Function Approximation with Spiked Random Networks

Erol Gelenbe, *Fellow IEEE* *
School of Computer Science
University of Central Florida
Orlando, FL 32789
erol@ee.duke.edu

Zhi-Hong Mao and Yan-Da Li
Department of Automation
Tsinghua University
Beijing 100084, P.R. China

September 2, 1998

Abstract

This paper examines the function approximation properties of the "random neural network model" (Gelenbe 89,90,93) [6, 8, 12]) or GNN. The output of the GNN can be computed from the firing probabilities of selected neurons. We consider a feedforward Bipolar GNN (BGNN) model (Gelenbe, Stafylopatis and Likas 91 [9]) which has both "positive and negative neurons" in the output layer, and prove that the BGNN is a universal function approximator. Specifically, for any $f \in C([0,1]^s)$ and any $\epsilon > 0$, we show that there exists a feedforward BGNN which approximates f uniformly with error less than ϵ. We also show that after some appropriate clamping operation on its output, the feedforward GNN is also a universal function approximator.

Keywords: Spiked neural networks, random neural networks, function approximation.

1 Introduction

The theory of function approximation by neural networks is a necessary underpinning to many applications such as pattern recognition, data compression, time series prediction, adaptive control by neural networks, etc.. In applications, the main design objective is often to find a network which is a good approximator to some desired input-output mapping. However, in addition to the conventional notion of approximation, neural networks are valued especially for their ability to generalize, i.e. to use information they have learned in order to synthesize similar but non-identical input-output mappings under novel circumstances. The desired mapping can be presented to the network via a set of examples, as is often the case in supervised learning, or by a time series (when neural nets are used for prediction), or even by the observation of an

*This author's work was supported by the Office of Naval Research under grant number N00014-97-1-0112.

unknown dynamical system. Many models of neural networks have been mathematically demonstrated to be universal approximators, and related results include proofs for the conventional multilayer perception (MLP) [4], the radial basis function (RBF) neural network [10], the fuzzy neural network [18], the wavelet neural network [19], and the rational function neural network [14]. Approximation theory [2] deals with the approximation of a function $f(X)$ of an input vector X by some other function $F(w, x)$ having a fixed number of parameters denoted by the vector w. The parameters w are chosen so as to achieve the best possible approximation of the function f. For instance one may choose w so as to minimize some norm or error criterion $||f(X) - F(w, X)||$. Since neural networks are used in so many applications, it is desirable that a neural network be a universal approximator in some useful sense, e.g. it should at least approximate any continuous function on a compact set to an arbitrary degree of accuracy. Obviously, one can consider that the neural network's inputs are described by X, while the parameters w are the network's "weights". On the other hand, it is well known [3] that the accurate approximation of a function class can lead to poor "generalization" capability, and generalization is the major attractive property of neural networks. Intuitively one may imagine that both approximation and generalization are desirable properties under different circumstances.

This paper is devoted to the approximating capabilities of spiked networks, which has not received much attention in the literature. Spiking models (see for instance [15, 16]) are of great importance because they represent more closely the manner in which signals are transmitted in real (biological) neural networks where they often are voltage (action potential) spikes rather than analog levels, although we need to recognize that no single mathematical model is known can capture all aspects of neuronal signaling. Historically, spiking behavior in neurons has been described by the well known Hodgkin-Huxley equations [1]. However the stochastic nature of the spiking behavior has been recognized by many authors and discussed in various publications (e.g. [15, 16]).

The neural network model we discuss in this paper [5, 6, 7, 8, 12] ("Gelenbe's neural network" or GNN) is a novel biophysically inspired spiked model which differs substantially from existing deterministic models (e.g. the MLP) and random models (e.g. the Boltzmann machine or the models described above). It is based on a direct point process representation, with a discrete state-space internal state representation of each interconnected neuron [5, 6, 7, 8]. The rich mathematical structure of the network in terms of an infinite set of Chapman-Kolmogorov

equations leads to compact closed form solution for the feedfowrward and recurrent case [5, 12] that the model has a closed form solution for network state even in the recurrent case. This in turn yields efficient numerical algorithms. Typically, a spiked stochastic model will include some internal representation of each neuron's state, and a probabilistic representation of successive firing times as a function of state (see for instance [16], p. 65); additionally, rules need to be given about the manner in which the internal state changes when excitatory and inhibitory spikes are received, and about how the internal state changes after firing. In our model, contrary to other models which are available, the internal state is a non-negative integer; rises or falls depending on the excitatory or inhibitory nature of incoming sipkes, and it drops each time the neuron fires. Inter-firing intervals are exponentially distributed (as in some of the models discussed in [16]). In our model the representation of a recurrent network results in a system of coupled Chapman-Kolmogorov equations [24] for a continuous time Markov chain with a countably infinite number of states.

