Annales UMCS Informatica AI 2 (2004) 135-141

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

The computational power and complexity of discrete feedforward neural networks

Barbara Borowik[a*], Sophie Laird[b]

[a]*Institute of Computer Modelling, Cracow University of Technology, Warszawska 24, 31-155 Kraków, Poland*
[b]*College of Notre Dame , Baltimore, Maryland, USA*

Abstract

The number of binary functions that can be defined on a set of L vectors in $\Re^N$ equals 2^L. For $L > N$ the total number of threshold functions in N-dimensional space grows polynomially $\left(2^{N(N-1)}\right)$, while the total number of Boolean functions, definable on N binary inputs, grows exponentially (2^{2^N}), and as N increases a percentage of threshold functions in relation to the total number of Boolean functions – goes to zero. This means that for the realization of a majority of tasks a neural network must possess at least two or three layers. The examples of small computational problems are arithmetic functions, like multiplication, division, addition, exponentiation or comparison and sorting. This article analyses some aspects of two- and more than two layers of threshold and Boolean circuits (feedforward neural nets), connected with their computational power and node, edge and weight complexity.

1. Introduction

A feedforward neural network is often modelled as a directed acyclic graph in which every directed edge represents a weighted net connection and each internal or output node (i.e. a gate) – reflects the net respective processing element. The terms feedforward networks and circuits – are in literature used interchangeably. The same can be said about the terms: Boolean, binary and logical (while specifying inputs, functions, circuits or nets).

Functions most commonly computed by a gate – are nonlinear ones, like threshold functions (called also jump or linearly separable functions), sigmoidal, AND-OR or a parity (i.e. XOR) functions.

A linear threshold function[1] $f(X):\{0;1\}^N \to \{0;1\}$ – is a **binary** (i.e. Boolean) function, such that:

* Corresponding author: *e-mail address*: plborowi@cyf-kr.edu.pl

$$f(\boldsymbol{X}) = \operatorname{sgn}\left(\sum_{j=1}^{N} w_j x_j - \theta\right) = \begin{cases} +1 \text{ if } \sum_{j=1}^{N} w_j x_j - \theta \geq 0 \\ 0 \text{ if } \sum_{j=1}^{N} w_j x_j - \theta < 0, \end{cases} \quad (1)$$

where $\boldsymbol{X} = (x_1,...,x_N)^T$ is the binary input vector, the real-valued coefficients w_j are the threshold function weights and θ - is the threshold value.

A node that computes a **l**inear **t**hreshold function is called an *LTE* **e**lement. Each LTE unit separates the input space into a closed positive and an open negative half-space. If for a given set of L input vectors there exists a hyperplane, such that each vector lies on a pre-assigned side of it, then the set of training input vectors is referred to as *linearly separable*. The number of Boolean functions[2] that can be defined on a set of L vectors $\boldsymbol{X_1},...,\boldsymbol{X_L}$ *in* $\Re^N$ $(L \leq N)$ equals 2^L. (The dimension of the space to which the input vectors belong – is not important[3].) If the L vectors $\boldsymbol{X_1},...,\boldsymbol{X_L}$ *in* $\Re^N$ $(L \leq N)$ are in a *general position*[4], then all Boolean-valued functions defined over these points are linearly separable (i.e. an N-input LTE is able to compute all 2^L Boolean functions. For a linear threshold function it is difficult to decide, whether a set of points contains an even or an odd number of elements, and a single LTE element cannot compute some simple functions, like XOR or ~XOR function.

If the number of input vectors L equals $2N$, then an LTE unit can compute with probability ≥ 1/2 any randomly chosen Boolean function over the set of inputs. As L ($L > 2N$) increases, then this probability decreases sharply to zero. Because of this, the capacity of an N-input LTE is equal[5] to $2(N+1)$ [1].

It can be concluded that the total number of threshold functions in an N-dimensional space for $L>N$ increases polynomially ($\boldsymbol{2^{N(N-1)}}$), while the total number of Boolean functions, definable by N Boolean inputs, grows

[1] (or equivalently $\boldsymbol{f}(X): \{-1;1\}^N \rightarrow \{-1;1\}$)

[2] i.e. a function of one of the forms: $f(\boldsymbol{X}): \{0;1\}^N \rightarrow \{0;1\}$ or $f(\boldsymbol{X}): \{-1;1\}^N \rightarrow \{-1;1\}$.

