mathrm{sgn}(f(x) - y)\}$$

according to the definition of the pseudo dimension of $\mathcal{N}$. If the output node of $\mathcal{N}$ is a threshold unit then it can easily be seen that for $s_1, \ldots, s_m \in \mathbb{R}^n$ and $u_1, \ldots, u_m \in \mathbb{R}$, $\mathcal{G}$ shatters $\{(s_1, u_1), \ldots, (s_m, u_m)\}$ if and only if $\mathcal{G}$ shatters $\{(s_1, 1/2), \ldots, (s_m, 1/2)\}$. Further, the latter set is shattered by $\mathcal{G}$ if and only if the set $\{s_1, \ldots, s_m\}$ is shattered by $\mathcal{N}$. Thus, $\mathrm{Pdim}(\mathcal{N}) = \mathrm{VCdim}(\mathcal{N})$.

If the output node is linear or sigmoidal we construct $\mathcal{N}'$ by adding to $\mathcal{N}$ a new input node for the variable y and a connection with weight -1 from this input node to the output node. Due to Proposition 1 we may employ without loss of generality a linear unit for the output node of $\mathcal{N}'$. Clearly, if $\mathcal{N}$ has a linear output node then the set $\{(s_1, u_1), \ldots, (s_m, u_m)\}$ is shattered by $\mathcal{G}$ if and only if the set $\{(s_1, u_1), \ldots, (s_m, u_m)\}$ is shattered by $\mathcal{N}'$. Further, if $\mathcal{N}$ has a sigmoidal output node then the set $\{(s_1, u_1), \ldots, (s_m, u_m)\}$ is shattered by $\mathcal{G}$ if and only if the set $\{(s_1, \sigma^{-1}(u_1)), \ldots, (s_m, \sigma^{-1}(u_m))\}$ is shattered by $\mathcal{N}'$. $\square$

3.3 Known Bounds on the VC Dimension of Neural Networks

We give a short survey of some previous results on the VC dimension of neural networks. Most of the bounds are for networks consisting solely of summing units. Since a summing unit can be considered as a multiplicative unit with order restricted to one, special cases of these results are relevant for the networks considered in this article. Further, some of the results hold for polynomial activation functions of restricted order and are given in terms of a bound on this order. It is therefore interesting to see how these bounds are related to the order-independent bounds derived here. We quote the results for the most part in their asymptotic form. The reader may find constants in the cited references or in the book of Anthony and Bartlett (1999). The bounds can be divided into three categories: for single units, for networks, and for classes of networks.

The VC dimension of a summing unit is known to be $n + 1$, where n is the number of variables, and is hence equal to the number of adjustable parameters. This holds for the threshold, sigmoidal, and linear unit. The proof of this fact goes back to an ancient result by Schläfli (1901). The pseudo dimension of these units has also been determined exactly, being $n+1$ for each of them as well (see, e.g., Anthony and Bartlett, 1999).

There is a large quantity of results on the VC dimension for neural networks that takes into account various unit types and network architectures. An upper bound for networks of threshold units is calculated by Baum and Haussler (1989). They show that a network with k computation nodes, which are all threshold units, and a total number of W weights has VC dimension $O(W \log k)$. This bound can also be derived from an earlier result of Cover (1968). Its tightness in the case of threshold units is established by two separate works. Sakurai (1993) shows that the architecture with one hidden layer has VC dimension $\Omega(W \log k)$

on real-valued inputs. Maass (1994) provides an architecture with two hidden layers respecting the bound $\Omega(W \log W)$ on Boolean inputs. There are also some bounds known for networks using more powerful unit types. A result of Goldberg and Jerrum (1995), which was independently obtained by Ben-David and Lindenbaum (1998), shows that neural networks employing as computation nodes higher-order units with an order bounded by some fixed value have VC dimension $O(W^2)$, where W is the number of weights. This bound even holds if the activation functions of the units are piecewise polynomial functions with an order bounded by a constant.[4] Koiran and Sontag (1997) construct networks of threshold and linear units with VC dimension $\Omega(W^2)$ thus showing that the quadratic upper bound of Goldberg and Jerrum (1995) is asymptotically optimal. Networks with depth restrictions are considered by Bartlett et al. (1998). They establish the bound $O(W \log W)$ for networks with piecewise polynomial activation functions, a fixed number of layers, and polynomials of bounded order. Expressed in terms of the number L of layers and the bound d on the order their result is $O(WL \log W + WL^2 \log d)$. A similar upper bound is obtained by Sakurai (1999). Bartlett et al. (1998) also show that $\Omega(WL)$ is a lower bound for piecewise polynomial networks with W weights and L layers. An upper bound for sigmoidal networks is due to Karpinski and Macintyre (1997). They show that higher-order sigmoidal networks with W weights and k computation nodes of order at most d have VC dimension $O(W^2k^2 + Wk \log d)$. Koiran and Sontag (1997) establish $\Omega(W^2)$ as a lower bound for sigmoidal networks.

Finally, some authors consider classes of networks. Hancock et al. (1994) show that the so-called class of nonoverlapping neural networks with n input nodes and consisting of threshold units has VC dimension $O(n \log n)$. In a nonoverlapping network all nodes, computation as well as input nodes, have fan-out at most one. That this bound is asymptotically tight for this class of networks is shown by Schmitt (1999) establishing the lower bound $\Omega(n \log n)$. Classes of higher-order units with order restrictions are studied by Anthony (1995). He shows that the VC dimension of the class of higher-order units with n inputs and order not larger than d is equal to $\binom{n+d}{d}$. Hence this VC dimension respects the upper and lower bound $\Theta(n^d)$. Karpinski and Werther (1993) consider classes of higher-order

[4]For polynomial neural networks Goldberg and Jerrum (1995) derive no explicit bound in terms of the order. The bound $O(W^2)$ for neural networks is obtained via an upper bound on the time needed by an algorithm to compute the output value of the network. This latter bound is linear in the running time of the algorithm. (More precisely $O(Wt)$, if the number of computation steps is bounded by t.) Since the algorithm may multiply two real numbers in constant time, the time required for the evaluation of a polynomial grows linearly in the order of the polynomial. These considerations lead to an upper bound that is linear in the order (see also Anthony and Bartlett, 1999, Theorem 8.7).

units with one input node, or univariate polynomials, with a restriction on the fan-in of the output node, or equivalently, on the number of monomials. They show that the class of higher-order units having one input node and at most k monomials has VC dimension $\Theta(k)$. Noteworthy, this result allows higher-order units of arbitrary order and does not depend on a bound for this order.

4 Upper Bounds

In the following we shall derive upper bounds on the VC dimension and the pseudo dimension of various feedforward networks with multiplicative units in terms of the number of parameters and the network size. We begin in Section 4.1 by considering networks with one hidden layer of product units and a summing unit as output node. Then, bounds for arbitrary networks, where each node may be a product or a sigmoidal unit, are established in Section 4.2. These results are employed in Section 4.3 to determine bounds for higher-order sigmoidal networks. In Section 4.4 we consider single higher-order units and, finally, in Section 4.5 we focus on product units and monomials.

4.1 Product Unit Networks with One Hidden Layer

We consider first the most widely used type of architecture which has one hidden layer of computation nodes such as shown in Fig. 1. Throughout this section we assume that the output node is a summing unit. The use of this class of networks is theoretically justified by the so-called universal approximation property. Results from approximation theory show that networks with one hidden layer of product or sigmoidal units and a linear unit as output node are dense in the set of continuous functions and hence can approximate any such function arbitrarily well (see, e.g., Leshno et al., 1993; Pinkus, 1999).

We recall from the condition on the parameter domain of product units stated in Section 3.1 that for product units to yield output values in the reals the input domain is restricted such that for a non-integral weight the corresponding input value is from the set $\mathbb{R}_0^+ = \{x \in \mathbb{R} : x \geq 0\}$. There is, however, still a problem when a product unit receives the value 0 and this input is weighted by a negative number. Then the output value of the unit is undefined. The possibility to forbid 0 as input value in general is certainly too restrictive and would go against many learning applications. Therefore, we use a default value in case that a unit raises the input value 0 to some negative power and say that the output value of such a unit is 0.[5] With these agreements the following can be shown. (Here and in

[5]Another way would be to introduce three-valued logic such that the output of a network that divides by 0 is some value 'undefined' that is neither 0 nor 1. This then leads to a different notion of dimension that takes multiple-valued

subsequent formulas we use "log" to denote the logarithm of base 2.)

Theorem 3. *Suppose that $\mathcal{N}$ is a neural network with one hidden layer consisting of k product units and let W be the total number of parameters (weights and threshold). Then $\mathcal{N}$ has pseudo dimension at most $(Wk)^2 + 8Wk\log(13Wk)$.*

The main ideas of the proof are first to derive from $\mathcal{N}$ and an arbitrary set S of input vectors a set of exponential polynomials in the parameter variables of $\mathcal{N}$. Then the next step is to consider the connected components of the parameter domain that arise from the zero-sets of these polynomials. A *connected component* (also referred to as a *cell*) of a set $D \subseteq \mathbb{R}^d$ is a maximal non-empty subset C of D such that any two points of C are connected by a continuous curve that lies entirely in C. The polynomials are constructed in such a way that the number of dichotomies that $\mathcal{N}$ induces on S is not larger than the number of connected components generated by these polynomials. Thus a bound on the number of connected components also limits the number of dichotomies. The basic tools for this calculation are provided by Karpinski and Macintyre (1997) who combined work of Warren (1968) and Khovanskiĭ (1991) to obtain for the first time polynomial bounds on the VC dimension of sigmoidal neural networks. These tools are explicated and further developed in the book of Anthony and Bartlett (1999).

We introduce two definitions from this book for sets of functions, namely the property to have regular zero-set intersections and the solution set components bound (Definitions 7.4 and 7.5 of Anthony and Bartlett, 1999). A set $\{f_1, \dots, f_k\}$ of differentiable real-valued functions on $\mathbb{R}^d$ is said to have *regular zero-set intersections* if for every non-empty set $\{i_1, \dots, i_l\} \subseteq \{1, \dots, k\}$ the Jacobian (i.e., the matrix of the partial derivatives) of $(f_{i_1}, \dots, f_{i_l}) : \mathbb{R}^d \to \mathbb{R}^l$ has rank l at every point of the set

$$\{a \in \mathbb{R}^d : f_{i_1}(a) = \dots = f_{i_l}(a) = 0\}.$$

A class $\mathcal{G}$ of real-valued functions defined on $\mathbb{R}^d$ has *solution set components bound B* if for every $k \in \{1, \dots, d\}$ and every $\{f_1, \dots, f_k\} \subseteq \mathcal{G}$ that has regular zero-set intersections the number of connected components of the set

$$\{a \in \mathbb{R}^d : f_1(a) = \dots = f_k(a) = 0\}$$

is at most B.

The proofs for the three subsequent lemmas can be found in the appendix.

---

outputs into account. Ben-David et al. (1995) study such dimensions and their relevance for learning. In particular, they show that a large variety of dimensions for multiple-valued functions is closely related in that they differ from each other at most by a constant factor. Therefore, should the three-valued approach be preferred the bounds derived here can also be used to obtain estimates for these generalized dimensions.

Lemma 4. *Let $\mathcal{G}$ be the class of polynomials of degree at most p in the variables $y_1, \ldots, y_d$ and in the exponentials $e^{g_1}, \ldots, e^{g_q}$, where $g_1, \ldots, g_q$ are fixed affine functions in $y_1, \ldots, y_d$. Then $\mathcal{G}$ has solution set components bound*

$$B \quad = \quad 2^{q(q-1)/2}[p(p+1)d+p]^d[(p+1)d(d+1)+1]^q.$$

Lemma 5. *Let q be a natural number and suppose $\mathcal{G}$ is the class of real-valued functions in the variables $y_1, \ldots, y_d$ satisfying the following conditions: For every $f \in \mathcal{G}$ there exist affine functions $g_1, \ldots, g_r$, where $r \leq q$, in the variables $y_1, \ldots, y_d$ such that f is an affine combination of $y_1, \ldots, y_d$ and $e^{g_1}, \ldots, e^{g_r}$. Then $\mathcal{G}$ has solution set components bound*

$$B \quad = \quad 2^{dq(dq-1)/2}[2(dq+d)+1]^{dq+d}[2(dq+d)(dq+d+1)+1]^{dq}.$$

A class $\mathcal{F}$ of real-valued functions is said to be *closed under addition of constants* if for every $c \in \mathbb{R}$ and $f \in \mathcal{F}$ the function $z \mapsto f(z)+c$ is a member of $\mathcal{F}$. The following result gives a stronger formulation of a bound stated in Theorem 7.6 of Anthony and Bartlett (1999).

Lemma 6. *Let $\mathcal{F}$ be a class of real-valued functions $(y_1, \ldots, y_d, x_1, \ldots, x_n) \mapsto f(y_1, \ldots, y_d, x_1, \ldots, x_n)$ that is closed under addition of constants and where each function in $\mathcal{F}$ is C^d in the variables $y_1, \ldots, y_d$. If the class $\mathcal{G} = \{(y_1, \ldots, y_d) \mapsto f(y_1, \ldots, y_d, s) : f \in \mathcal{F}, s \in \mathbb{R}^n\}$ has solution set components bound B then for any sets $\{f_1, \ldots, f_k\} \subseteq \mathcal{F}$ and $\{s_1, \ldots, s_m\} \subseteq \mathbb{R}^n$, where $m \geq d/k$, the set $T \subseteq \{0,1\}^{mk}$ defined as*

$$\begin{aligned} T = \{(&\mathrm{sgn}(f_1(a, s_1)), \ldots, \mathrm{sgn}(f_1(a, s_m)), \ldots \\ &\ldots, \mathrm{sgn}(f_k(a, s_1)), \ldots, \mathrm{sgn}(f_k(a, s_m))) : a \in \mathbb{R}^d\} \end{aligned}$$

satisfies

$$|T| \leq B \sum_{i=0}^{d} \binom{mk}{i} \leq B \left(\frac{emk}{d}\right)^d.$$

Now we have all that is required for the proof of Theorem 3.

