Properties of Parametric Feedforward Neural Networks

Claudio Moraga

Department of Computer Science and Computer Engineering
University of Dortmund
Germany

moraga@cs.uni-dortmund.de

Abstract

Feedforward Neural Networks are considered to be "black boxes" in the sense that even though they may solve classes of complex problems, it is not evident how these networks solve those problems. Parametric Neural Networks are introduced and it is shown that every classical feedforward Neural Network has an equivalent Parametric Neural Network with the same topology. Under some scaling constraints, every Parametric Neural Network has also an equivalent feedforward Neural Network with the same topology. Moreover, Parametric Neural Networks support an interpretation of the work of a Network at the level of nodes and in this sense may be considered to be "grey boxes". Finally it will be shown that both classical as well as Parametric Neural Networks (using non-linear Activations functions at *all* nodes) are universal approximators.

1 INTRODUCTION

Feedforward Neural Networks with supervised learning may be associated to a non-parametric regression system in the sense that the final values adjusted for the weights to minimize the mean square error over a finite subset of training pairs from $(R^n) \times (R^m)$ or more frequently, from $(R^n) \times ([0.1]^m)$, give no information on how a Neural Network works or why it gives an acceptable solution to the problem. It is said in this context, that Neural Networks are considered to be "black boxes".

The problem may be seen as a case of low level coding. Human beings seem to have difficulties to understand pieces of informations if the degree of granularity chosen for its representation alphabet is too small. (Who can, e.g., understand a piece of a program written in machine language?). Consider the problem of searching for partial symmetries in boolean functions of, say 6 variables, by inspecting their Karnaugh maps. This

is a very cumbersome problem for the human observer, since binary subpatterns have to be recognized against a "binary background". If instead of doing it in this way the Walsh Spectrum [Wal23] of such functions is presented on a spectral map analog to the Karnaugh maps, the problem (for the human observer) turns out to be much simpler, since the Walsh spectral coefficients take even values in the intervall [-64, 64] and the distinguishability of subpatterns is highly improved. The sequential algorithms to solve the problem, however have on the average the same complexity as in the binary case. Interestingly enough, Neural Networks seem to have the same problem if they are used to solve similar problems. In a very striking experiment H. Katayama and his colleagues [KSH92], reported to have trained a Neural Network to successfully classify the first 10 digits represented as 8x8 "black and white" matrices. The Neural Network was trained with the pixels of the matrices. The training was later repeated, except that instead of using the original matrices, their Walsh spectra were used. The Neural Network did again succeed in correctly classifying the digits (represented by the Walsh-transformed matrices), but it learned much faster to do it!

The main result of the present paper is the following. If we consider the structure of a Neural Network as a graph and associate the weights to the edges of the graph then it is possible to transfer a part of the information coded by the weights to the nodes. This supports a possible (partial) interpretation of the work of the nodes and improves the speed of learning. In the next section we present a new model of feedforward Neural Networks, which uses as activation function a parameterized sigmoid [Han96], and discuss its relationship with classical Networks. Furthermore we discuss the interpretation supported by the new model. Finally we prove that Parametric Neural Networks (PNNs) are universal approximators.

2 PARAMETRIC NEURAL NETWORKS MODEL

From the structural point of view, feedforward Neural Networks are directed acyclic graphs with labelled edges. A predefined transfer function is associated to each node of the graph. This transfer function is the result of the composition of two auxiliary functions: an input function, which evaluates the excitation of the node, usually by computing the weighted sum of the input signals minus a reference threshold value (i.e, the input function is an affine function); and an output function known as activation function, that is evaluated according to the argument computed by the input function. Classical Neural Networks mostly use a sigmoidal activation function.

A Parametric Neural Network (PNN) [Han96] is characterized by elementary processors at the nodes having a parameterized sigmoidal activation function given by:

$$f(x) = [c_1/(1 + exp(-c_2(x - c_4))] - c_3,$$

where $x \in R$, represents the weighted summation of the input signals of the processor under consideration including a possible bias, and the new parameters have the following

meaning:

- c_1. gain (attenuation)
- c_2. slope at the inflection point of the sigmoid, which is inverse proportional to the range of (approximate) linearity of the elementary processor
- c_3. bias, which modifies the degree of symmetry of the transfer function
- c_4. delay of the argument.