[3] This is equivalent to the problem when L cuts with $(N-1)$ dimensional hyperplanes in N-dimensional space define 2^L different regions.

[4] A set of L vectors in $\Re^N$ is said to be in a general position if every subset of L or fewer points forms a linearly independent set.

[5] Very often in the literature the threshold value θ is assumed to equal zero. This is achieved by increasing the dimension of every input vector by augmenting it with an entry that equals –1, and by increasing the dimension of the weight vector by augmenting it with θ. In such a case the capacity of an LTE equals $2N$.

exponentially (2^{2^N}), and as N increases a percentage of threshold functions in relation to the total number of Boolean functions – goes to zero.

A modern variant of Kolmogorov's theorem states that for any continuous function $\boldsymbol{f}:[0,1]^N \to [0,1]$ there exist functions of one argument $\boldsymbol{g}$ and Φ_q (for $q = 1,...,2N+1$), such that:

$$\boldsymbol{f}(x_1, x_2, ..., x_N) = \sum_{q=1}^{2N+1} g\left(\sum_{p=1}^{N} \lambda_p \phi_q(x_p)\right). \tag{2}$$

In other words, any continuous function can be reproduced exactly by a finite network of processing elements, providing enough nodes to compute the necessary primitive functions.

And if there is a need to approximate functions and no demand to reproduce them exactly, except for a bounded approximation error, then there is a possibility to look for the best possible approximation to a given[6] function $\boldsymbol{f}$.

2. Node, edge and weights complexity of threshold circuits

The complexity of a circuit (or a net) is usually expressed in terms of the input size (i.e. the number of input variables of the circuit), the size, depth or fan-in/fan-out of this circuit. A common objective is to minimize the depth and the size of a circuit.

In analysing neural nets (circuits), the main difficulty lies in the nonlinearity of the nets functions. With regard to analysing threshold gates, especially suitable are geometric approaches, in which an N-input threshold function (gate) corresponds to a hyperplane in $\Re^N$. This interpretation (as mentioned before) can be used to determine the functions that cannot be computed by an LTE unit or just to count the number of Boolean functions computable by it. But this is not easy if a threshold circuit has a depth of 2 or more, because the inputs to the gates in the second level are LTE units themselves and the linear combination computed in the second level is a nonlinear function of its inputs.

One of the theorems that can be easily proved[7] states that:

Every N-variable Boolean function $\boldsymbol{f}(x_1, x_2, ..., x_N): \{-1;1\}^N \to \{-1;1\}$ *can be computed by a depth-2 threshold*[8] *(or AND-OR) circuit – with at most* $\left(2^{N-1}+1\right)$

[6] This is a version of Kolmogorov's (1957) and Sprecher (1965) theorem, extended by Hornik, Strinchcombe, White (1989) and other researchers.

[7] The proof follows from this that every Boolean function of N variables can be written in two forms: a) in a canonical Sum of Products notation, where each of k $(k \leq 2N)$ product-terms is a distinct AND function, where a subset of the variables are complemented; and b) in a canonical Product of Sums notation, where each of l $(l \leq 2N)$ sum-terms is a distinct OR function, where a subsets of the variables are complemented [2].

[8] The depth of a gate – is the maximum number of edges along any directed path from the input

LTE elements or by a depth-3 threshold circuit – with at most $O\left(2^{N/2}\right)$ *LTE units.*

But the implementation of such a circuit would require too many resources, even for a small number of *N.* Therefore, the objective is to minimize the amount of hardware and computation time (i.e. a circuit size and depth). A circuit of smaller depth is a faster circuit. It is more difficult to implement an LTE element with exponentially large integer weights than an LTE with integer weights that are polynomially bounded. It has been proved that a polynomial-size threshold circuit with polynomially bounded weights can be simulated by another polynomial-size threshold circuit with unit weights and the same depth, where only the edge complexity is increased by a polynomial factor [2]. By using sigmoidal functions instead of LTE elements - for some Boolean functions the size of the network can be reduced, at least by a logarithmic factor [2].

Most common arithmetic functions can be efficiently realized by small depth-d polynomial size threshold circuit (represented by LT_d). The Addition and Comparison functions can be computed by LT_2 circuits [2]. A network that for a given set of L input vectors can compute all of the 2^L possible Boolean functions (called a universal network) - must have at least $\frac{L}{1+\log_2 L}$ number of weights [2].