Proof of Theorem 3. Assume that $\mathcal{N}$ is given as supposed, having k hidden product units and W parameters. Denote the weights of the hidden nodes by $w_{i,j}$ and let $v_0, v_1, \ldots, v_k$ be the weights of the output nodes where v_0 is the threshold. We first use an idea from Karpinski and Macintyre (1997) and divide for each $i \in \{1, \ldots, k\}$ the functions computed by $\mathcal{N}$ into three categories corresponding to the sign of the parameter v_i, that is, depending on whether $v_i < 0$, $v_i = 0$, or $v_i > 0$. This results in a total of 3^k categories for all $v_1, \ldots, v_k$. We consider each category separately and introduce new variables $v_1', \ldots, v_k'$ for the weights of the output node by defining

$$v_i' = \begin{cases} \ln v_i & \text{if } v_i > 0, \\ 0 & \text{if } v_i = 0, \\ \ln(-v_i) & \text{if } v_i < 0, \end{cases}$$

for $i = 1, \ldots, k$. Thus, within this category we can write the computation of $\mathcal{N}$ on some real input vector s as a function of the network parameters in the form

$$(v_0, v', w) \mapsto v_0 + b_1 e^{v_1'} s_{1,1}^{w_{1,1}} \cdots s_{1,p_1}^{w_{1,p_1}} + \cdots + b_k e^{v_k'} s_{k,1}^{w_{k,1}} \cdots s_{k,p_k}^{w_{k,p_k}},$$

where the $s_{i,j}$ are components of the input vector and $b_i \in \{0, 1\}$ is defined by

$$b_i = \begin{cases} 0 & \text{if } v_i' = 0 \text{ or } s_{i,j} = 0 \text{ for some } j \in \{1, \ldots, p_i\}, \\ 1 & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, k$. Note that if $s_{i,j} = 0$ and $w_{i,j} < 0$ for some i, j we define the output of the affected product unit to be 0. Thus we may assume that $s_{i,j} \neq 0$ for all i, j without loss of generality. Recall further that, according to the condition on the parameter domain of product units stated in Section 3.1, for $s_{i,j} < 0$ we have required $w_{i,j}$ to be an integer. In this case, however, the sign of $s_{i,j}$ can have an effect only when the weight is odd. And this effect on the product unit consists only in a possible change of the sign of its output value. Since the sign of $s_{i,j}$ is not known we consider the worst case and assume that each input vector generates all possible signs at the outputs of the product units. Therefore, we can restrict the input vectors to the positive reals if we take note of the fact that each input vector gives rise to at most 2^k functions of the form

$$(v_0, v', w) \mapsto v_0 \pm b_1 e^{v_1'} s_{1,1}^{w_{1,1}} \cdots s_{1,p_1}^{w_{1,p_1}} \pm \cdots \pm b_k e^{v_k'} s_{k,1}^{w_{k,1}} \cdots s_{k,p_k}^{w_{k,p_k}}.$$

For positive input values these functions can be rewritten as

$$\begin{aligned} (v_0, v', w) \mapsto v_0 \pm b_1 \exp(v_1' + w_{1,1} \ln s_{1,1} + \cdots + w_{1,p_1} \ln s_{1,p_1}) \pm \cdots \\ \cdots \pm b_k \exp(v_k' + w_{k,1} \ln s_{k,1} + \cdots + w_{k,p_k} \ln s_{k,p_k}). \end{aligned}$$

To obtain a bound on the pseudo dimension we want to estimate the number of dichotomies that are induced on some arbitrary set $\{(s_1, u_1), \ldots, (s_m, u_m)\}$, where $s_1, \ldots, s_m$ are input vectors for $\mathcal{N}$ and $u_1, \ldots, u_m$ are real numbers, by functions of the form $(x, z) \mapsto \operatorname{sgn}(f(x) - z)$, where f is computed by $\mathcal{N}$. We do this for each of the categories defined above separately. Thus, within one such category the number of dichotomies induced is at most as large as the cardinality of the set $T \subseteq \{0, 1\}^{m2^k}$ satisfying

$$\begin{aligned} T = \{(&\operatorname{sgn}(f_1(a, s_1', u_1)), \ldots, \operatorname{sgn}(f_1(a, s_m', u_m)), \ldots \\ &\ldots, \operatorname{sgn}(f_{2^k}(a, s_1', u_1)), \ldots, \operatorname{sgn}(f_{2^k}(a, s_m', u_m))) : a \in \mathbb{R}^W\} \end{aligned}$$

for real vectors $s'_1, \ldots, s'_m$ and functions $f_1, \ldots, f_{2^k}$, where each of these functions has the form

$$(y, x, z) \mapsto c_0 + y_0 + c_1 \exp(y_1 + y_{1,1}x_{1,1} + \cdots + y_{1,p_1}x_{1,p_1}) + \cdots \\ \cdots + c_k \exp(y_k + y_{k,1}x_{k,1} + \cdots + y_{k,p_k}x_{k,p_k}) - z$$

for real numbers $c_0, c_1, \ldots, c_k$ with $c_0 = 0$ and $c_i \in \{-1, 0, 1\}$ for $i = 1, \ldots, k$. The variables y_i and $y_{i,j}$ play the role of the network parameters, $x_{i,j}$ are the input variables receiving values $s'_{i,j} = \ln s_{i,j}$, and z is the input variable for $u_1, \ldots, u_m$. Let $\mathcal{F}$ denote the class of these functions arising for arbitrary real numbers $c_0, c_1, \ldots, c_k$. We have introduced c_0 in particular to make this class $\mathcal{F}$ closed under addition of constants. Now, for the vectors $s'_1, \ldots, s'_m$ and the real numbers $u_1, \ldots, u_m$ we consider the function class $\mathcal{G} = \{y \mapsto f(y, s'_i, u_i) : f \in \mathcal{F}, i = 1, \ldots, m\}$. Clearly, every element of $\mathcal{G}$ is an affine combination of W variables and k exponentials of affine functions in these variables. According to Lemma 5, $\mathcal{G}$ has solution set components bound

$$B = 2^{Wk(Wk-1)/2}[2(Wk+W)+1]^{Wk+W}[2(Wk+W)(Wk+W+1)+1]^{Wk}. \tag{1}$$

Since $\mathcal{F}$ is closed under addition of constants, we have from Lemma 6 that $|T| \leq B(em2^k/W)^W$, which is by the construction of T an upper bound on the number of dichotomies that are induced on any set of m vectors $\{(s_1, u_1), \ldots, (s_m, u_m)\}$. Since this bound is derived for network parameters chosen within one category, we obtain an upper bound for all parameter values by multiplying with the number of categories, which is 3^k. This yields the bound

$$|T| \leq B(em2^k/W)^W 3^k. \tag{2}$$

If there is a set of m vectors that is shattered then all 2^m dichotomies of this set must be induced. Since $|T|$ is an upper bound on this number this implies

$$m \leq \log B + W \log(em2^k/W) + k \log 3.$$

From the well-known inequality $\ln \alpha \leq \alpha\beta + \ln(1/\beta) - 1$, which holds for all $\alpha, \beta > 0$ (see, e.g., Anthony and Bartlett, 1999, Appendix A.1.1), we obtain for $\alpha = m$ and $\beta = (\ln 2)/(2W)$ that $W \log m \leq m/2 + W \log(2W/(e \ln 2))$. Using this in the above inequality we get

$$m \leq \log B + m/2 + W \log(2^{k+1}/\ln 2) + k \log 3,$$

which is equivalent to

$$m \leq 2 \log B + 2Wk + 2W \log(2/\ln 2) + 2k \log 3.$$

Substitution of the value for B yields

$$\begin{aligned} m &\leq Wk(Wk-1) + 2(Wk+W)\log[2(Wk+W)+1] \\ &\quad + 2Wk\log[2(Wk+W)(Wk+W+1)+1] \\ &\quad + 2Wk + 2W\log(2/\ln 2) + 2k\log 3. \end{aligned}$$

Simplifying and rearranging we obtain

$$\begin{aligned} m &\leq (Wk)^2 + (2Wk+2W)\log[2(Wk+W)+1] \\ &\quad + 2Wk\log[2(Wk+W)(Wk+W+1)+1] \\ &\quad + Wk + 2W\log(2/\ln 2) + 2k\log 3. \end{aligned}$$

The second and the third line together are less or equal to

$$2Wk(\log[13(Wk)^2] + 1/2 + \log(2/\ln 2) + \log 3)$$

which is equal to $2Wk\log[78\sqrt{2}(Wk)^2/\ln 2]$ and hence less than $4Wk\log(13Wk)$. The last term in the first line is at most $4Wk\log(5Wk)$. Using these relations we arrive at

$$m \;<\; (Wk)^2 + 4Wk\log(5Wk) + 4Wk\log(13Wk)$$

and hence at $m < (Wk)^2 + 8Wk\log(13Wk)$, which completes the proof of the theorem. $\square$

The constants in the bound of Theorem 3 can be made slightly smaller and one can get rid of the dependency on k of the term in the logarithm. Karpinski and Macintyre (1997) show by means of a more direct application of the method of Warren (1968) that for obtaining a solution set components bound for the class of functions considered here it is not necessary to introduce additional parameters as intermediate variables as we have done it in Lemma 5.[6] Instead, it is sufficient to employ the bound provided by Lemma 4 for affine functions in W variables and Wk fixed exponentials. This yields the improved solution set components bound

$$B \;=\; 2^{Wk(Wk-1)/2}[2W+1]^W[2W(W+1)+1]^{Wk} \tag{3}$$

which can be used in the proof of Theorem 3 in place of equation (1) to derive the following improved version of this theorem.

[6]Theorem 7.6 in Anthony and Bartlett (1999), which also builds on the method of Warren (1968), provides a more general method suitable for any function class that satisfies the conditions of Lemma 6. In particular, it applies to networks with more than one hidden layer. For such networks the direct method of Karpinski and Macintyre (1997) bears no advantage. The latter method also deals with these networks by introducing intermediate variables. We have chosen to employ the result of Anthony and Bartlett (1999) in the proof of Lemma 6 because it is more general and makes the proof of Theorem 3 shorter and easier to read.

Corollary 7. *Let $\mathcal{N}$ be a neural network with one hidden layer of k product units and W parameters. Then $\mathcal{N}$ has pseudo dimension at most $(Wk)^2 + 6Wk\log(8W)$.*

Proof. Using solution set components bound (3) in the proof of Theorem 3 we get

$$\begin{aligned} m &\leq (Wk)^2 + 2W\log(2W+1) \\ &\quad + 2Wk\log[2W(W+1)+1] + Wk + 2W\log(2/\ln 2) + 2k\log 3. \end{aligned}$$

The last line is less than $4Wk\log(8W)$ and the last term of the first line is at most $2W\log(3W)$. This implies $m < (Wk)^2 + 6Wk\log(8W)$. □

One of the constants can be further improved for networks that operate in the nonnegative domain only. The improvement is marginal, but the result demonstrates how the input restriction affects the calculation of the bound.

Corollary 8. *Let $\mathcal{N}$ be a neural network with one hidden layer of k product units and W parameters. If the inputs for $\mathcal{N}$ are restricted to nonnegative real numbers then $\mathcal{N}$ has pseudo dimension at most $(Wk)^2 + 6Wk\log(6W)$.*

Proof. If all inputs are positive then no sign changes have to be taken into account for the output values of the product units. Therefore, we can use in the proof of Theorem 3 that each input vector gives rise to 1 instead of 2^k functions. This gives the new bound $|T| \leq B(em/W)^W 3^k$ for inequality (2). Using solution set components bound (3) yields

$$\begin{aligned} m &\leq (Wk)^2 + 2W\log(2W+1) \\ &\quad + 2Wk\log[2W(W+1)+1] - Wk + 2W\log(2/\ln 2) + 2k\log 3. \end{aligned}$$

The last line is less than $4Wk\log(6W)$ implying $m < (Wk)^2 + 6Wk\log(6W)$. □

The networks considered thus far in this section have in common a rigid architecture with a prescribed number of hidden nodes. In some learning applications, however, it is customary not to fix the architecture in advance but to let the networks grow. In this case there is a variety of networks that can result from the learning algorithm. It might be possible to accommodate all these networks in a single large network so that a bound for the VC dimension of the class of networks is obtained in terms of a bound for the large network. Often, however, better bounds can be derived if one takes into account the constraint that underlies the growth of the network. In the following we assume that this growth is limited by a bound on the fan-out of the input nodes. Such networks with sparse connectivity have been suggested, for instance, by Lee et al. (1986) and Hancock et al. (1994).

Corollary 9. *Let $\mathcal{C}$ be the class of networks with one hidden layer of product units and n input nodes where every input node has fan-out at most l. Then $\mathcal{C}$ has pseudo dimension at most $4(nl)^4 + 18(nl)^2 \log(23nl)$.*

Proof. The number of networks in $\mathcal{C}$ is not larger than $(nl)^{nl}$ since there are at most this many ways to assign nl connections to nl hidden units. Each network has at most $2nl + 1$ parameters, where nl are for connections leading to hidden nodes and $nl + 1$ are for the output node. Thus, with $r = nl$, the number of dichotomies induced by $\mathcal{C}$, or more precisely, by functions of the form $(x, z) \mapsto \mathrm{sgn}(f(x) - z)$ where f is computed by $\mathcal{C}$, on a set of cardinality m is at most r^r times the number of dichotomies induced by a network with r hidden nodes and $2r + 1$ parameters. Using solution set components bound (3) we get from the proof of Corollary 7 with $k = r$ and $W = 2r + 1$

$$\begin{aligned} m \quad \leq \quad & (r(2r+1))^2 + 2(2r+1)\log(2(2r+1)+1) \\ & + 2r(2r+1)\log[2(2r+1)(2r+2)+1] + r(2r+1) \\ & + 2(2r+1)\log(2/\ln 2) + 2r\log 3 + r\log r, \end{aligned}$$

where the last term is due to the factor r^r. From this we obtain $m < 4r^4 + 6r\log(7r) + 12r^2\log(23r)$, and hence $m < 4r^4 + 18r^2\log(23r)$. Resubstituting $r = nl$ yields the claimed result. □

4.2 Networks with Product and Sigmoidal Units

We now consider feedforward architectures with an arbitrary number of layers. The networks are non-homogeneous in that each node may be a product or a sigmoidal unit independently of the other nodes. Pseudo dimension bounds for pure product unit networks with the output node being a summing unit are known from the previous section. We already noted in Section 2.2.2 that a network of product units only is equivalent to a single product unit. A bound for the single product unit will be given later in Section 4.5. Also, bounds for networks consisting solely of sigmoidal units have been established earlier by Karpinski and Macintyre (1997) and Anthony and Bartlett (1999). In the following we calculate bounds for networks containing both unit types.

Theorem 10. *Suppose $\mathcal{N}$ is a feedforward neural network with k computation nodes and W parameters where each computation node is a sigmoidal or a product unit. Then the pseudo dimension of $\mathcal{N}$ is at most $4(Wk)^2 + 20Wk\log(36Wk)$.*

Before proving this we determine a solution set components bound for classes of functions arising from feedforward networks. The following result considering arbitrarily many layers corresponds to Lemma 5, which was for networks with one hidden layer only. The proof is given in the appendix.

Lemma 11. *Let $\mathcal{G}$ be the class of real-valued functions in d variables computed by a network with r computation nodes where q of these nodes compute one of the functions $\alpha \mapsto c + 1/(1 + e^{-\alpha})$, $\alpha \mapsto c \pm e^{\alpha}$, or $\alpha \mapsto \ln \alpha$ for some arbitrary constant c, and $r - q$ nodes compute some polynomial of degree 2. Then $\mathcal{G}$ has solution set components bound*

$$B \;=\; 2^{dq(dq-1)/2}[6(2dr + d) + 2]^{2dr+d}[3(2dr + d)(2dr + d + 1) + 1]^{dq}.$$

Besides the above solution set components bound, the following proof uses Lemma 6 from the appendix.