It has been shown [Mor97] that every PNN has an equivalent classical Backpropagation Neural Network with the same topology, where the new parameters c_1 and c_2 may be expressed as scaling factors affecting it several weights at the same time. From a local point of view, parameter c_3 may be given the value $(c_1/2)$ and changes the activation function from sigmoid into tanh. Moreover the value of all parameter c_3 of one layer affect the thresholds of the succeeding layer. Finally, parameter c_4 allows interpreting the threshold of a node as a delay in applying the corresponding input signal. It is fairly obvious, that every calssical Neural Network has an equivalent PNN with the same topology and the following parameters: $c_1 = 1$, $c_2 = 1$, $c_3 = 0$ and $c_4 = 0$. Backpropagation Neural Network considered to be "black boxes", in the sense that they may solve a problem by converging to an appropriate set of weights, however from the values of the weights it is not simple to infer how the network solves the problem. PNNs might well be considered "grey boxes", since some more information becomes availabe at the node level partially illustrating how the elementary processors work. A gain factor, a linearity factor and a symmetry factor become explicit; even the own threshold at every processor, which under the Backpropagation algorithm [RHW86] only receives a numerical adjustment, may be interpreted as a measure of the delay of reaction of the corresponding processor.

The new parameters of a PNN add degrees of freedom (and help to better understand the work of the elementary processors); but they must be adjusted additionally to the weights. Which is the resulting overhead?

Hybrid training methods have been proposed by Han [Han96], [HMS96], by using a Genetic Algorithm to obtain a layerwise preliminary adjustment of the new parameters followed by a gradient descent algorithm to tune the parameters and adjust the weights. A full integration of adjustment of the new parameters together with the weights within R-Prop [RiB94] has been successfully done by Bui [Bui96]. Inspite of having more parameters to adjust, PNNs come faster than classical Neural Networks to levels of acceptable minimal errors and reach better minimal mean square errors [Han96], [Bui96], [HMS96]. Both the fact that adjusting the c_1 and c_2 parameters is equivalent to simultaneously adjusting several weights of the Network -(what is not possible with the Backpropagation Algorithm)- and the fact that by increasing the dimensionality of the error-space the former local minima may disappear or become weaker attractors, support a plausible explanation for the better performance of PNNs.

3 PARAMETRIC NEURAL NETWORKS ARE UNIVERSAL APPROXIMATORS

3.1 Notation and Background

Any (Borel) measurable function $f : R \to [0,1]$ that is increasing and satisfies the following boundary conditions: $lim_{x\to\infty} f(x) = 1$ and $lim_{x\to-\infty} f(x) = 0$ is called a *squashing* function. Notice that squashing functions are not necessarily continuous, but only measurable. Let Γ denote the set of all squashing functions.

Let $\mathbf{A}^r = \{A : R^r \to R | A \text{ an affine function}\}$ and let K be a compact subset of R^r

A network that realizes the function

$$\sum^r(G)(x) = \sum_{i=1}^{q} \beta_i G(A_i(x)),\ x \in K \subset R^r,\ A_i \in \mathbf{A}^r,\ \beta_i \in R,$$

$$q \in N, G \in \Gamma$$

will be called a $\sum^k(G)$-network.

A feedforward network that realizes the function

$$\sum\prod^r(G)(x) = \sum_{j=1}^{q} \beta_j \prod_{k=1}^{p_j} G(A_{jk}(x)),\ x \in K \subset R^r,\ A_{jk} \in \mathbf{A}^r,$$

$$\beta_j \in R,\ q \in N,\ p_j \in N$$

where G is a nonconstant continuous function, will be called $\sum\prod^r(G)$-network.

Theorem 1 [HSW89]: For every squashing function Ψ, every r, and every probability measure μ on (K, B^r), $\sum^r(\Psi)$ is uniformly dense on compacta in C^r and ρ_μ-dense in M. (B^r denotes the Borel σ-field of R^r, C^r denotes the set of continuous functions and M^r, the set of all Borel measurable functions from K to R.)

Theorem 2 [HSW89]: For every continuous nonconstant function G maping R to R, every r, and every probability measure μ on (R^r, B^r), $\sum\prod^r(G)$ is uniformly dense on compacta in C^r and ρ_μ-dense in M^r.

Corollary 2.1 [HSW89]: For every continuous squashing function Ψ, every r, and every probability measure measure μ on (R^r, B^r), $\sum\prod^r(\Psi)$ is uniformly dense on compacta in

C^r and ρ_μ-dense in M^r.

From Theorem 1 and Corollary 2.1 follows that both $\sum^r(G)$-networks, i.e. feedforward networks with one hidden layer (with squashing activation functions) and $\sum\prod^r(G)$-networks (using continuous squashing functions) can approximate any continuous function uniformly on any compact set and any measurable function arbitrarily well in the corresponding metric, regardless of the squashing or continuous squashing functions, respectively, regardless of the dimension of the input data and of the probability measure μ. Both $\sum^r(G)$-networks and $\sum\prod^r(G)$-networks are universal approximators (in the sense explained above).