If the hidden layer of a net contains m units and the input vector is of the dimension N, then the maximum number of classification regions equals $2\sum_{i=0}^{N-1}\binom{m-1}{i}$. If the number of input vectors is higher than their dimension, it may happen that there may not be enough classification regions to compute a given Boolean function, and there will be a need to increase the number of net input lines.

3. Limitations of AND-OR circuits compared to threshold circuits

Threshold circuits[9] are especially valuable in an application in which we want to reduce the execution time of a circuit (net) to two or three layers of computational delay without employing a large number of gates [2]. The parity and majority functions cannot be implemented in a fixed number of layers without using an exponentially growing number of AND-OR gates, even when unbounded fan-in is used. The majority function k out of N can be built by a single LTE unit. On the other hand, instead of using AND-OR gates, we can use

nodes to that gate. The depth of a circuit – is the maximum depth among all gates in the circuit.

[9] An AND-OR circuit is a Boolean circuit in which every gate is an AND gate, OR gate or an inverter.

a direct realization of Parity in depth-2 threshold circuit comprising of *(N+1)* LTE elements, which means that in this sense, an LTE gate is exponentially more powerful than an AND-OR gate.

The main difference between conventional AND-OR and threshold circuits – is that AND-OR circuits *cannot guarantee a constant delay*. An LTE unit can be used to build multiplication or division circuits that guarantee a constant delay for 32 or 64 bit operands [3].

4. The *Exponential* or *polynomial* number of threshold input function

When the input patterns are linearly nonseparable, the behaviour of two or more deep circuits (nets) is not well understood. An analysis of the relation between the output function of a threshold gate and its arbitrary set of input functions, using geometrical and linear algebraic tools allows to derive a variety of lower bound results. In this approach an *N*-variable Boolean function is represented as a vector in $\Re^N$. Below is shown a method of correlations which allows to use procedures for decomposing[10] a given Boolean function $\boldsymbol{f}$, which together show, that *every N-variable Boolean function is a threshold function of* ***polynomially*** *many input functions, none of which is significantly correlated with* $\boldsymbol{f}$ [2].

A Boolean function $\boldsymbol{f}\left(x_1, x_2, ..., x_N\right): \{-1;1\}^N \to \{-1;1\}$ can be considered as a (column) vector in $\boldsymbol{2^{2^N}}$. In this case each of $\boldsymbol{f}$'s 2^N components represents $\boldsymbol{f(x)}$ for a distinct value assignment $x \in \{-1,1\}^N$ of the *N* Boolean variables of $\boldsymbol{x}$.

Let the Boolean functions $\boldsymbol{f_1}, ..., \boldsymbol{f_s}$ be the inputs of a threshold gate with the weight vector $\boldsymbol{w} = \left(w_1, ..., w_s\right)^T$ $\left(\boldsymbol{w} \in \Re^S\right)$. We say that a gate computes a Boolean function if the following vector equation holds:

$$\boldsymbol{f} = \operatorname{sgn}\left(\sum_{j=1}^{S} \boldsymbol{f}_j w_j\right), \tag{3}$$

or in a matrix form: $\boldsymbol{f} = \operatorname{sgn}\left(\boldsymbol{\Phi w}\right)$, where the input matrix $\boldsymbol{\Phi} = \left[\boldsymbol{f_1}, ..., \boldsymbol{f_S}\right]$ is a $\left(2^N \times S\right)$ - dimensional matrix, whose columns are the input functions. The Boolean function $\boldsymbol{f}$ is a threshold function of $\boldsymbol{f_1}, ..., \boldsymbol{f_S}$, if there exists a threshold gate with the weights[11] $\boldsymbol{w} = \left(w_1, ..., w_s\right)^T$ and with inputs $\boldsymbol{f_1}, ..., \boldsymbol{f_S}$ that computes $\boldsymbol{f}$.

[10] called *threshold decomposing* procedures.

[11] The dimension of the weight vector has been increased by augmenting it with this gate threshold value θ.

At the same time the Boolean function is the output of a threshold gate, whose input functions are $\boldsymbol{f_1},...,\boldsymbol{f_S}$ if, and only if the linear combination $\boldsymbol{\Phi w} = \sum_{j=1}^{S} \boldsymbol{f_j} w_j$, defined by the gate, lies in the interior of $\boldsymbol{f}$'s orthant. (The inner product of two vectors is non-negative if they lie in the same orthant.)