Proof of Theorem 10. We show first that we can confine the argumentation to positive inputs. For some input vector s consider all functions computed by $\mathcal{N}$ that arise when s is fed into the network and the signs of the parameters are varied in all possible ways. Treating input values 0 to product units similarly as in the proof of Theorem 3 and taking into account the at most 2^W functions thus generated by each input vector, we may henceforth assume without loss of generality that all input values to product units are positive real numbers. (A number of 2^k functions is in general not sufficient, as it was in the proof of Theorem 3, since changing the sign of the output of some input unit, for instance, can modify the sign of an input to a sigmoidal unit. This cannot be compensated for by changing the sign of the sigmoidal unit, but by changing the sign of a weight.)

Thus, the number of dichotomies that are induced by functions of the form $(x, z) \mapsto \mathrm{sgn}(f(x) - z)$ with f being computed by $\mathcal{N}$ on some arbitrary set $\{(s_1, u_1), \ldots, (s_m, u_m)\}$ with input vectors $s_1, \ldots, s_m$ and real numbers $u_1, \ldots, u_m$ is at most as large as the cardinality of the set $T \subseteq \{0, 1\}^{m2^W}$ defined by

$$\begin{aligned} T = \{(&\mathrm{sgn}(f_1(a, s'_1, u_1)), \ldots, \mathrm{sgn}(f_1(a, s'_m, u_m)), \ldots \\ &\ldots, \mathrm{sgn}(f_{2^W}(a, s'_1, u_1)), \ldots, \mathrm{sgn}(f_{2^W}(a, s'_m, u_m))) : a \in \mathbb{R}^W\} \end{aligned}$$

where $s'_1, \ldots, s'_m$ are positive real vectors and $f_1, \ldots, f_{2^W}$ are the functions arising from $\mathcal{N}$ after making the sign variations described above. We allow that any arbitrary constant may be added to these functions and use $\mathcal{F}$ to denote this class of functions $(w, x, z) \mapsto f(w, x, z)$ in the network parameters w and the input variables x and z. Clearly then, $\mathcal{F}$ is closed under addition of constants.

Consider the class $\mathcal{G} = \{w \mapsto f(w, s'_i, u_i) : f \in \mathcal{F}, i = 1, \ldots, m\}$ of functions in W variables. Every function in $\mathcal{G}$ can be computed by a network where each computation node computes one of the functions $\alpha \mapsto c + 1/(1 + e^{-\alpha})$, $\alpha \mapsto c \pm e^{\alpha}$, $\alpha \mapsto \ln \alpha$, or some polynomial of degree 2. Here, c is some arbitrary constant that is required for the output nodes due to the closedness of $\mathcal{F}$ under addition of constants and also accommodates the subtraction of some u_i from the output. The "$\pm$" comes from the sign variation of the output of the product unit. The networks result as follows: Each product unit receives only positive inputs and can thus be written as

$$c_i \pm \exp(w_{i,1} \ln v_{i,1} + \cdots + w_{i,p_i} \ln v_{i,p_i}),$$

and each sigmoidal unit has the form

$$c_i + 1/(1 + \exp(-w_{i,1} v_{i,1} - \cdots - w_{i,p_i} v_{i,p_i} + t_i)),$$

where the $v_{i,j}$ are output values of other nodes. Therefore, each product unit can be decomposed into one function $\alpha \mapsto c \pm e^\alpha$, one polynomial of degree 2 and functions $\alpha \mapsto \ln \alpha$ applied to the outputs of other nodes. Further, each sigmoidal unit can be decomposed into one function $\alpha \mapsto c + 1/(1 + e^{-\alpha})$ and one polynomial of degree 2. This leads to a network with at most $k - 1$ nodes computing $\alpha \mapsto \ln \alpha$ (the logarithm of the output node is not needed), at most k nodes computing a polynomial of degree 2, and at most k nodes each of which computes either $\alpha \mapsto c \pm e^\alpha$ or $\alpha \mapsto c + 1/(1 + e^{-\alpha})$. In total, every function in $\mathcal{G}$ can be computed by a network with $3k - 1$ computation nodes of which $2k - 1$ nodes compute one of the functions $\alpha \mapsto c + 1/(1 + e^{-\alpha})$, $\alpha \mapsto c \pm e^\alpha$, or $\alpha \mapsto \ln \alpha$. Lemma 11 with $d = W$, $r = 3k - 1$, and $q = 2k - 1$ shows that $\mathcal{G}$ has solution set components bound

$$\begin{aligned} B \;=\; & 2^{W(2k-1)(W(2k-1)-1)/2}[6(2W(3k-1)+W)+2]^{2W(3k-1)+W} \\ & \cdot [3(2W(3k-1)+W)(2W(3k-1)+W+1)+1]^{W(2k-1)}. \end{aligned}$$

Since $\mathcal{F}$ is closed under addition of constants, it follows from Lemma 6 that $B(2em2^W/W)^W$ is an upper bound on the cardinality of T and, thus, on the number of dichotomies induced on any set of cardinality m. If such a set is shattered this implies

$$m \;\leq\; \log B + W \log(em2^W/W).$$

Using $W \log m \leq m/2 + W \log(2W/(e \ln 2))$ (see the proof of Theorem 3) we obtain

$$m \;\leq\; 2 \log B + 2W^2 + 2W \log(2/\ln 2)$$

and, after substituting the value for B,

$$\begin{aligned} m \;\leq\; & W(2k-1)(W(2k-1)-1) \\ & + (4W(3k-1)+2W)\log[6(2W(3k-1)+W)+2] \\ & + 2W(2k-1)\log[3(2W(3k-1)+W)(2W(3k-1)+W+1)+1] \\ & + 2W^2 + 2W\log(2/\ln 2). \end{aligned}$$

Simplifying and rearranging leads to

$$\begin{aligned} m \ \leq \ & 4(Wk)^2 - 4W^2k + 3W^2 + (12Wk - 2W)\log[6(6Wk - W) + 2]) \\ & + 2W(2k-1)\log[3(6Wk - W)(6Wk - W + 1) + 1] \\ & - W(2k-1) + 2W\log(2/\ln 2). \end{aligned}$$

The last two lines together are less than

$$4Wk(\log[109(Wk)^2] + \log(2/\ln 2))$$

which equals $4Wk\log[218(Wk)^2/\ln 2]$ and is less than $8Wk\log(18Wk)$. The last three terms of the first line together are less than $12Wk\log(36Wk)$. Thus, we may conclude that $m < 4(Wk)^2 + 20Wk\log(36Wk)$. □

The networks considered in Theorem 10 have fixed units in the sense that it is determined in advance for a computation node whether it is to be a product or a sigmoidal unit. One can imagine situations in which the learning algorithm chooses the unit type for each node. Then the function class is no longer represented by a single network, but by a class of networks. An inspection of the proof of Theorem 10 in this regard shows that its argumentation does not depend on knowing which unit type is given. Hence, the same bound holds if the unit type of each computation node is variable.

Corollary 12. *Let $\mathcal{C}$ be a class of feedforward neural networks where each network has k computation nodes and W parameters, and where each computation node is a product unit or a sigmoidal unit. Then $\mathcal{C}$ has pseudo dimension at most $4(Wk)^2 + 20Wk\log(36Wk)$.*

4.3 Higher-Order Sigmoidal Networks

A neural network consisting of higher-order sigmoidal units can be considered as a network of product and sigmoidal units where the weights of the product units are restricted to the nonnegative integers. Thus, an upper bound on the VC dimension or pseudo dimension for a higher-order network is obtained by means of the same network with the monomials replaced by product units and the exponents considered as variables. We distinguish between two ways of viewing higher-order sigmoidal networks: The exponents of the monomials can be parameters of the network or they can be fixed. We consider the latter case first, that is, when the exponents are not variable. We further emphasize that we do not explicitly count the number of monomials in a higher-order network. For both cases we obtain bounds that do not impose any restriction on the order of the monomials.

Theorem 13. *Suppose $\mathcal{N}$ is a network with k computation nodes and W parameters where each computation node is a higher-order sigmoidal unit with fixed exponents. Then $\mathcal{N}$ has pseudo dimension at most $36(Wk)^4 + 136(Wk)^2\log(12Wk)$.*

Proof. The parameters of a sigmoidal higher-order network are the weights and thresholds of the sigmoidal units only. Further, a sigmoidal higher-order network has the following properties: First, every input node and every non-output node that is a sigmoidal unit feeds its output only to monomials. Second, every sigmoidal unit receives its input only from monomials through parameterized connections. Thus, after replacing the monomials by product units we can guide the proof analogously to that of Theorem 10 with the following ingredients: Since sign variations of input vectors affect product units only and there are at most W product units, it suffices to consider as upper bound on the number of dichotomies that $\mathcal{N}$ induces via functions of the form $(x, z) \mapsto \mathrm{sgn}(f(x) - z)$ on a set of cardinality m the cardinality of a set T with $T \subseteq \{0,1\}^{m2^W}$ defined as in the proof of Theorem 10. The networks that give rise to the function class $\mathcal{G}$ have at most

- k nodes computing the function $\alpha \mapsto c + 1/(1 + e^{-\alpha})$ (due to the sigmoidal units),
- W nodes computing the function $\alpha \mapsto c \pm e^{-\alpha}$ (due to the product units),
- k nodes computing the function $\alpha \mapsto \ln \alpha$ (only logarithms of sigmoidal units are needed),
- $W + k$ nodes computing a polynomial of degree 2 (due to the product and sigmoidal units).

This yields networks having at most $2W + 3k \leq 5Wk$ nodes of which up to $W + 2k \leq 3Wk$ compute a non-polynomial function. Since each product unit, receiving inputs from sigmoidal units only, has at most k exponents, the functions in $\mathcal{G}$ have in total at most $W + Wk \leq 2Wk$ variables. Thus, with these assumptions we get from Lemma 11 using $d = 2Wk$, $r = 5Wk$, and $q = 3Wk$ the solution set components bound

$$\begin{aligned} B &= 2^{6(Wk)^2(6(Wk)^2-1)/2}[6(20(Wk)^2 + 2Wk) + 2]^{20(Wk)^2+2Wk} \\ &\qquad \cdot [3(20(Wk)^2 + 2Wk)(20(Wk)^2 + 2Wk + 1) + 1]^{6(Wk)^2}. \end{aligned}$$

As in the proof of Theorem 10 we infer from Lemma 6 using $T \subseteq \{0,1\}^{m2^W}$ that $B(em2^W/(2Wk))^{2Wk}$ is an upper bound on the number of dichotomies induced by $\mathcal{N}$ on any set of cardinality m. Thus, if such a set is shattered we have

$$m \leq \log B + 2Wk \log(em2^W/(2Wk)),$$

from which we get

$$m \leq 2\log B + 4W^2k + 4Wk\log(2/\ln 2)$$

using $2Wk \log m \leq m/2 + 2Wk \log(4Wk/(e \ln 2))$ (see the proof of Theorem 3). With the above solution set components bound this implies

$$\begin{aligned} m \leq\ & 6(Wk)^2(6(Wk)^2 - 1) \\ & + (40(Wk)^2 + 4Wk) \log[6(20(Wk)^2 + 2Wk) + 2] \\ & + 12(Wk)^2 \log[3(20(Wk)^2 + 2Wk)(20(Wk)^2 + 2Wk + 1) + 1] \\ & + 4W^2k + 4Wk \log(2/\ln 2). \end{aligned}$$

From this we obtain

$$\begin{aligned} m \leq\ & 36(Wk)^4 + (40(Wk)^2 + 4Wk) \log[6(20(Wk)^2 + 2Wk) + 2] \\ & + 12(Wk)^2 \log[3(20(Wk)^2 + 2Wk)(20(Wk)^2 + 2Wk + 1) + 1] \\ & - 6(Wk)^2 + 4W^2k + 4Wk \log(2/\ln 2). \end{aligned}$$

The last two lines together are less than

$$12(Wk)^2(\log[1519(Wk)^4] + \log(2/\ln 2))$$

which is equal to $12(Wk)^2 \log[3038(Wk)^4/\ln 2]$ and less than $48(Wk)^2 \log(9Wk)$. Since the second term of the first line is less than $88(Wk)^2 \log(12Wk)$ we get $m < 36(Wk)^4 + 136(Wk)^2 \log(12Wk)$ as claimed. □

The bound we derive next concerns higher-order sigmoidal networks with variable exponents. As is to be expected, the bound is smaller since more parameters are counted.

Theorem 14. *Suppose $\mathcal{N}$ is a higher-order sigmoidal network with k computation nodes and W parameters that include the exponents of the monomials. Then the pseudo dimension of $\mathcal{N}$ is at most $9(W^2k)^2 + 34W^2k \log(68W^2k)$.*

Proof. In comparison to the case with fixed exponents in Theorem 13 the difference here is that the exponents of the monomials do not increase the number of parameters since they are already counted in W. Thus, Lemma 11 with $d = W$, $r = 5Wk$, and $q = 3Wk$ provides solution set components bound

$$\begin{aligned} B =\ & 2^{3W^2k(3W^2k-1)/2}[6(10W^2k + W) + 2]^{10W^2k+W} \\ & \cdot [3(10W^2k + W)(10W^2k + W + 1) + 1]^{3W^2k}, \end{aligned}$$

and Lemma 6 yields $B(em2^W/W)^W$ as upper bound on the number of dichotomies induced on a set of cardinality m. Assuming that such a set is shattered we obtain

$$m \leq 2 \log B + 2W^2 + 2W \log(2/\ln 2)$$

similarly as in the proof of Theorem 10. After substituting the above value for B we get

$$\begin{aligned} m \leq\ & 3W^2k(W^2k-1) + (20W^2k+2W)\log[6(10W^2k+W)+2] \\ & + 6W^2k\log[3(10W^2k+W)(10W^2k+W+1)+1] \\ & + 2W^2 + 2W\log(2/\ln 2), \end{aligned}$$

which implies

$$\begin{aligned} m \leq\ & 9(W^2k)^2 + (20W^2k+2W)\log[6(10W^2k+W)+2] \\ & + 6W^2k\log[3(10W^2k+W)(10W^2k+W+1)+1] \\ & - 3W^2k + 2W^2 + 2W\log(2/\ln 2). \end{aligned}$$

The last two lines together are less than

$$6W^2k(\log[397W^4k^2] + \log(2/\ln 2))$$

which is equal to $6W^2k\log[794W^4k^2/\ln 2]$ and less than $12W^2k\log(34W^2k)$. The second term of the first line is less than $22W^2k\log(68W^2k)$. Thus, we have $m < 9(W^2k)^2 + 34W^2k\log(68W^2k)$. □

4.4 Single Higher-Order Units

Since a single unit can be viewed as a small network, bounds on the VC dimension and pseudo dimension for product units and higher-order units can be obtained from previous sections. For particular cases, however, we shall establish significant improvements in the following. We look at single higher-order units and classes of these units first, then, in the following section, we consider single product units and classes of monomials.