3.2 New Results

PNNs as well as classical Backpropagation Neural Networks use all over sigmoid functions (which are continuous squashing functions) and are not restricted to have a single hidden layer. In what follows, the universal approximation capability of these Neural Networks will be proven.

Theorem 3: A Backpropagation Neural Network with one hidden layer using squashing functions and one output node with a *continuous*, *invertible* squashing function as activation function is a universal approximator for any continuous function $f : K \rightarrow [0,1]$. For any $\varepsilon > 0$ if the approximating Neural Network realizes $\varphi : K \rightarrow [0,1]$, then $\sup_K(|f(x) - \varphi(x)|) < \varepsilon$.

Proof: Let $S \in \Gamma$ be a continuous invertible squashing function and for all $x \in K$ let

$$\Phi(x) = (S^{-1} \circ f)(x)$$

It becomes apparent that Φ is a continuous function with the same domain as f. According to Theorem 1, there exists a $\sum^r(G)$-network that can approximate Φ uniformly.

Adding in cascade a new output processor realizing $S(A(y))$, $A \in \mathbf{A}^k$, $y \in R$ and $k \in N$, the resulting network produces

$$\varphi(x) = (S \circ \Phi)(x) = (S \circ (S^{-1} \circ f))(x) = f(x)$$

Finally, the activity of the original linear output processor may be realized by the affine function at the input of the new S-processor (see Figure 1). The assertion follows.

The author gladly acknowledges that this proof was suggested to him by Dr. Juan Luis Castro, ETS Ingeniería Informática, Departamento de Ciencias de Computación e Inteligencia Artificial, Universidad de Granada, Spain.

It becomes apparent, that if $f : K \rightarrow [0,1]$ is a measurable function, the same proof-strategy may be used.

Theorem 4: A Backpropagation Neural Network with one or *more* hidden layers using *continuous* squashing functions and one output node with a *continuous*, *invertible* squashing function as activation function is a universal approximator for any continuous function $f : K \rightarrow [0,1]$. For any $\varepsilon > 0$ if the approximating Neural Network realizes $\varphi : K \rightarrow [0,1]$, then $\sup_K(|f(x) - \varphi(x)|) < \varepsilon$.

Proof: Let f, Φ and S be defined as in Theorem 3. According to Corollary 2.1 there exists a $\sum \prod^r(G)$-network with continuous squashing functions at the first hidden layer, that approximates Φ uniformly. According to Theorem 1, every $\prod$-processor may be realized arbitrarily well with a $\sum^k(G)$-network, where k is given by the number of inputs of the corresponding $\prod$-processor and G is a continuous squashing function. (Obviously $G \in \Gamma$), (see Figure 2, left and center). Connecting an S-processor in cascade, a new network is obtained (Figure 2, center) that computes:

$$\varphi(x) = (S \circ \Phi)(x) = (S \circ (S^{-1} \circ f))(x) = f(x).$$

Because of the associativity of the addition all linear processors in the last and second last hidden layers may be combined into just one and the task of this resulting processor may be realized by the affine function at the input of the S-processor, as discussed in Theorem 3 (see Figure 2, right). This concludes the proof for networks with two hidden layers. It is simple to see that since in the former process of replacement of the $\prod$-processors by a $\sum^r(G)$-networks the G functions were selected to be continuous squashing functions, then every G-processor in the last hidden layer may be realized by an appropriate $\prod^r(G)$-network and the new $\prod$-processors may be realized by a $\sum^r(G)$-network, where G would be selected to be a continuous squashing function. As discussed above, the remaining linear processor may be absorbed by the output processor. This replacement strategy allows the generation of equivalent Neural Networks with increasing number of hidden layers. Since from Theorems 1 and 2 $\sum^r(G)$ and $\prod^r(G)$-networks can approximate their goal continuous functions with a given arbitrary accuracy it will always be possible to make all required replacements and still satisfy the overall accuracy requirements for the approximation. The assertion follows.

It becomes apparent, that if $f : K \rightarrow [0,1]$ is a measurable function, the same proof-strategy may be used.

Both Theorem 3 and Theorem 4 also apply to PNNs if the $c_1 = 1$ and $c_3 = 0$ at the output processor, since then the resulting sigmoid is a continuous, invertible squashing function and for every Backpropagation feedforward network there exists an equivalent PNN with the same structure.

Theorem 5: A PNN with one or more hidden layers (using a parameterized sigmoid at

every processor) is a universal approximator for any continuous function $f : K \rightarrow [-a, b]$.