If this function $\boldsymbol{f}$ is orthogonal to $\boldsymbol{\Phi} = [\boldsymbol{f_1},..., \boldsymbol{f_S}]$ (i.e. the correlation between $\boldsymbol{f}$ and $\boldsymbol{\Phi}$ is zero), then $\boldsymbol{f}$ is not a threshold function of $\boldsymbol{\Phi}$.

The correlation $C_{\boldsymbol{f_1 f_2}}$ between two *N*-variable Boolean functions $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ is equal to [2]:

$$C_{f_1 f_2} = \frac{\left(\boldsymbol{f_1^T f_2}\right)}{2^N} = 1 - \left[2 d_H\left(\boldsymbol{f_1}, \boldsymbol{f_2}\right) / 2^N\right], \tag{4}$$

where $d_H(\cdot)$ is the Hamming distance. This allows to interpret it as a measure of how 'close' the two Boolean functions are. This correlation is a multiple of $2^{-(N-1)} \in [-1,1]$. The correlation vector of a function $\boldsymbol{f}$ with its input functions is defined as:

$$\boldsymbol{C_{f\Phi}} = \frac{\left(\boldsymbol{\Phi^T f}\right)}{2^N} = \left[C_{ff_1}, C_{ff_2}, ..., C_{ff_S}\right]^T. \tag{5}$$

and it may take $2^N + 1$ values. This means that for a given $\boldsymbol{f_1},..., \boldsymbol{f_S}$ functions the correlation vector $\boldsymbol{C_{f\Phi}} = \left[C_{ff_1}, C_{ff_2}, ..., C_{ff_S}\right]^T$ may assume at most $\left(2^N + 1\right)^S$ different values over all $\boldsymbol{2^{2^N}}$ Boolean functions of *N* variables.

This implies that many Boolean functions share the same correlation vector $\boldsymbol{C_{f\Phi}}$, but at the same time the Boolean threshold function $\boldsymbol{f}$ with $\boldsymbol{\Phi} = [\boldsymbol{f_1},..., \boldsymbol{f_S}]$ input Boolean functions does not share its correlation vector (i.e. $\boldsymbol{C_{g\Phi}} \neq \boldsymbol{C_{f\Phi}}$) with any other Boolean function $\boldsymbol{g}$ ($\boldsymbol{g} \neq \boldsymbol{f}$) [2]. This implies that there are at most $\left(2^N + 1\right)^S$ threshold functions of any set of *S* input functions. It may be further deduced that every Boolean function of *N* variables (*N*-even) may be expressed as a Boolean threshold function of *2N* Boolean input functions $\boldsymbol{f_1}, \boldsymbol{f_2},..., \boldsymbol{f_{2N}}$, such that [2]: 1) $\boldsymbol{C_{ff_i}} = 0$ for $i \in \{1,...,2N-1\}$, $i \neq 4$, and 2) $\boldsymbol{C_{ff_4}} = 2^{-(N-1)}$.

5. Conclusions

The above approach was possible owing to the integration of the above mentioned geometric interpretations of the Boolean function, with some tools borrowed from linear programming.

The geometric framework is often integrated with the theory of linear programming, e.g. linear programming algorithms are used to determine the LTE's weights. Linear programming concepts, like duality or Farkas' lemma can be used to describe functions that cannot be computed by a single LTE or when the training set is linearly nonseparable [2].

References

[1] Gardener E., *Maximum Storage Capacity in Neural Networks*, Europhysics, Letters, 4 (1987) 481.
[2] Siu K-Y., Roychowdhury V., Kailath T., *Discrete Neural Computation*, A Theoretical Foundation, Prentice Hall PTR, NJ, (1996).
[3] Rojas R., Neural Networks. *A Systematic Introduction*, Springer-Verlag, Berlin, (1996).
[4] Tadeusiewicz R., *Sieci neuronowe jako współbieżne i adaptacyjne systemy przetwarzania informacji*, Scientific Raports: AGH Automation & Robotics Inst., AGH, 1 (1991), in Polish.
[5] Borowik B., *Pamięci asocjacyjne*, Mikom, Warszawa, (2002), in Polish.