We recall from Section 2.2.2 the definition of a higher-order unit which has the form

$$w_1M_1 + w_2M_2 + \cdots + w_kM_k - t$$

where $M_1, \ldots, M_k$ are monomials. We also remember that the set $\{M_1, \ldots, M_k\}$ is called the structure of the higher-order unit. We can view such a unit as a network with one hidden layer and a linear unit as output node. If a threshold or sigmoidal unit is employed for the output node, we have the higher-order variants of the threshold and sigmoidal unit, respectively, which were also defined in Section 2.2.2. According to Proposition 1, when studying the VC dimension it makes no difference which summing unit is employed for the output node. Thus, we may focus on linear output units without loss of generality.

If the structure of a higher-order unit is fixed, its only parameters are the weights and the threshold of the output node. Thus, with fixed structure the VC dimension cannot be larger than the VC dimension of a summing unit with the same number of parameters. This fact is employed in the following result.

Lemma 15. *Let $\mathcal{N}$ be a higher-order unit with fixed structure that consists of k monomials. Then the number of dichotomies that $\mathcal{N}$ induces on a set of cardinality m is at most*

$$2\sum_{i=0}^{k}\binom{m-1}{i} < 2\left(\frac{e(m-1)}{k}\right)^{k},$$

for $m > k \geq 1$, and the VC dimension of $\mathcal{N}$ is at most $k+1$.

Proof. Let $\{M_1, \ldots, M_k\}$ be the structure of the higher-order unit. Assume further that S is a set of m input vectors and consider the set of vectors

$$S' = \{(M_1(s), \ldots, M_k(s)) : s \in S\}.$$

Obviously, every dichotomy induced by a summing unit on S' corresponds to at least one dichotomy induced by $\mathcal{N}$ on S. Hence, the number of dichotomies that $\mathcal{N}$ induces on S cannot be larger than the number of dichotomies that a summing unit with k input variables induces on S'. The latter quantity is known to be not larger than $2\sum_{i=0}^{k}\binom{m-1}{i}$, an expression which is less than $2(e(m-1)/k)^k$ (see Anthony and Bartlett, 1999, Theorems 3.1 and 3.7, respectively). Thus, we have the claimed upper bound on the number of dichotomies. The VC dimension of a summing unit with k input variables is known to be $k+1$ (Anthony and Bartlett, 1999, Section 3.3). $\square$

We apply this result in the next two theorems where we consider higher-order units with variable structure. The variability is given in terms of a class of units which underly a certain connectivity constraint. In the first class the fan-in of the output node, or the number of monomials, is limited. In the second class we have a bound on the fan-out of the input nodes, or the number of monomials in which each variable may occur. Both classes can be considered as multivariate generalizations of a function class studied by Karpinski and Werther (1993). They define a polynomial to be t-sparse if it has at most t non-zero coefficients. Thus, a t-sparse univariate polynomial has at most t monomials, and each variable occurs in at most t of them. Karpinski and Werther (1993) show that the class of t-sparse univariate polynomials has VC dimension $\Theta(t)$.

Theorem 16. *Let $\mathcal{C}$ be the class of higher-order units with n input variables and at most k monomials where each variable has an exponent of value at most d. Then $\mathcal{C}$ has VC dimension at most*

$$\min\{(k(nk+k+1))^2 + 6k(nk+k+1)\log(8(nk+k+1)),\ 2nk\log(9d)\}.$$

Proof. Obviously, a network with one hidden layer of product units and a linear unit as output node comprises the set of functions computed by the units in $\mathcal{C}$. Thus, the first bound is obtained from Corollary 7 considering the exponents of the variables as parameters and using the fact that a product unit can compute any monomial. Since in a higher-order unit with at most k monomials there are no more than nk occurrences of variables, this leads to a total number of $nk + k + 1$ parameters.

The second bound is established as follows: Each occurrence of a variable can have an exponent from $\{0, 1, \ldots, d\}$ (where we consider a monomial to be 0 if all its variables have exponent 0). Therefore, $(d+1)^{nk}$ is an upper bound on the number of structures in $\mathcal{C}$. From this bound and Lemma 15 we infer that the number of dichotomies induced by $\mathcal{C}$ on a set S of m input vectors is at most

$$(d+1)^{nk} \cdot 2\left(\frac{e(m-1)}{k}\right)^k.$$

If S is shattered, its cardinality satisfies

$$m \quad \leq \quad nk\log(d+1) + k\log(e(m-1)/k) + 1.$$

Now we use that $\ln \alpha \leq \alpha\beta + \ln(1/\beta) - 1$ for $\alpha, \beta > 0$ (see, e.g., Anthony and Bartlett, 1999, Appendix A.1.1). Assuming $m > 1$, we may substitute $\alpha = m-1$ and $\beta = (\ln 2)/(2k)$ to obtain $k\log(m-1) \leq (m-1)/2 + k\log(2k/(e\ln 2))$. From this we have

$$m \quad \leq \quad 2nk\log(d+1) + 2k\log(2/\ln 2) + 1.$$

The right-hand side is at most $2nk(\log(d+1) + \log(2/\ln 2) + 1/2)$ which is equal to $2nk\log(2\sqrt{2}(d+1)/\ln 2)$, and this is less than $2nk\log(9d)$. $\square$

Theorem 17. *Let $\mathcal{C}$ be the class of higher-order units with n input variables where each variable occurs in at most l monomials and has an exponent of value at most d. Then $\mathcal{C}$ has VC dimension at most*

$$\min\{4(nl)^4 + 18(nl)^2\log(23nl),\ 2n^2l\log(9d),\ 2nl\log(5dnl)\}.$$

Proof. The first bound is due to Corollary 9 by considering a higher-order unit as a network with one hidden layer of product units and a linear unit as output node. The second bound is obtained from Theorem 16 using the fact that every unit in $\mathcal{C}$ has at most nl monomials.

We derive the third bound as follows: Since there are at most nl connections between the input nodes and the monomials, there are at most $(nl)^{nl}$ possibilities of connecting the input nodes with the monomials. Further, there are at most d^{nl} ways to assign exponents from $\{1, \ldots, d\}$ to the occurrences of the variables. Thus, there are at most $(dnl)^{nl}$ different structures in $\mathcal{C}$. This bound together with Lemma 15 implies that the number of dichotomies induced by $\mathcal{C}$ on a set S of cardinality m is not larger than

$$(dnl)^{nl} \cdot 2\left(\frac{e(m-1)}{nl}\right)^{nl}.$$

This expression is equal to $2(ed(m-1))^{nl}$. If S is shattered by $\mathcal{C}$ it follows that

$$m \leq nl \log(ed(m-1)) + 1.$$

Similarly as in the previous proof, assuming $m > 1$, we can make use of $nl \log(m-1) \leq (m-1)/2 + nl \log(2nl/(e \ln 2))$ to obtain

$$m \leq 2nl \log(2dnl/\ln 2) + 1.$$

The right-hand side is not larger than $2nl(\log(2dnl/\ln 2) + 1/2)$ which equals $2nl \log(2\sqrt{2}dnl/\ln 2)$. Since this is less than $2nl \log(5dnl)$, the third bound follows. $\square$

4.5 Single Product Units and Monomials

Next, we look at a single product unit. Since it can be viewed as a (trivial) network with one hidden unit, an upper bound on its VC dimension is immediately obtained from Corollary 7 in the form of $n^2 + 6n \log(8n)$, where n is the number of variables. The following statement shows that the exact values for its VC dimension and pseudo dimension are considerably smaller. This result also contains the first lower bound of this article.

Theorem 18. *The VC dimension and the pseudo dimension of a product unit with n input variables are both equal to n.*

Proof. That n is a lower bound easily follows from the fact that a monomial with n variables shatters the set of unit vectors from $\{0,1\}^n$, that is, the set of vectors with a 1 in exactly one position. We show now that n is an upper bound on the pseudo dimension, so that the theorem follows. The idea is to derive the upper bound by means of the pseudo dimension of a linear unit.

Let $(w,x) \mapsto f(w,x)$ be the function computed by a product unit, that is,

$$f(w,x) = x_1^{w_1} x_2^{w_2} \cdots x_n^{w_n},$$

where $w_1, \ldots, w_n$ are parameters and $x_1, \ldots, x_n$ are input variables. Consider some arbitrary set $S = \{(s_1, u_1), \ldots, (s_m, u_m)\}$ where $s_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}$ for $i = 1, \ldots, m$. According to the definition, the pseudo dimension of a product unit is the cardinality of the largest such S that is shattered by functions of the form

$$(w,x,y) \mapsto \operatorname{sgn}(f(w,x) - y) \tag{4}$$

with parameters $w_1, \ldots, w_n$ and input variables $x_1, \ldots, x_n, y$. We use the same idea as in the proof of Theorem 3 to get rid of negative input values: According to the assumptions on the parameters (see Section 3.1), if some input value is negative, its weight may take on integer values only. The sole effect of changing the sign of an input value, therefore, is to possibly change the sign of the output of the product unit. Hence, if we consider the set $S' = \{(s'_1, u_1), \ldots, (s'_m, u_m)\}$, where s'_i arises from s_i by taking absolute values in all components, then the number of dichotomies induced on S is less or equal to the cardinality of the set $T \subseteq \{0,1\}^{2m}$ defined by

$$\begin{aligned} T = \{(&\mathrm{sgn}(f(a,s'_1) - u_1), \ldots, \mathrm{sgn}(f(a,s'_m) - u_m), \\ &\mathrm{sgn}(-f(a,s'_1) - u_1), \ldots, \mathrm{sgn}(-f(a,s'_m) - u_m)) : a \in \mathbb{R}^n\}. \end{aligned}$$

Since we are interested in an upper bound for $|T|$, we may assume without loss of generality that no s'_i has some component equal to 0, because in that case the value $\mathrm{sgn}(f(a,s'_i) - u_i)$ is the same for all $a \in \mathbb{R}^n$. Thus, for inputs from S' the function f can be written as

$$f(w,x) = \exp(w_1 \ln x_1 + \cdots + w_n \ln x_n).$$

Since $f(a,s'_i) > 0$ for every $a \in \mathbb{R}^n$ and $i = 1, \ldots, m$, we may suppose that each of $u_1, \ldots, u_m$ is different from 0. Now, depending on whether $u_i > 0$ or $u_i < 0$, exactly one of

$$\mathrm{sgn}(f(a,s'_i) - u_i), \mathrm{sgn}(-f(a,s'_i) - u_i)$$

changes when a varies while the other one remains constant. Hence, by defining

$$b_i = \begin{cases} 1 & \text{if } u_i > 0, \\ -1 & \text{if } u_i < 0, \end{cases}$$

we select the varying components and obtain with

$$T' = \{(\mathrm{sgn}(b_1 f(a,s'_1) - u_1), \ldots, \mathrm{sgn}(b_m f(a,s'_m) - u_m)) : a \in \mathbb{R}^n\}.$$

a set $T' \subseteq \{0,1\}^m$ which has the same cardinality as T. Consider the function $(w,x) \mapsto g(w,x)$ defined for positive input vectors x as

$$g(w,x) = w_1 \ln x_1 + \cdots + w_n \ln x_n,$$

that is, $g = \ln \circ f$. If $u_i > 0$ then

$$\begin{aligned} \mathrm{sgn}(b_i f(a,s'_i) - u_i) &= \mathrm{sgn}(f(a,s'_i) - b_i u_i) \\ &= \mathrm{sgn}(\ln(f(a,s'_i)) - \ln(b_i u_i)) \\ &= \mathrm{sgn}(g(a,s'_i) - \ln(b_i u_i)), \end{aligned}$$

and if $u_i < 0$ then

$$\begin{aligned} \mathrm{sgn}(b_i f(a,s'_i) - u_i) &= \mathrm{sgn}(-f(a,s'_i) + b_i u_i) \\ &= \mathrm{sgn}(-\ln(f(a,s'_i)) + \ln(b_i u_i)) \\ &= \mathrm{sgn}(-g(a,s'_i) + \ln(b_i u_i)). \end{aligned}$$

This implies that

$$\operatorname{sgn}(b_i f(a, s'_i) - u_i) \;=\; \operatorname{sgn}(b_i g(a, s'_i) - b_i \ln(b_i u_i))$$

for every $a \in \mathbb{R}^n$ and $i = 1, \ldots, m$. From this we have that $|T'|$ is not larger than the number of dichotomies induced on the set

$$S'' = \{(b_1 \ln(s'_1), b_1 \ln(b_1 u_1)), \ldots, (b_m \ln(s'_m), b_m \ln(b_m u_m))\}$$

(where logarithms of vectors are taken component-wise) by functions of the form

$$(w, x, y) \mapsto \operatorname{sgn}(w_1 x_1 + \cdots + w_n x_n - y). \tag{5}$$

To conclude the proof, assume that some set of cardinality $n + 1$ is shattered by functions of the form (4). Then from the reasoning above we may infer that some set of cardinality $n + 1$ is shattered by functions of the form (5). This, however, contradicts the fact that the pseudo dimension of a linear unit with n parameters is at most n (see, e.g., Anthony and Bartlett, 1999, Theorem 11.6). □

Since a single product unit can compute any monomial, the previous result implies that the VC dimension and the pseudo dimension of the class of monomials do not grow with the degree of the monomials, but are identical with the number of variables.

Corollary 19. *The VC dimension and the pseudo dimension of the class of monomials with n input variables are both equal to n.*

5 Lower Bounds

The results presented thus far exclusively dealt with upper bounds, except for the single product unit and the class of monomials for which a lower bound has been given in Section 4.5. In the following we establish some further lower bounds for the VC dimension which are then, by definition, also lower bounds for the pseudo dimension. The results in Section 5.1 concern classes of higher-order units and show that some upper bounds given in Section 4.4 cannot be improved in a certain sense. The main result of Section 5.2 is a superlinear lower bound for networks that consist of product and linear units and have constant depth.

5.1 Higher-Order Units

Two types of restrictions have been considered for classes of higher-order units in Section 4.4. First, a bound k was imposed on the number of monomials or, equivalently, on the fan-in of the output node. Second, the number of occurrences of each variable or, equivalently, the fan-out of the input nodes was limited by some bound l. We give a lower bound for the latter class first. The following result provides the essential means.