Proof: Define $g : K \rightarrow [0, 1]$ such that for all $x \in K$

$$\begin{aligned} g(x) &= [f(x) + a]/(a + b). \\ \text{Obviously } f(x) &= (a + b)g(x) - a \quad \text{for all } x \in K. \end{aligned}$$

After Theorems 3 and 4 there exists a PNN using parameterized sigmoids, with $c_1 = 1$ and $c_3 = 0$ at the output processor that approximates arbitrarily well $g(x)$. Since to obtain f(x) scaling and shifting is necessary, this may be easily obtained by using $c_1 = (a + b)$ and $c_3 = a$ at the output processor of the PNN. This concludes the proof.

A detailled analysis of tolerances may be found in [Mor97].

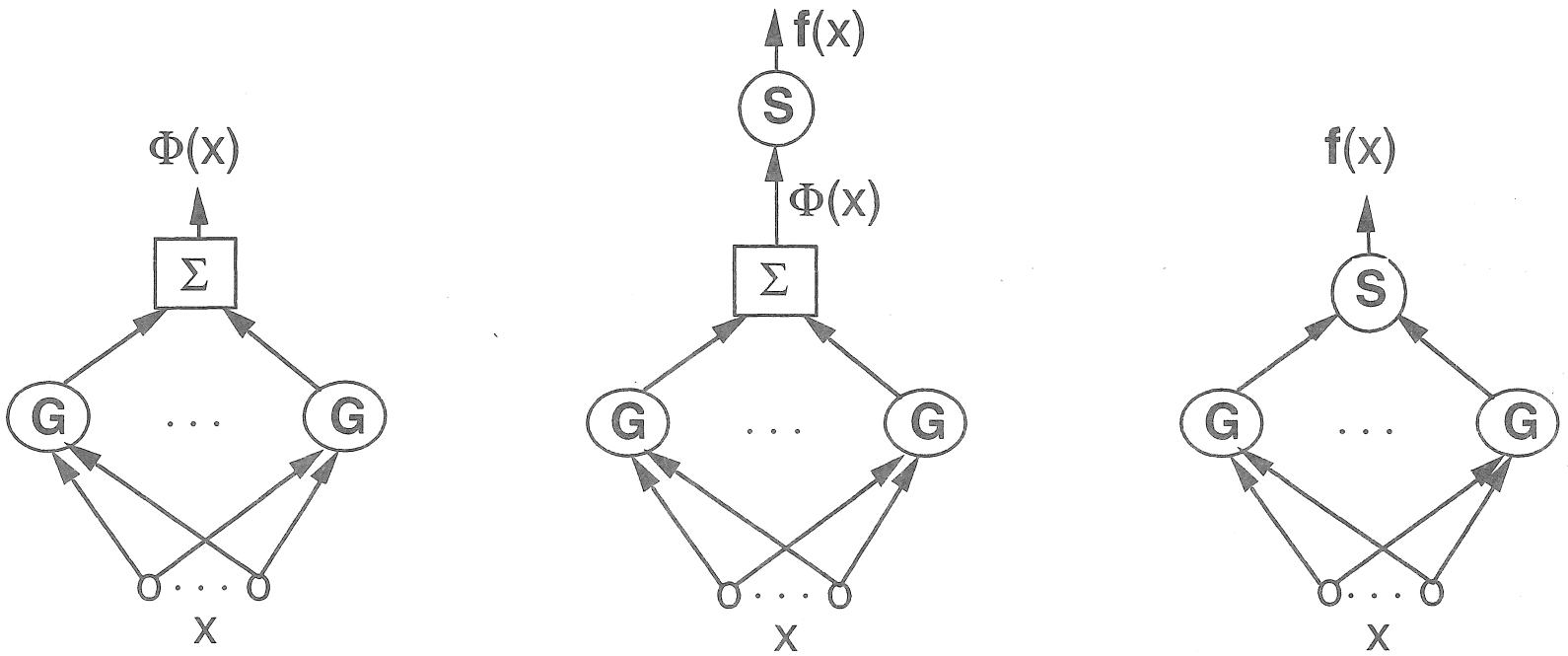

Figure 1. Net-transformations related to Theorem 3

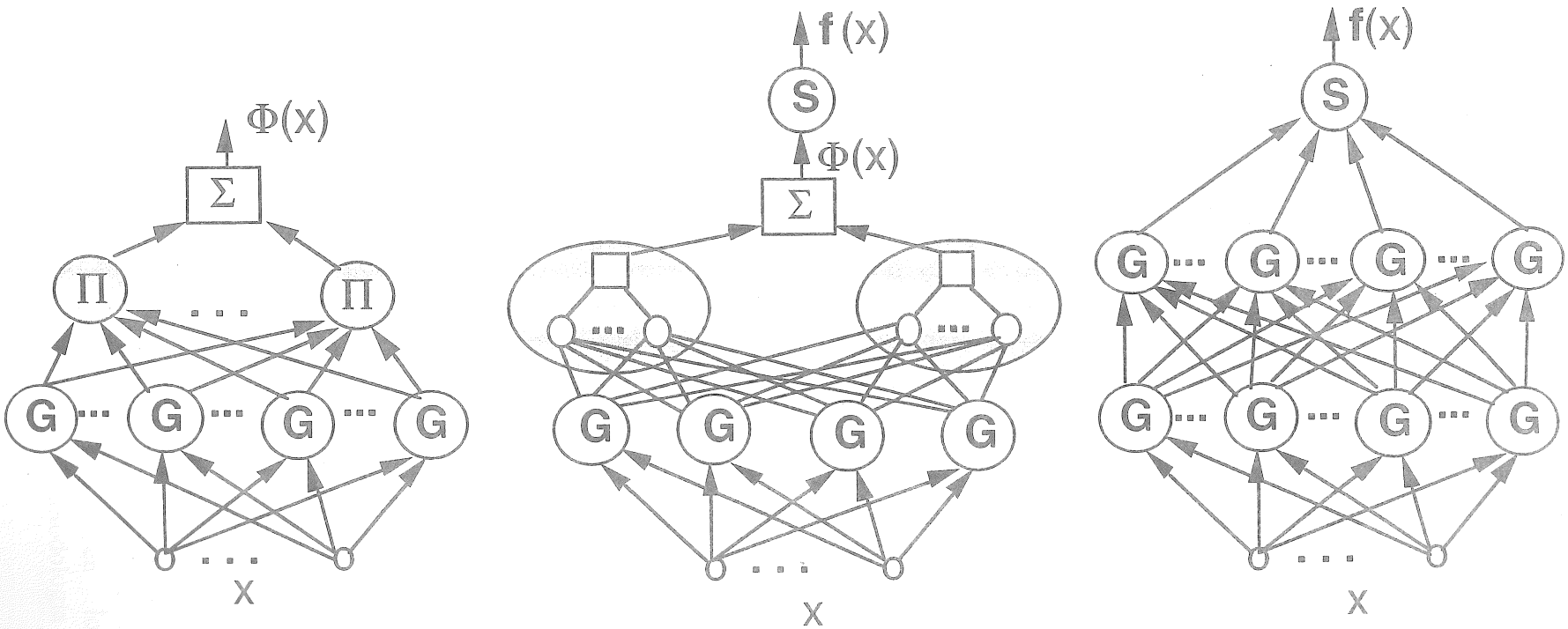

Figure 2: Net-transformations related to Theorem 4

References

[Bui96] Bui T.B.: Vergleichsanalyse zweier Optimierungsstrategien für vorwärtsgerichtete Netze. Diplomarbeit, FB Informatik, Universität Dortmund, (1996)

[Han96] Han J.: Optimization of Feedforward Neural Networks. Dissertation, FB Informatik, Universität Dortmund, (1997)