Theorem 20. *Let $m, r \geq 1$ be natural numbers. Suppose $\mathcal{C}$ is the class of higher-order units with $m + 2^r$ variables, where each variable occurs in at most one monomial and, if so, with exponent 1. Then there is a set of cardinality $m \cdot r$ that is shattered by $\mathcal{C}$.*

Proof. We show that the class $\mathcal{C}$ shatters some set $S \subseteq \{-1, 1\}^{m+2^r}$ which is constructed as the direct product of a set $U \subseteq \{-1, 1\}^m$ and a set $V \subseteq \{-1, 1\}^{2^r}$. First, let $U = \{u_1, \dots, u_m\}$ be defined by

$$u_{i,j} = \begin{cases} -1 & \text{if } i = j, \\ 1 & \text{otherwise,} \end{cases}$$

for $i, j = 1, \dots, m$, where $u_{i,j}$ denotes the j-th component of u_i. Second, given an enumeration $L_1, \dots, L_{2^r}$ of all subsets of the set $\{1, \dots, r\}$, we define $V = \{v_1, \dots, v_r\}$ by

$$v_{k,j} = \begin{cases} -1 & \text{if } k \in L_j, \\ 1 & \text{otherwise,} \end{cases}$$

for $k = 1, \dots, r$ and $j = 1, \dots, 2^r$. Then the set

$$S = \{u_i : i = 1, \dots, m\} \times \{v_k : k = 1, \dots, r\}$$

obviously has cardinality $m \cdot r$.

To verify that S is shattered by $\mathcal{C}$, assume that some dichotomy (S_0, S_1) of S is given. We denote the $m + 2^r$ input variables of the units in $\mathcal{C}$ by $x_1, \dots, x_m, y_1, \dots, y_{2^r}$ such that $x_1, \dots, x_m$ receive inputs from U and $y_1, \dots, y_{2^r}$ receive inputs from V. We construct monomials $M_1, \dots, M_{2^r}$ in these variables as follows: Let the function $h : \{1, \dots, m\} \to \{1, \dots, 2^r\}$ satisfy

$$L_{h(i)} = \{k : u_i v_k \in S_1\},$$

where $u_i v_k$ is the vector resulting from the concatenation of u_i and v_k. Clearly, h is well-defined. Then we build the monomials by defining

$$M_j = y_j \cdot \prod_{i:h(i)=j} x_i$$

for $j = 1, \dots, 2^r$. Obviously, every variable occurs in at most one monomial and with exponent 1. Hence, the function $f : \mathbb{R}^{m+2^r} \to \mathbb{R}$ with

$$f(x_1, \dots, x_m, y_1, \dots, y_{2^r}) = M_1 + M_2 + \dots + M_{2^r}$$

can be computed by a member of $\mathcal{C}$. We claim that the function $\operatorname{sgn} \circ f$ induces the dichotomy (S_0, S_1).

Let $u_i v_k$ be some element of S. Since k occurs in exactly half of the sets $L_1, \dots, L_{2^r}$, we have from the definition of v_k that

$$v_{k,1} + v_{k,2} + \cdots + v_{k,2^r} = 0.$$

If $u_i v_k \in S_0$ then $k \notin L_{h(i)}$ according to the definition of h. Then v_k satisfies $v_{k,h(i)} = 1$. Since $u_{i,i}$ is the only component of u_i with value -1 and x_i occurs only in monomial $M_{h(i)}$, we have $M_{h(i)}(u_i v_k) = -1$ and thus $f(u_i v_k) = -2$.

On the other hand, if $u_i v_k \in S_1$ then $k \in L_{h(i)}$ and $v_{k,h(i)} = -1$. Now $M_{h(i)}(u_i v_k) = 1$ implies $f(u_i v_k) = 2$. Thus, (S_0, S_1) is induced by $\operatorname{sgn} \circ f$. □

We can now derive a lower bound for the class of higher-order units where the number of occurrences of the variables is restricted. The result implies that the bound $O(nl \log(dnl))$, where n is the number of variables, l the number of occurrences, and d the largest degree, given in Theorem 17 is asymptotically optimal with respect to n. Moreover, this optimality even holds if each variable is allowed to occur at most once and with exponent 1. By $\lfloor x \rfloor$ we denote the largest integer less or equal to x.

Corollary 21. *Suppose $\mathcal{C}$ is the class of higher-order units with n variables, where each variable occurs in at most one monomial and, if so, with exponent 1. Then the VC dimension of $\mathcal{C}$ is at least $\lfloor n/2 \rfloor \cdot \lfloor \log(n/2) \rfloor$.*

Proof. If $m = \lfloor n/2 \rfloor$ and $r = \lfloor \log(n/2) \rfloor$ then $m + 2^r \leq n$. Hence, Theorem 20 shows that there exists a set $S \subseteq \mathbb{R}^n$ of cardinality $m \cdot r = \lfloor n/2 \rfloor \cdot \lfloor \log(n/2) \rfloor$ that is shattered by $\mathcal{C}$, even if each variable has exponent 1. □

The next result paves the way to a lower bound for the class of higher-order units with a limited number of monomials.

Theorem 22. *Let $m, r \geq 1$ be natural numbers. Suppose $\mathcal{C}$ is the class of higher-order units with $m + 2r$ variables such that each unit consists of at most 2^r monomials and each variable occurs with exponent 1 only. Then there is a set of cardinality $m \cdot 2^r$ that is shattered by $\mathcal{C}$.*

Proof. We construct $S \subseteq \{-1, 0, 1\}^{m+2r}$ as the direct product of two sets $U \subseteq \{-1, 1\}^m$ and $V \subseteq \{0, 1\}^{2r}$. As in the previous proof we define $U = \{u_1, \dots, u_m\}$ with $u_{i,j}$, the j-th component of u_i, being

$$u_{i,j} = \begin{cases} -1 & \text{if } i = j, \\ 1 & \text{otherwise,} \end{cases}$$

for $i, j = 1, \dots, m$. For the definition of $V = \{v_1, \dots, v_{2^r}\}$ let $L_1, \dots, L_{2^r}$ be an enumeration of all subsets of the set $\{1, \dots, r\}$. Then the components of v_k are defined by

$$v_{k,j} = \begin{cases} 1 & \text{if } j \in L_k, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad v_{k,r+j} = \begin{cases} 0 & \text{if } j \in L_k, \\ 1 & \text{otherwise,} \end{cases}$$

for $k = 1, \ldots, 2^r$ and $j = 1, \ldots, r$. Clearly, the set

$$S = \{u_i : i = 1, \ldots, m\} \times \{v_k : k = 1, \ldots, 2^r\}$$

has cardinality $m \cdot 2^r$. It remains to show that $\mathcal{C}$ shatters S.

Let (S_0, S_1) be some arbitrary dichotomy of S. Denote the input variables by $x_1, \ldots, x_m, y_1, \ldots, y_{2r}$ such that $x_1, \ldots, x_m$ and $y_1, \ldots, y_{2r}$ receive inputs from U and V, respectively. First, we define monomials $N_1, \ldots, N_{2^r}$ in the variables $y_1, \ldots, y_{2r}$ by

$$N_k = \prod_{j \in L_k} y_j \cdot \prod_{j \notin L_k} y_{r+j}$$

for $k = 1, \ldots, 2^r$. Next, we use them to construct monomials $M_1, \ldots, M_{2^r}$ defined by

$$M_k = N_k \cdot \prod_{i : u_i v_k \in S_1} x_i$$

for $k = 1, \ldots, 2^r$, where $u_i v_k$ denotes the concatenation of u_i and v_k. Then, the function $f : \mathbb{R}^{m+2r} \to \mathbb{R}$ with

$$f(x_1, \ldots, x_m, y_1, \ldots, y_{2r}) = -M_1 - \cdots - M_{2^r}$$

can clearly be computed by a higher-order unit in $\mathcal{C}$. We show that (S_0, S_1) is induced by $\operatorname{sgn} \circ f$. Let $u_i v_k$ be some element of S. The definitions of v_k and the monomials $N_1, \ldots, N_{2^r}$ imply that

$$N_l(u_i v_k) = \begin{cases} 1 & \text{if } l = k, \\ 0 & \text{otherwise,} \end{cases}$$

for $l = 1, \ldots, 2^r$. Hence, we have

$$f(u_i v_k) = -M_k(u_i v_k) = -\prod_{h : u_h v_k \in S_1} u_{i,h},$$

which together with the definition of u_i implies

$$f(u_i v_k) = \begin{cases} -1 & \text{if } u_i v_k \in S_0, \\ 1 & \text{if } u_i v_k \in S_1. \end{cases}$$

Thus, $\operatorname{sgn} \circ f$ induces the dichotomy (S_0, S_1) as claimed and, consequently, S is shattered by $\mathcal{C}$. □

Finally, we obtain a lower bound for the class of higher-order units with a restricted number of monomials. In particular, the result shows that the bound $O(nk \log d)$, where k is the number of monomials, obtained in Theorem 16 cannot be improved with respect to nk.

Corollary 23. *Suppose $\mathcal{C}$ is the class of higher-order units with n variables where each variable has exponent* 1 *and each unit consists of at most k monomials for some arbitrary k satisfying $k \leq 2^{n/4}$. Then $\mathcal{C}$ has VC dimension at least $\lfloor n/2 \rfloor \cdot \lfloor k/2 \rfloor$.*

Proof. Let r be the largest integer satisfying $2^r \leq k$. Clearly, then $2^r > \lfloor k/2 \rfloor$. If we choose $m = \lfloor n/2 \rfloor$ then $m + 2r \leq \lfloor n/2 \rfloor + 2 \log k \leq n$, and Theorem 22 implies that there is a set $S \subseteq \mathbb{R}^n$ of cardinality $m \cdot 2^r \geq \lfloor n/2 \rfloor \cdot \lfloor k/2 \rfloor$ that is shattered by the class of higher-order units with at most $2^r \leq k$ monomials where each variable has exponent 1. □

We conclude by observing that the previous result also yields a lower bound for the class of higher-order units with restricted fan-out of the input nodes since, clearly, each variable occurs in at most k monomials.

5.2 Networks with Product Units

Multiplication is certainly the simplest type of arithmetical operation that can be performed by a product unit. All weights just need to be set to 1. Koiran and Sontag (1997) show that there exist networks consisting of linear and multiplication units that have VC dimension quadratic in the number of weights. Hence, this bound remains valid when product units are used instead of multiplication units, and Corollary 1 of Koiran and Sontag (1997) implies that for every W there is a network with $O(W)$ weights that consists only of linear and product units and has VC dimension W^2. This lower bound is based on the use of networks with unrestricted depth.

An extension of the result of Koiran and Sontag (1997) is obtained by Bartlett et al. (1998) who give a lower bound for layered sigmoidal networks in terms of the number of weights and the number of layers. Using the the constructions of Koiran and Sontag (1997) and Bartlett et al. (1998) in terms of linear and multiplication units we deduce that for every L and sufficiently large W there is a network with L layers and $O(W)$ weights that consists only of linear and product units and has VC dimension at least $\lfloor L/2 \rfloor \cdot \lfloor W/2 \rfloor$.

Thus, in terms of the number of weights we have a quadratic lower bound for arbitrary networks and a linear lower bound for networks of constant-depth. It is known, however, that networks of summing units can have constant depth *and* superlinear VC dimension. For threshold units such networks have been constructed by Sakurai (1993) and Maass (1994). We show now that also product unit networks of constant depth can have a superlinear VC dimension. In particular, we establish this for networks consisting of product and linear units and having two hidden layers. The numbering of the hidden layers in the following statement is done from the input nodes toward the output node.

Theorem 24. *Let n, k be natural numbers satisfying $k \leq 2^{n+2}$. There is a network $\mathcal{N}$ with the following properties: It has n input nodes, at most k hidden nodes arranged in two layers with product units in the first hidden layer and linear units in the second, and a product unit as output node; furthermore, $\mathcal{N}$ has $2n\lfloor k/4 \rfloor$ adjustable and $7\lfloor k/4 \rfloor$ fixed weights. The VC dimension of $\mathcal{N}$ is at least $(n - \lfloor \log(k/4) \rfloor) \cdot \lfloor k/8 \rfloor \cdot \lfloor \log(k/8) \rfloor$.*

With the aim of proving this we first establish a lemma in which we introduce a new kind of summing unit and make use of a property of sets of vectors. A set of m vectors in $\mathbb{R}^n$ is said to be *in general position* if every subset of at most n vectors is linearly independent. Obviously, a set in general position can be constructed for any m and n. The new summing unit has weights and a threshold as parameters and computes its output by applying the activation function $\tau(y) = 1 + 1/\cosh(y)$ to the weighted sum. This function has its maximum at $y = 0$ with $\tau(0) = 2$ and satisfies $\lim \tau(y) = 1$ for $y \to -\infty$ as well as for $y \to \infty$. Further, $\tau(y) \geq 1$ always holds.

Lemma 25. *Let h, m, r be arbitrary natural numbers. Suppose $\mathcal{N}$ is a network with $m + r$ input nodes, one hidden layer of $h + 2^r$ nodes which are summing units with activation function $1+1/\cosh$, and a monomial as output node. Then there is a set of cardinality $h \cdot m \cdot r$ that is shattered by $\mathcal{N}$.*

Proof. The construction is based on methods due to Sakurai and Yamasaki (1992) and Sakurai (1993). We choose a set $\{s_1, \ldots, s_{h \cdot m}\} \subseteq \mathbb{R}^m$ in general position and let $e_1, \ldots, e_r$ be the unit vectors in $\mathbb{R}^r$, that is, they have a 1 in exactly one component and 0 elsewhere. Clearly then, the set

$$S \quad = \quad \{s_i : i = 1, \ldots, h \cdot m\} \times \{e_j : j = 1, \ldots, r\}$$

is a subset of $\mathbb{R}^{m+r}$ with cardinality $h \cdot m \cdot r$. We show that it can be shattered by the network $\mathcal{N}$ as claimed.

Assume that (S_0, S_1) is a dichotomy of S. Let $L_1, \ldots, L_{2^r}$ be an enumeration of all subsets of the set $\{1, \ldots, r\}$ and define the function $g : \{1, \ldots, h \cdot m\} \to \{1, \ldots, 2^r\}$ to satisfy

$$L_{g(i)} \quad = \quad \{j : s_i e_j \in S_1\},$$

where $s_i e_j$ denotes the concatenated vectors s_i and e_j. For $l = 1, \ldots, 2^r$ let $R_l \subseteq \{s_1, \ldots, s_{h \cdot m}\}$ be the set

$$R_l \quad = \quad \{s_i : g(i) = l\}.$$

For each R_l we use $\lceil |R_l|/m \rceil$ hidden nodes of which we define the weights as follows: We partition R_l into $\lceil |R_l|/m \rceil$ subsets $R_{l,p}, p = 1, \ldots, \lceil |R_l|/m \rceil$, each of which has cardinality m, except for possibly one set of cardinality less than m.

For each subset $R_{l,p}$ there exist real numbers $w_{l,p,1}, \ldots, w_{l,p,m}, t_{l,p}$ such that every $s_i \in \{s_1, \ldots, s_{h \cdot m}\}$ satisfies

$$(w_{l,p,1}, \ldots, w_{l,p,m}) \cdot s_i - t_{l,p} = 0 \quad \text{if and only if} \quad s_i \in R_{l,p}. \tag{6}$$

This follows from the fact that the set $\{s_1, \ldots, s_{h \cdot m}\}$ is in general position. (In other words, $(w_{l,p,1}, \ldots, w_{l,p,m}, t_{l,p})$ represents the hyperplane passing through all points in $R_{l,p}$ and through none of the other points.[7]) With subset $R_{l,p}$ we associate a hidden node with threshold $t_{l,p}$ and with weights $w_{l,p,1}, \ldots, w_{l,p,m}$ for the connections from the first m input nodes. Since of all subsets $R_{l,p}$ at most h have cardinality m and at most 2^r have cardinality less than m, this construction can be done with at most $h + 2^r$ hidden nodes.

Thus far, we have specified the weights for the connections outgoing from the first m input nodes. The connections from the remaining r input nodes are weighted as follows: Let $\varepsilon > 0$ be a real number such that for every $s_i \in \{s_1, \ldots, s_{h \cdot m}\}$ and every weight vector $(w_{l,p,1}, \ldots, w_{l,p,m}, t_{l,p})$:

$$\text{If} \;\; s_i \notin R_{l,p} \;\; \text{then} \;\; |(w_{l,p,1}, \ldots, w_{l,p,m}) \cdot s_i - t_{l,p}| > \varepsilon.$$

According to the construction of the weight vectors in (6), such an ε clearly exists. We define the remaining weights $w_{l,p,m+1}, \ldots, w_{l,p,m+r}$ by

$$w_{l,p,m+j} \;=\; \begin{cases} 0 & \text{if } j \in L_l, \\ \varepsilon & \text{otherwise.} \end{cases} \tag{7}$$

This completes the definition of the hidden nodes. We show that they have the following property:

Claim. *If $s_i e_j \in S_1$ then there is exactly one hidden node with output value 2; if $s_i e_j \in S_0$ then all hidden nodes yield an output value less than 2.*

In order to establish this we observe that according to (6) there is exactly one weight vector $(w_{l,p,1}, \ldots, w_{l,p,m}, t_{l,p})$, where $l = g(i)$, that yields 0 on s_i. If $s_i e_j \in S_1$ then $j \in L_{g(i)}$, which together with (7) implies that the weighted sum $(w_{l,p,m+1}, \ldots, w_{l,p,m+r}) \cdot e_j$ is equal to 0. Hence, this node gets the total weighted sum 0 and, applying $1 + 1/\cosh$, outputs 2. The input vector e_j changes the weighted sums of the other nodes by an amount of at most ε. Thus, the total weighted sums for these nodes remain different from 0 and, hence, the output values are less than 2.

---

[7] If $|R_{l,p}| = m$ then this hyperplane is unique. In case $|R_{l,p}| < m$ we select one of the hyperplanes containing $R_{l,p}$ and none of the other points. This can be done, e.g., by extending the unique $(|R_{l,p}| - 1)$-dimensional hyperplane determined by $R_{l,p}$ to an appropriate $(m - 1)$-dimensional hyperplane.

On the other hand, if $s_i e_j \in S_0$ then $j \notin L_{g(i)}$ and the node that yields 0 on s_i receives an additional amount ε through weight $w_{l,p,m+j}$. This gives a total weighted sum different from 0 and an output value less than 2. All other nodes fail to receive 0 by an amount of more than ε and thus have total weighted sum different from 0 and, hence, an output value less than 2. Thus the claim is proven.

Finally to complete the proof, we do one more modification with the weight vectors and define the weights for the ouptut node. Clearly, if we multiply all weights and thresholds defined thus far with any real number $\alpha > 0$ the claim above remains true. Since $\lim(1 + 1/\cosh(y)) = 1$ for $y \to -\infty$ and $y \to \infty$, we can find an α such that on every $s_i e_j \in S$ the output values of those hidden nodes that do not output 2 multiplied together yield a value as close to 1 as necessary. Further, this value is at least 1, since $1 + 1/\cosh(y) \geq 1$ for all y. Thus, if we employ a monomial with all exponents equal to 1 for the output node, it follows from the reasoning above that the output value of the network is at least 2 if and only if $s_i e_j \in S_1$. This shows that S is shattered by $\mathcal{N}$. □

We now employ the previous result and give a proof of Theorem 24.

Proof of Theorem 24. The idea is to take a set S' constructed as in Lemma 25 and, as shown there, shattered by a network $\mathcal{N}'$ with a monomial as output node and one hidden layer of summing units that use the activation function $1 + 1/\cosh$. Then S' is transformed into a set S and $\mathcal{N}'$ into a network $\mathcal{N}$ such that for every dichotomy (S'_0, S'_1) induced by $\mathcal{N}'$ on S' the network $\mathcal{N}$ induces the corresponding dichotomy (S_0, S_1) of S.

Assume that n and k are given as supposed and let S' be the set defined in Lemma 25 choosing $h = \lfloor k/8 \rfloor$, $m = n - \lfloor \log(k/4) \rfloor$, and $r = \lfloor \log(k/8) \rfloor$. Note that the assumption $k \leq 2^{n+2}$ ensures that $m \geq 0$. Then S' has cardinality

$$m \cdot h \cdot r \quad = \quad (n - \lfloor \log(k/4) \rfloor) \cdot \lfloor k/8 \rfloor \cdot \lfloor \log(k/8) \rfloor.$$

Furthermore, we have $m + r = n - 1$ and hence $S' \subseteq \mathbb{R}^{n-1}$, and Lemma 25 implies that S' is shattered by a network $\mathcal{N}'$ with $n-1$ input nodes, a monomial as output node and one hidden layer of $h + 2^r \leq \lfloor k/4 \rfloor$ summing units with activation function $1 + 1/\cosh$. From S' we construct $S \subseteq \mathbb{R}^n$ by defining

$$S \quad = \quad \{(e^{s'_1}, \ldots, e^{s'_{n-1}}, e) : (s'_1, \ldots, s'_{n-1}) \in S'\}.$$

In other words, S is obtained from S' by appending a component containing 1 to each vector and applying the function $y \mapsto \exp(y)$ to every component. On some input vector $s' \in S'$ a hidden node of $\mathcal{N}'$ with weight vector w and threshold t computes

$$1 + \frac{1}{\cosh(w \cdot s' - t)} \quad = \quad \frac{\exp(w \cdot s' - t) - \exp(-w \cdot s' + t) + 2}{\exp(w \cdot s' - t) + \exp(-w \cdot s' + t)}. \tag{8}$$

If $s = (s_1, \ldots, s_n)$ is the vector in S obtained from the (unique) vector $s' = (s'_1, \ldots, s'_{n-1})$ in S' then according to the construction of S

$$(s'_1, \ldots, s'_{n-1}, 1) \quad = \quad (\ln(s_1), \ldots, \ln(s_{n-1}), \ln(s_n)),$$

which implies that an exponential in the right-hand side of (8) with weights w and threshold t yields on input vector s' the same output as a product unit with weights w, t on input vector s. (It is clear now that the reason for appending one component to the vectors in S' was to accommodate the threshold t as a weight in a product unit.) Therefore, the computation of a summing unit with activation function $1 + 1/\cosh$ on $s' \in S'$ can be simulated by feeding the vector $s \in S$ into a network with two hidden layers, where the first layer consists of two product units, the second layer has two linear units, and the output node computes a division. Furthermore, this network of 4 hidden nodes has $2n$ connections with adjustable weights and 7 connections with fixed weights (two for each linear unit, one for the threshold of the linear unit computing the numerator, and two for the division).

Replacing all $\lfloor k/4 \rfloor$ hidden nodes of $\mathcal{N}'$ in this way we obtain the network $\mathcal{N}$ which has at most k hidden nodes arranged in two layers where the first hidden layer consists of product units and the second of linear units. The output node has to compute a product of divisions which can be done by a single product unit. Further, $\mathcal{N}$ has $2n\lfloor k/4 \rfloor$ adjustable and $7\lfloor k/4 \rfloor$ fixed weights. Thus, $\mathcal{N}$ has the properties as claimed and shatters the set S which has the same cardinality as S'. $\square$

From the previous result we derive the following more simplified statement of a superlinear lower bound.

Corollary 26. *Let n, k be natural numbers where $16 \leq k \leq 2^{n/2+2}$. There is a network of product and linear units with n input units, at most k hidden nodes in two layers, and at most nk weights that has VC dimension at least $(nk/32)\log(k/16)$.*

Proof. The network constructed in the proof of Theorem 24 has $2n\lfloor k/4 \rfloor + 7\lfloor k/4 \rfloor \leq nk/2 + 2k$ weights, which are, using $2 \leq n/2$ from the assumptions, not more than nk weights. The VC dimension of this network was shown to be at least

$$(n - \lfloor \log(k/4) \rfloor) \cdot \lfloor k/8 \rfloor \cdot \lfloor \log(k/8) \rfloor.$$

Now, $k \leq 2^{n/2+2}$ implies $n - \lfloor \log(k/4) \rfloor \geq n/2$, from $k \geq 16$ we get $\lfloor k/8 \rfloor \geq k/8 - 1 \geq k/16$, and at last we use $\lfloor \log(k/8) \rfloor \geq \log(k/8) - 1 = \log(k/16)$. $\square$

Architecture	Class/Single	Unit Types	Bound	Remarks	Reference
general feed-forward	class	product and sigmoidal	$4(Wk)^2 + O(Wk\log(Wk))$	unit type variable	Corollary 12
	single	higher-order sigmoidal	$9(W^2k)^2 + O(W^2k\log(W^2k))$	exponents as parameters	Theorem 14
			$36(Wk)^4 + O((Wk)^2\log(Wk))$	exponents fixed	Theorem 13
two hidden layers	single	product and summing	$\Omega(W\log k)$	summing units in second hidden layer	Corollary 26
one hidden layer	single	product and summing	$(Wk)^2 + O(Wk\log W)$	k hidden nodes	Corollary 7
	class	product and summing	$4(nl)^4 + O((nl)^2\log(nl))$	input nodes fan-out $\leq l$	Corollary 9
single unit	class	higher-order	$O((nl)^4), O(n^2l\log d), O(nl\log dnl)$	input nodes fan-out $\leq l$, exponents $\leq d$	Theorem 17
			$\Omega(nl)$	input nodes fan-out $\leq l$, exponents 1	Corollary 23
			$\Omega(n\log n)$	input nodes fan-out 1, exponents 1	Corollary 21
			$O(n^2k^4), O(nk\log d)$	k monomials, exponents $\leq d$	Theorem 16
			$\Omega(nk)$	k monomials, exponents 1	Corollary 23
	single	product	n		Theorem 18
	class	monomial	n		Corollary 19

Table 1: A survey of the results. If not otherwise stated, W, k, and n refer to the number of parameters, computation nodes, and input nodes, respectively. Upper bounds are valid for the pseudo dimension, lower bounds for the VC dimension.

6 Summary and Conclusions

Multiplication is an arithmetical operation that when used in neural networks certainly helps to increase their computational power by allowing neural inputs to interact nonlinearly. The question is how this gain is reflected in quantitative measures of complexity and of, in particular, analog computational power. In this article we have dealt with two such measures: the Vapnik-Chervonenkis dimension and the pseudo dimension. We have derived upper and lower bounds on these dimensions for neural networks in which multiplication occurs as a fundamental operation in the interaction of network elements. An overview of the results is given in Table 1, where we present the bounds mainly in asymptotic form, abstracting from most of the constant factors.

The bounds are given in terms of the numbers of network parameters and computation nodes and, for classes, in terms of the restrictions that characterize the architectures in the respective class. We would like to highlight two features: First, the upper bounds are all polynomials of low order. In particular, the bound for general feedforward networks exhibits the same order of magnitude as the best known upper bound for purely sigmoidal networks. And this even in the case when it is not predetermined whether a node is to become a summing or a product unit. Second, the upper bounds for higher-order networks and some of the bounds for classes of higher-order units do not involve any constraint on the order. It is therefore impossible to find lower bounds that exhibit a growth in terms of the order only. This limitation is also indicated by the fact that some lower bounds for classes of higher-order units are already tight for order one. In this case, the degree of multiplicativity cannot help in proving better lower bounds. In general, the results show that multiplication in neural networks does not lead to an immeasurable growth of VC and pseudo dimension.

In practical uses of artificial neural networks, such as, for instance, in pattern recognition, higher-order networks and product unit networks are considered as natural extensions of the classical linear summing networks. We have reported about some applications where learning algorithms have been designed for training multiplicative networks. The question, how well networks resulting from these algorithms generalize is theoretically studied in numerous models of learning. In a major part of them the VC dimension and the pseudo dimension play a central role. They can be used to estimate the number of training examples required by learning algorithms to generate hypotheses with low generalization error. The results given here imply that estimates can now be given for higher-order sigmoidal networks that do not come with an a priori restriction of their order. Hence, one need not cope with a sample complexity that grows with the order. For learning applications this suggests the use of higher-order networks without any limit on the order. Further, the estimates are valid for a class of neural network learning algorithms that has yet to be developed: They hold even if the algorithm is allowed to decide for each node whether it is to be a summing or a product unit.

Apart from applications of learning, multiplicative neural networks are used for modeling the behavior of biological nervous systems or parts thereof. In this context questions arise as to what type of functions have been captured in a model that has been constructed in accordance with experimental observations. The VC dimension and the pseudo dimension are combinatorial measures for the complexity and diversity of function classes. As such they can be used to compare networks with respect to their expressiveness. Moreover, using upper bounds on these dimensions, lower bounds on the size of networks for the computation and approximation of functions can be calculated. By means of the results given here, such calculations can now be done for multiplicative neural networks. Thus, a new tool is available for the assessment of these networks and for the verification of their proper use in neural modeling.

Our investigations naturally raise some new questions. Most prominently, since also an open problem for networks of sigmoidal units, is the issue whether significantly better upper bounds can be shown for networks of fixed depth. The bounds for depth-restricted networks established so far coincide with the bounds for general feedforward networks. For the latter, however, quadratic lower bounds have been derived using a method that does not apply to constant-depth networks. Thus, the gap between upper and lower bound for depth-restricted networks is larger than in the general feedforward case.

The so-called fat-shattering dimension is a further combinatorial measure that is known to give bounds on the complexity of learning. Since it is bounded from above by the pseudo dimension, the results in this article imply upper bounds on the fat-shattering dimension. Moreover, when the output node is a linear unit, the fat-shattering dimension is equal to the pseudo dimension. It is an interesting question whether for networks with nonlinear output nodes bounds can be obtained for the fat-shattering dimension that are significantly smaller than the pseudo dimension of the network.

The lower bounds we presented are all derived for the VC dimension and, hence, are by definition also valid for the pseudo dimension. It is currently not known how to obtain lower bounds for the pseudo dimension of neural networks directly.

Finally, our calculations resulted in several constants appearing in the bounds. We did not strive for obtaining optimal values but were content with the constants being small. Certainly, improvements might be possible using more detailed calculations or new approaches.

Appendix

A Solution Set Components Bounds

In the follwing we give the proofs of the Lemmas required for the upper bounds in Section 4.