[HMS96] Han J., Moraga C., Sinne S. : Optimization of Feedforward Neural Networks. Engineering Applications of Artificial Intelligence 9 (2), 109-119, (1996)

[HSW89] Hornik K., Stinchcomb M., White H.: Multilayer Feedforward Networks are Universal Approximators. Neural Networks 2, 359-366, (1989)

[KSH92] Katayama H., Shimomura T., Harada H., Konishi R.: The learning of a Neural Network using the Hadamard Transform. Proc. 2nd. International Conf. on Fuzzy Logic and Neural Networks, 1049-1052, Iizuka, Japan, (1992)

[Mor97] Moraga C.: A grey model for Parametric Feedforward Neural Networks. Forschungsbericht 644, Fachbereich Informatik, Universität Dortmund, (1997)

[RHW86] Rumelhart D.E., Hinton G.E., Williams R.J.: Learning Representations by back-propagating error. Nature 323, 533-536, (1986)

[RiB93] Riedmiller M.H., Braun H.: A direct adaptive method for faster backpropagation learning: The Rprop algorithm. Proc. IEEE Int. Conf. on Neural Networks (ICCN), 586-591, (1993)

[RuM86] Rumelhart D.E., McClelland J.L. (Eds.): Parallel Distributed Processing. MIT Press, Cambridge MA, (1986)

[Wal23] Walsh J.L.: A closed set of orthogonal functions. American Journal of Math. 45, 5-24, (1923)