Lemma 4. *Let $\mathcal{G}$ be the class of polynomials of degree at most p in the variables $y_1, \ldots, y_d$ and in the exponentials $e^{g_1}, \ldots, e^{g_q}$, where $g_1, \ldots, g_q$ are fixed affine functions in $y_1, \ldots, y_d$. Then $\mathcal{G}$ has solution set components bound*

$$B \quad = \quad 2^{q(q-1)/2}[p(p+1)d+p]^d[(p+1)d(d+1)+1]^q.$$

Proof. Consider for $1 \leq k \leq d$ an arbitrary set $\{f_1, \ldots, f_k\} \subseteq \mathcal{G}$ that has regular zero-set intersections and let p_i be the degree of f_i. It follows from Khovanskiĭ (1991), p. 91, Corollary 3, that if l is the dimension of the set

$$\{a \in \mathbb{R}^d : f_1(a) = \cdots = f_k(a) = 0\}$$

then this set has at most $2^{q(q-1)/2}p_1 \cdots p_k S^l[(l+1)S-l]^q$ connected components where $S = \sum_{i=1}^{k} p_i + l + 1$. From $p_i \leq p$, $k \leq d$, and $l \leq d$ we get $S \leq (p+1)d+1$ and $(l+1)S - l \leq (p+1)d(d+1)+1$, which implies the result. $\square$

Lemma 5. *Let q be a natural number and suppose $\mathcal{G}$ is the class of real-valued functions in the variables $y_1, \ldots, y_d$ satisfying the following conditions: For every $f \in \mathcal{G}$ there exist affine functions $g_1, \ldots, g_r$, where $r \leq q$, in the variables $y_1, \ldots, y_d$ such that f is an affine combination of $y_1, \ldots, y_d$ and $e^{g_1}, \ldots, e^{g_r}$. Then $\mathcal{G}$ has solution set components bound*

$$B \quad = \quad 2^{dq(dq-1)/2}[2(dq+d)+1]^{dq+d}[2(dq+d)(dq+d+1)+1]^{dq}.$$

Proof. Let $k \leq d$ and consider some arbitrary set $\{f_1, \ldots, f_k\} \subseteq \mathcal{G}$ that has regular zero-set intersections. According to the assumptions, for $i = 1, \ldots, k$ each f_i can be written in the form

$$(y_1, \ldots, y_d) \mapsto a_i + b_{i,1}y_1 + \cdots + b_{i,d}y_d + c_{i,1}e^{g_{i,1}} + \cdots + c_{i,r_i}e^{g_{i,r_i}},$$

where $r_i \leq q$, the $g_{i,j}$ are affine functions in $y_1, \ldots, y_d$, and a_i, $b_{i,1}, \ldots, b_{i,d}$, $c_{i,1}, \ldots, c_{i,r_i}$ are real numbers. We introduce new functions $\tilde{f}_i$ and $h_{i,j}$ in $y_1, \ldots, y_d$ and in new variables $z_{i,j}$ by defining

$$\begin{aligned} \tilde{f}_i(y_1, \ldots, y_d, z_{i,1}, \ldots, z_{i,r_i}) &= a_i + b_{i,1}y_1 + \cdots + b_{i,d}y_d + c_{i,1}e^{z_{i,1}} + \cdots + c_{i,r_i}e^{z_{i,r_i}}, \\ h_{i,j}(y_1, \ldots, y_d, z_{i,j}) &= g_{i,j}(y_1, \ldots, y_d) - z_{i,j}, \end{aligned}$$

for $i = 1, \ldots, k$ and $j = 1, \ldots, r_i$. Let $\tilde{\mathcal{G}}$ be the class of affine functions in $y_1, \ldots, y_d$, and in $z_{i,j}$ and $e^{z_{i,j}}$, for $i = 1, \ldots, k$ and $j = 1, \ldots, q$. Clearly, the functions $\tilde{f}_i$ and $h_{i,j}$ are elements of $\tilde{\mathcal{G}}$. Furthermore, since $f_1, \ldots, f_k$ are chosen arbitrarily from $\mathcal{G}$ and at most q new variables are introduced for each f_i, the classes $\mathcal{G}$ and $\tilde{\mathcal{G}}$ satisfy Definition 7.12 of Anthony and Bartlett (1999), that is, $\tilde{\mathcal{G}}$ computes $\mathcal{G}$ with q intermediate variables. In particular, the partial derivative of $h_{i,j}$ with respect to the variable $z_{i,j}$ is -1, and hence non-zero. Thus, the derivative condition (iii) of the aforementioned Definition 7.12 is met. It follows from Theorem 7.13 of Anthony and Bartlett (1999) that any solution set components bound for $\tilde{\mathcal{G}}$ is also a solution set components bound for $\mathcal{G}$. Since $\tilde{\mathcal{G}}$ consists of polynomials of degree 1 in $dq + d$ variables and dq fixed exponentials, by virtue of Lemma 4 class $\tilde{\mathcal{G}}$ has solution set components bound

$$B \quad = \quad 2^{dq(dq-1)/2}[2(dq+d)+1]^{dq+d}[2(dq+d)(dq+d+1)+1]^{dq}$$

which is hence also a solution set components bound for $\mathcal{G}$ as claimed. $\square$

Lemma 6. *Let $\mathcal{F}$ be a class of real-valued functions $(y_1, \ldots, y_d, x_1, \ldots, x_n) \mapsto f(y_1, \ldots, y_d, x_1, \ldots, x_n)$ that is closed under addition of constants and where each function in $\mathcal{F}$ is C^d in the variables $y_1, \ldots, y_d$. If the class $\mathcal{G} = \{(y_1, \ldots, y_d) \mapsto f(y_1, \ldots, y_d, s) : f \in \mathcal{F}, s \in \mathbb{R}^n\}$ has solution set components bound B then for any sets $\{f_1, \ldots, f_k\} \subseteq \mathcal{F}$ and $\{s_1, \ldots, s_m\} \subseteq \mathbb{R}^n$, where $m \geq d/k$, the set $T \subseteq \{0,1\}^{mk}$ defined as*

$$\begin{aligned} T = \{(&\operatorname{sgn}(f_1(a, s_1)), \ldots, \operatorname{sgn}(f_1(a, s_m)), \ldots \\ &\ldots, \operatorname{sgn}(f_k(a, s_1)), \ldots, \operatorname{sgn}(f_k(a, s_m))) : a \in \mathbb{R}^d\} \end{aligned}$$

satisfies

$$|T| \leq B \sum_{i=0}^{d} \binom{mk}{i} \leq B \left(\frac{emk}{d}\right)^d.$$

Proof. Let $\{f_1, \ldots, f_k\} \subseteq \mathcal{F}$ and $\{s_1, \ldots, s_m\} \subseteq \mathbb{R}^n$ be given, and T be defined as above. In the proof of Theorem 7.8 in Anthony and Bartlett (1999) it is shown that then there exist real numbers $\lambda_{1,1}, \ldots, \lambda_{k,m}$ such that the following holds: Let C denote the number of connected components of the set

$$\mathbb{R}^d - \bigcup_{i=1}^{k} \bigcup_{j=1}^{m} \{a \in \mathbb{R}^d : f_i(a, s_j) - \lambda_{i,j} = 0\}.$$

Then T satisfies $|T| \leq C$, and the set of functions

$$\{a \mapsto f_i(a, s_j) - \lambda_{i,j} : i = 1, \ldots, k; j = 1, \ldots, m\}$$

has regular zero-set intersections. Clearly, this set is a subset of $\mathcal{G}$, which has solution set components bound B. In the proof of Theorem 7.6 in Anthony and Bartlett (1999) it is shown that this implies

$$C \leq B \sum_{i=0}^{d} \binom{mk}{i} \leq B \left(\frac{emk}{d}\right)^d$$

for $m \geq d/k$. Hence, the claimed result follows using $|T| \leq C$. $\square$

Lemma 11. *Let $\mathcal{G}$ be the class of real-valued functions in d variables computed by a network with r computation nodes where q of these nodes compute one of the functions $\alpha \mapsto c + 1/(1 + e^{-\alpha})$, $\alpha \mapsto c \pm e^{\alpha}$, or $\alpha \mapsto \ln \alpha$ for some arbitrary constant c, and $r - q$ nodes compute some polynomial of degree 2. Then $\mathcal{G}$ has solution set components bound*

$$B \quad = \quad 2^{dq(dq-1)/2}[6(2dr + d) + 2]^{2dr+d}[3(2dr + d)(2dr + d + 1) + 1]^{dq}.$$

Proof. Let $\{f_1, \ldots, f_k\} \subseteq \mathcal{G}$, where $k \leq d$, be some arbitrary set of functions that has regular zero-set intersections. According to the assumptions, there is for each f_i a network that computes f_i with r and q computation nodes as described. We number the nodes such that the computation of each node depends only on nodes with a smaller number. Then for $i = 1, \ldots, k$ and $j = 1, \ldots, r$ the computation performed by node j in the network for f_i can be represented by a function $n_{i,j}$ in the variables $y_1, \ldots, y_d$ that is recursively defined by

$$n_{i,j}(y) \quad = \quad \begin{cases} c_{i,j} + 1/(1 + \exp(-n_{i,l}(y)), & l < j, \\ c_{i,j} \pm \exp(n_{i,l}(y)), & l < j, \\ \ln(n_{i,l}(y)), & l < j, \\ p_{i,j}(y, n_{i,1}(y), \ldots, n_{i,j-1}(y)), & \end{cases}$$

depending on whether node j computes the function $\alpha \mapsto c_{i,j} + 1/(1 + e^{-\alpha})$, $\alpha \mapsto c_{i,j} \pm e^{\alpha}$, $\alpha \mapsto \ln \alpha$, or the degree 2 polynomial $p_{i,j}$, respectively. We introduce new functions $g_{i,j}, \tilde{g}_{i,j}$ in $y_1, \ldots, y_d$ and in new variables $z_{i,j}, \tilde{z}_{i,j}$ corresponding to the above four cases for $n_{i,j}$ as follows: If node j in the network for f_i computes

(a) the function $\alpha \mapsto c_{i,j} + 1/(1 + e^{-\alpha})$ then

$$\begin{aligned} \tilde{g}_{i,j}(y, \tilde{z}_{i,j}) &= n_{i,l}(y) + \tilde{z}_{i,j}, \\ g_{i,j}(z_{i,j}, \tilde{z}_{i,j}) &= (z_{i,j} - c_{i,j})(1 + \exp(\tilde{z}_{i,j})) - 1, \end{aligned}$$

(b) the function $\alpha \mapsto c_{i,j} \pm e^{\alpha}$ then

$$\begin{aligned} \tilde{g}_{i,j}(y, \tilde{z}_{i,j}) &= n_{i,l}(y) - \tilde{z}_{i,j}, \\ g_{i,j}(z_{i,j}, \tilde{z}_{i,j}) &= \exp(\tilde{z}_{i,j}) \pm c_{i,j} - z_{i,j}, \end{aligned}$$

(c) the function $\alpha \mapsto \ln \alpha$ then

$$\begin{aligned} \tilde{g}_{i,j}(y, \tilde{z}_{i,j}) &= n_{i,l}(y) - \exp(\tilde{z}_{i,j}), \\ g_{i,j}(z_{i,j}, \tilde{z}_{i,j}) &= \tilde{z}_{i,j} - z_{i,j}, \end{aligned}$$

(d) the degree 2 polynomial $p_{i,j}$ then

$$g_{i,j}(y, z_{i,1}, \ldots, z_{i,j}) \quad = \quad p_{i,j}(y, z_{i,1}, \ldots, z_{i,j-1}) - z_{i,j},$$

where l, $c_{i,j}$, and $p_{i,j}$ are as in the corresponding definition of $n_{i,j}$ above. Let $\tilde{\mathcal{G}}$ be the class of polynomials of degree at most 2 in the variables $y_1, \ldots, y_d$ and in $z_{i,j}$, $\tilde{z}_{i,j}$ and $\exp(\tilde{z}_{i,j})$. There are at most kr variables $z_{i,j}$ and—since the variables $\tilde{z}_{i,j}$ are introduced only for those nodes that compute a non-polynomial function—at most kq variables $\tilde{z}_{i,j}$ and kq exponentials $\exp(\tilde{z}_{i,j})$. Clearly, $\mathcal{G}$ contains the functions $\tilde{g}_{i,j}$ and $g_{i,j}$, which implicitly define the variables $\tilde{z}_{i,j}$ and $z_{i,j}$, respectively. The partial derivative of $\tilde{g}_{i,j}$ with respect to $\tilde{z}_{i,j}$ in cases (a)–(c) is $1, -1$, and $-\exp(\tilde{z}_{i,j})$, respectively. For $g_{i,j}$ the partial derivative with respect to $z_{i,j}$ is $1 + \exp(\tilde{z}_{i,j})$ in case (a), and -1 in cases (b)–(d). All these partial derivatives are everywhere non-zero. Hence, condition (iii) of Definition 7.12 in Anthony and Bartlett (1999) is satisfied. Furthermore, Theorem 7.13 of Anthony and Bartlett (1999) implies that $\tilde{\mathcal{G}}$ computes $\mathcal{G}$ with $r+q$ intermediate variables and any solution set components bound for $\tilde{\mathcal{G}}$ is also a solution set components bound for $\mathcal{G}$. The polynomials in $\tilde{\mathcal{G}}$ are of degree at most 2 in no more than $2dr + d$ variables and dq fixed exponentials. Thus, Lemma 4 shows that $\tilde{\mathcal{G}}$ has solution set components bound

$$B \;=\; 2^{dq(dq-1)/2}[6(2dr+d)+2]^{2dr+d}[3(2dr+d)(2dr+d+1)+1]^{dq},$$

and we conclude that this is also a solution set components bound for $\mathcal{G}$. □

Acknowledgments

I thank the anonymous referees for helpful comments. This work has been supported in part by the ESPRIT Working Group in Neural and Computational Learning II, NeuroCOLT2, No. 27150. Some of the results have been presented at the NeuroCOLT Workshop "New Perspectives in the Theory of Neural Nets" in Graz, Austria, on May 3, 2000. I am grateful to Wolfgang Maass for the invitation to give a talk at this meeting.

References

Andersen, R. A., Essick, G. K., and Siegel, R. M. (1985). Encoding of spatial location by posterior parietal neurons. *Science*, 230:456–458.

Anthony, M. (1995). Classification by polynomial surfaces. *Discrete Applied Mathematics*, 61:91–103.

Anthony, M. and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge.

Anzai, A., Ohzawa, I., and Freeman, R. D. (1999a). Neural mechanisms for processing binocular information I. Simple cells. *Journal of Neurophysiology*, 82:891–908.

Anzai, A., Ohzawa, I., and Freeman, R. D. (1999b). Neural mechanisms for processing binocular information II. Complex cells. *Journal of Neurophysiology*, 82:909–924.

Bartlett, P. L., Maiorov, V., and Meir, R. (1998). Almost linear VC dimension bounds for piecewise polynomial networks. *Neural Computation*, 10:2159–2173.

Baum, E. B. and Haussler, D. (1989). What size net gives valid generalization? *Neural Computation*, 1:151–160.

Ben-David, S., Cesa-Bianchi, N., Haussler, D., and Long, P. M. (1995). Characterizations of learnability for classes of $\{0,\ldots,n\}$-valued functions. *Journal of Computer and System Sciences*, 50:74–86.

Ben-David, S. and Lindenbaum, M. (1998). Localization vs. identification of semi-algebraic sets. *Machine Learning*, 32:207–224.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford.

Blomfield, S. (1974). Arithmetical operations performed by nerve cells. *Brain Research*, 69:115–124.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36:929–965.

Bugmann, G. (1991). Summation and multiplication: Two distinct operation domains of leaky integrate-and-fire neurons. *Network: Computation in Neural Systems*, 2:489–509.

Bugmann, G. (1992). Multiplying with neurons: Compensation for irregular input spike trains by using time-dependent synaptic efficiencies. *Biological Cybernetics*, 68:87–92.

Burshtein, D. (1998). Long-term attraction in higher order neural networks. *IEEE Transactions on Neural Networks*, 9:42–50.

Carandini, M. and Heeger, D. J. (1994). Summation and division by neurons in primate visual cortex. *Science*, 264:1333–1336.

Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, 14:326–334.

Cover, T. M. (1968). Capacity problems for linear machines. In Kanal, L. N., editor, *Pattern Recognition*, pages 283–289, Thompson Book Co., Washington.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* Wiley, New York.

Durbin, R. and Rumelhart, D. (1989). Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Computation*, 1:133–142.

Fahner, G. and Eckmiller, R. (1994). Structural adaptation of parsimonious higher-order neural classifiers. *Neural Networks*, 7:279–289.

Feldman, J. A. and Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science*, 6:205–254.

Gabbiani, F., Krapp, H. G., and Laurent, G. (1999). Computation of object approach by a wide-field, motion-sensitive neuron. *Journal of Neuroscience*, 19:1122–1141.

Ghosh, J. and Shin, Y. (1992). Efficient higher-order neural networks for classification and function approximation. *International Journal of Neural Systems*, 3:323–350.

Giles, C. L. and Maxwell, T. (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, 26:4972–4978.

Giles, C. L., Miller, C. B., Chen, D., Chen, H. H., Sun, G. Z., and Lee, Y. C. (1992). Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4:393–405.

Giles, C. L., Sun, G. Z., Chen, H. H., Lee, Y. C., and Chen, D. (1990). Higher order recurrent networks and grammatical inference. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 380–387, Morgan Kaufmann, San Mateo, CA.

Goldberg, P. W. and Jerrum, M. R. (1995). Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, 18:131–148.

Hancock, T. R., Golea, M., and Marchand, M. (1994). Learning nonoverlapping Perceptron networks from examples and membership queries. *Machine Learning*, 16:161–183.

Hatsopoulos, N., Gabbiani, F., and Laurent, G. (1995). Elementary computation of object approach by a wide-field visual neuron. *Science*, 270:1000–1003.

Haussler, D. (1992). Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100:78–150.

Haussler, D., Kearns, M., and Schapire, R. E. (1994). Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. *Machine Learning*, 14:83–113.

Heywood, M. and Noakes, P. (1995). A framework for improved training of sigma-pi networks. *IEEE Transactions on Neural Networks*, 6:893–903.

Hofmeister, T. (1994). Depth-efficient threshold circuits for arithmetic functions. In Roychowdhury, V., Siu, K.-Y., and Orlitsky, A., editors, *Theoretical Advances in Neural Computation and Learning*, chapter 2, pages 37–84. Kluwer Academic Publishers, Boston, MA.

Ismail, A. and Engelbrecht, A. P. (2000). Global optimization algorithms for training product unit neural networks. In *International Joint Conference on Neural Networks IJCNN'2000*, volume I, pages 132–137, IEEE Computer Society, Los Alamitos, CA.

Janson, D. J. and Frenzel, J. F. (1993). Training product unit neural networks with genetic algorithms. *IEEE Expert*, 8(5):26–33.

Karpinski, M. and Macintyre, A. (1997). Polynomial bounds for VC dimension of sigmoidal and general Pfaffian neural networks. *Journal of Computer and System Sciences*, 54:169–176.

Karpinski, M. and Werther, T. (1993). VC dimension and uniform learnability of sparse polynomials and rational functions. *SIAM Journal on Computing*, 22:1276–1285.

Khovanskiĭ, A. G. (1991). *Fewnomials*, volume 88 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, RI.

Koch, C. (1999). *Biophysics of Computation*. Oxford University Press, New York.

Koch, C. and Poggio, T. (1992). Multiplying with synapses and neurons. In McKenna, T., Davis, J., and Zornetzer, S., editors, *Single Neuron Computation*, chapter 12, pages 315–345. Academic Press, Boston, MA.

Koch, C., Poggio, T., and Torre, V. (1983). Nonlinear interactions in a dendritic tree: Localization, timing, and role in information processing. *Proceedings of the National Academy of Sciences USA*, 80:2799–2802.

Koiran, P. (1996). VC dimension in circuit complexity. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity CCC'96*, pages 81–85, IEEE Computer Society Press, Los Alamitos, CA.

Koiran, P. and Sontag, E. D. (1997). Neural networks with quadratic VC dimension. *Journal of Computer and System Sciences*, 54:190–198.

Koiran, P. and Sontag, E. D. (1998). Vapnik-Chervonenkis dimension of recurrent neural networks. *Discrete Applied Mathematics*, 86:63–79.

Kowalczyk, A. and Ferrá, H. L. (1994). Developing higher-order networks with empirically selected units. *IEEE Transactions on Neural Networks*, 5:698–711.

Küpfmüller, K. and Jenik, F. (1961). Über die Nachrichtenverarbeitung in der Nervenzelle. *Kybernetik*, 1:1–6.

Lee, Y. C., Doolen, G., Chen, H. H., Sun, G. Z., Maxwell, T., Lee, H., and Giles, C. L. (1986). Machine learning using a higher order correlation network. *Physica D*, 22:276–306.

Leerink, L. R., Giles, C. L., Horne, B. G., and Jabri, M. A. (1995a). Learning with product units. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems 7*, pages 537–544, MIT Press, Cambridge, MA.

Leerink, L. R., Giles, C. L., Horne, B. G., and Jabri, M. A. (1995b). Product unit learning. Technical Report UMIACS-TR-95-80, University of Maryland.

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6:861–867.

Maass, W. (1994). Neural nets with super-linear VC-dimension. *Neural Computation*, 6:877–884.

Maass, W. (1995a). Agnostic PAC learning of functions on analog neural nets. *Neural Computation*, 7:1054–1078.

Maass, W. (1995b). Vapnik-Chervonenkis dimension of neural nets. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 1000–1003. MIT Press, Cambridge, MA.

Maass, W. (1997). Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In Mozer, M., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pages 211–217. MIT Press, Cambridge, MA.

Maass, W. (1998). A simple model for neural computation with firing rates and firing correlations. *Network: Computation in Neural Systems*, 9:381–397.

Maass, W. and Turán, G. (1992). Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9:107–145.

Maxwell, T., Giles, C. L., Lee, Y. C., and Chen, H. H. (1986). Nonlinear dynamics of artificial neural systems. In Denker, J. S., editor, *Neural Networks for Computing*, volume 151 of *AIP Conference Proceedings*, American Institute of Physics, New York.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Mel, B. W. (1992a). The clusteron: Toward a simple abstraction for a complex neuron. In Moody, J., Hanson, S., and Lippmann, R., editors, *Advances in Neural Information Processing Systems 4*, pages 35–42, Morgan Kaufmann, San Mateo, CA.

Mel, B. W. (1992b). NMDA-based pattern discrimination in a modeled cortical neuron. *Neural Computation*, pages 502–517.

Mel, B. W. (1993). Synaptic integration in an excitable dendritic tree. *Journal of Neurophysiology*, pages 1086–1101.

Mel, B. W. (1994). Information processing in dendritic trees. *Neural Computation*, 6:1031–1085.

Mel, B. W. and Koch, C. (1990). Sigma-pi learning: On radial basis functions and cortical associative learning. In Touretzky, D. S., editor, *Advances in Neural Information Processing 2*, pages 474–481, Morgan Kaufmann, San Mateo, CA.

Mel, B. W., Ruderman, D. L., and Archie, K. A. (1998). Translation-invariant orientation tuning in visual "complex" cells could derive from intradendritic computations. *Journal of Neuroscience*, pages 4325–4334.

Minsky, M. L. and Papert, S. A. (1988). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, expanded edition.

Nilsson, N. J. (1965). *Learning Machines*. McGraw-Hill, New York.

Olshausen, B. A., Anderson, C. H., and Van Essen, D. C. (1993). A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of Neuroscience*, pages 4700–4719.

Omlin, C. W. and Giles, C. L. (1996a). Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the Association for Computing Machinery*, 43:937–972.

Omlin, C. W. and Giles, C. L. (1996b). Stable encoding of large finite-state automata in recurrent neural networks with sigmoid discriminants. *Neural Computation*, 8:675–696.

Perantonis, S. J. and Lisboa, P. J. G. (1992). Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers. *IEEE Transactions on Neural Networks*, 3:241–251.

Pinkus, A. (1999). Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195.

Poggio, T. (1990). A theory of how the brain might work. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 55, pages 899–910, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.

Poggio, T. and Girosi, F. (1990a). Networks for approximation and learning. *Proceedings of the IEEE*, 78:1481–1497.

Poggio, T. and Girosi, F. (1990b). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982.

Pollack, J. B. (1991). The induction of dynamical recognizers. *Machine Learning*, 7:227–252.

Pouget, A. and Sejnowski, T. J. (1997). Spatial transformations in the parietal cortex using basis functions. *Journal of Cognitive Neuroscience*, 9:222–237.

Psaltis, D., Park, C. H., and Hong, J. (1988). Higher order associative memories and their optical implementations. *Neural Networks*, 1:149–163.

Rebotier, T. P. and Droulez, J. (1994). Sigma vs pi properties of spiking neurons. In Mozer, M., Smolensky, P., Touretzky, D., Elman, J., and Weigend, A., editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 3–10, Erlbaum, Hillsdale, NJ.

Redding, N. J., Kowalczyk, A., and Downs, T. (1993). Constructive higher-order network algorithm that is polynomial time. *Neural Networks*, 6:997–1010.

Ring, M. (1993). Learning sequential tasks by incrementally adding higher orders. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pages 115–122, Morgan Kaufmann Publishers, San Mateo, CA.

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Roy, A. and Mukhopadhyay, S. (1997). Iterative generation of higher-order nets in polynomial time using linear programming. *IEEE Transactions on Neural Networks*, 8:402–412.

Rumelhart, D. E., Hinton, G. E., and McClelland, J. L. (1986a). A general framework for parallel distributed processing. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 2, pages 45–76. MIT Press, Cambridge, MA.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986b). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge, MA.

Sakurai, A. (1993). Tighter bounds of the VC-dimension of three layer networks. In *Proceedings of the World Congress on Neural Networks*, volume 3, pages 540–543. Erlbaum, Hillsdale, NJ.

Sakurai, A. (1999). Tight bounds for the VC-dimension of piecewise polynomial networks. In Kearns, M. S., Solla, S. A., and Cohn, D. A., editors, *Advances in Neural Information Processing Systems 11*, pages 323–329, MIT Press, Cambridge, MA.

Sakurai, A. and Yamasaki, M. (1992). On the capacity of n-h-s networks. In Aleksander, I. and Taylor, J., editors, *Artificial Neural Networks*, volume 2, pages 237–240, Elsevier, Amsterdam.

Salinas, E. and Abbott, L. F. (1996). A model of multiplicative neural responses in parietal cortex. *Proceedings of the National Academy of Sciences USA*, 93:11956–11961.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:211–229.

Schläfli, L. (1901). *Theorie der vielfachen Kontinuität.* Zürcher & Furrer, Zürich. Reprinted in: Schläfli, L. (1950). *Gesammelte Mathematische Abhandlungen. Band I.* Birkhäuser, Basel.

Schmidt, W. A. C. and Davis, J. P. (1993). Pattern recognition properties of various feature spaces for higher order neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:795–801.

Schmitt, M. (1999). On the sample complexity for nonoverlapping neural networks. *Machine Learning*, 37:131–141.

Schmitt, M. (2000). Lower bounds on the complexity of approximating continuous functions by sigmoidal neural networks. In Solla, S. A., Leen, T. K., and Müller, K.-R., editors, *Advances in Neural Information Processing Systems 12*, pages 328–334, MIT Press, Cambridge, MA.

Sejnowski, T. J. (1986). Higher-order Boltzmann machines. In Denker, J. S., editor, *Neural Networks for Computing*, volume 151 of *AIP Conference Proceedings*, pages 398–403, American Institute of Physics, New York.

Shawe-Taylor, J. and Anthony, M. (1991). Sample sizes for multiple-output threshold networks. *Network: Computation in Neural Systems*, 2:107–117.

Shin, Y. and Ghosh, J. (1995). Ridge polynomial networks. *IEEE Transactions on Neural Networks*, 6:610–622.

Siu, K.-Y., Roychowdhury, V., and Kailath, T. (1995). *Discrete Neural Computation: A Theoretical Foundation*. Information and System Sciences Series. Prentice Hall, Englewood Cliffs, NJ.

Softky, W. and Koch, C. (1995). Single-cell models. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 879–884. MIT Press, Cambridge, MA.

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

Spirkovska, L. and Reid, M. B. (1994). Higher-order neural networks applied to 2D and 3D object recognition. *Machine Learning*, 15:169–199.

Srinivasan, M. V. and Bernard, G. D. (1976). A proposed mechanism for multiplication of neural signals. *Biological Cybernetics*, pages 227–236.

Suga, N. (1990). Cortical computational maps for auditory imaging. *Neural Networks*, 3:3–21.

Suga, N., Olsen, J. F., and Butman, J. A. (1990). Specialized subsystems for processing biologically important complex sounds: Cross-correlation analysis for ranging in the bat's brain. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 55, pages 585–597, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.

Tal, D. and Schwartz, E. L. (1997). Computing with the leaky integrate-and-fire neuron: Logarithmic computation and multiplication. *Neural Computation*, 9:305–318.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27:1134–1142.

Venkatesh, S. S. and Baldi, P. (1991a). Programmed interactions in higher-order neural networks: Maximal capacity. *Journal of Complexity*, 7:316–337.

Venkatesh, S. S. and Baldi, P. (1991b). Programmed interactions in higher-order neural networks: The outer-product algorithm. *Journal of Complexity*, 7:443–479.

Warren, H. E. (1968). Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, 133:167–178.

Watrous, R. L. and Kuhn, G. M. (1992). Induction of finite-state languages using second-order recurrent networks. *Neural Computation*, 4:406–414.

Williams, R. J. (1986). The logic of activation functions. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, chapter 10, pages 423–443. MIT Press, Cambridge, MA.

Yoon, H. and Oh, J.-H. (1998). Learning of higher-order perceptrons with tunable complexities. *Journal of Physics A: Math. Gen.*, 31:7771–7784.