Recent work on a variety of significant applications of this model [9, 12, 17, 21, 20, 22, 23] has shown that the GNN is able to efficiently carry out desirable functions such as optimization, learning, associative memory, pattern recognition, etc.. It has been successfully used in some classical NP-hard combinatorial optimization problems such as the minimum vertex covering problem [11, 17], the traveling salesman problem [13], and the Steiner problem in networks [23], where its main advantage is that it can be easily solved numerically, without the use of lengthy Monte Carlo simulations which are needed for addressing similar problems with the Hopfield network or the Boltzmann machine.

The GNN learning algorithm has been described previously [12] and has been used extensively (e.g. [21, 20, 22]). It is based on a gradient descent based minimization, and is of complexity $O(n^3)$ for a fully recurrent n-neuron network. The ability of the GNN to generalize after being trained with this learning algorithm has been demonstrated in several significant applications. For instance, a GNN trained to compress a specific image ("Lena") has been demonstrated to compress efficiently (high compression ratio, high signal to noise ratio) long video sequences which contain images that the network has never been shown [20, 21]. Another network which was trained on some 20 to 30 samples of 5 by 5 pixel image elements in brain magnetic resonance images has been able to accurately segment large images into areas of grey matter, white matter and cerebrospinal fluid [22]. This paper therefore addresses a dif-

ferent aspect. Although many applications have testified to the efficiency and accuracy of the GNN's learning ability, a theoretical investigation of its power as a universal approximator is also needed.

The remainder of this paper is organized as follows. In Section 2, a brief introduction of the GNN and of the bipolar GNN [9] (BGNN) are presented. The output of a GNN is computed from the firing probabilities of selected output neurons and these obviously have a range between 0 and 1. One approach to approximate functions with full range on R using the GNN obtains output values greater than 1 by considering the "average potential" [6, 8] of a neuron as the output value, while negative values are obtained by considering networks with "negative and positive neurons" [9]. In this paper we study two extensions of the feedforward GNN with expanded network output range. In Section 3, we prove the approximation capability of these extended GNN models by showing that they are universal function approximators. The last section presents some conclusions.

2 The GNN and Related Models

Consider a GNN [6, 8, 12] with n neurons in which "positive" and "negative" signals circulate. This is a continuous time, discrete state space model. The $i-th$ neuron's state is represented at any time t by its "potential" $k_i(t) \geq 0$, which is a non-negative integral number. Positive signals represent excitation and effect a (+1) operation on the potential of the neuron to which they arrive. Negative signals represent inhibition and effect a (-1) operation on the potential of the neuron where they arrive, if the potential is positive; otherwise they have no effect. If the potential of neuron i is positive it may "fire" in a random sequence with inter-firing times being independent, exponentially distributed random variables of rate $r(i) \geq 0$, sending signals out to other neurons or to outside of the network. When a neuron fires, then the potential of the neuron also drops by 1, depleting the neuron's potential. Signals arriving to a neuron may be exogenous (coming from outside sources). Exogenous excitatory signals arrive to neuron i in a Poisson stream of rate $\Lambda(i)$. Similarly exogenous inhibitory signals arrive to neuron i in a Poisson stream of rate $\lambda(i)$. All these Poisson streams for all values of $i = 1, ..., n$ are independent of each other. When neuron i fires, an excitatory (or positive) signal may go to neuron j with probability $p^+(i,j)$, or as a negative signal with probability $p^-(i,j)$, or it may

depart from the network with probability $d(i)$. We will have:

$$\sum_{j=1}^{m}[p^+(i,j) + p^-(i,j)] + d(i) = 1, i = 1, ..., n. \tag{1}$$

To simplify the notation, in the sequel we will write:

$$\omega^+(i,j) = r(i)p^+(i,j), \tag{2}$$

$$\omega^-(i,j) = r(i)p^-(i,j). \tag{3}$$

Consider the quantities $\lambda^+(i)$ and $\lambda^-(i)$ represent the average arrival rates of positive and negative signals to each neuron i, respectively. The key results about the GNN developed in [6, 8, 12] are summarized below:

Theorem 1. (Proposition 1 in the Appendix of [12]) *There always exists a solution such that* $\lambda^+(i) \geq 0$, $\lambda^-(i) \geq 0$ *to the equations:*

$$\lambda^+(i) = \Lambda(i) + \sum_j q_j \omega^+(j,i), \tag{4}$$

$$\lambda^-(i) = \lambda(i) + \sum_j q_j \omega^-(j,i), \tag{5}$$

for $i = 1, ..., n$ where

$$q_i = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)}. \tag{6}$$

The second key result concerns the specific form taken by the stationary joint probability of network state:

$$p(k) = \lim_{t \to \infty} p(k(t) = k). \tag{7}$$

Theorem 2. (Theorem 1 of [6]) *For an n neuron GNN, let the vector of neuron potentials at time t be* $k(t) = (k_1(t), k_2(t), ..., k_n(t))$, *and let* $k = (k_1, k_2, ..., k_n)$ *be an n-vector of non-negative integers. Then if the* q_i *in (6) satisfy* $0 \leq q_i < 1$, *the stationary joint probability of network state is given by:*

$$p(k) = \prod_{i=1}^{n}(1 - q_i)q_i^{k_i}. \tag{8}$$

Note that if the conditions of the Theorem 2 are satisfied then $p(k_i) = \lim_{t \to \infty} p(k_i(t) = k_i)$, the stationary probability of the state of neuron i, is given by:

$$p(k_i) = (1 - q_i)q_i^{k_i}, \tag{9}$$

and

$$q_i = \lim_{t \to \infty} Prob\{k_i(t) > 0\}. \tag{10}$$

2.1 The Bipolar GNN (BGNN)

In order to represent bipolar patterns and reinforce the associative memory capabilities of the GNN, Gelenbe, Stafylopatis and Likas [9] extended the original model by introducing two types of nodes – positive and negative neurons. The Bipolar GNN (BGNN) can also be viewed as the coupling of two complementary standard GNN models.

In the BGNN the two types of neurons have opposite roles. A positive neuron behaves exactly as a neuron in the original GNN. A negative neuron has a completely symmetrical behavior, namely only negative signals can accumulate at this neuron, and the role of positive signals is to eliminate negative signals which have accumulated in a negative neuron's potential. A positive signal arriving to a negative neuron i cancels a negative signal (adds $+1$ to the neuron's negative potential), and has no effect if $k_i = 0$. This extension is in fact mathematically equivalent to the original GNN described above, with respect to Theorems 1 and 2, as will be summarized below. In the sequel we shall show that the BGNN is a universal approximator for continuous functions.

In the BGNN, the emission of signals from a positive neuron is the same as in the original GNN. Similarly, a negative neuron may emit negative signals. A signal leaving negative neuron i arrives to neuron j as a negative signal with probability $p^+(i,j)$ and as a positive signal with probability $p^-(i,j)$. Also, a signal departs from the network upon leaving neuron i with probability $d(i)$. Other assumptions and denotations retain as in the original model.

Let us consider a BGNN with n nodes. Since negative signals account for the potential of negative neurons, we could use negative values for k_i if neuron i is negative. If we take into account the distinction between positive and negative neurons, Theorems 1 and 2 can be summarized as follows for the BGNN. The flow of signals in the network is described by the following equations:

$$\lambda^+(i) = \Lambda(i) + \sum_{j \in P} q_j \omega^+(j,i) + \sum_{j \in N} q_j \omega^-(j,i), \tag{11}$$

$$\lambda^-(i) = \lambda(i) + \sum_{j \in P} q_j \omega^-(j,i) + \sum_{j \in N} q_j \omega^+(j,i), \tag{12}$$

where we denote by P and N the set of positive and negative neurons respectively, and

$$q_i = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)}, \ i \in P, \tag{13}$$

$$q_i = \frac{\lambda^-(i)}{r(i) + \lambda^+(i)}, \ i \in N. \tag{14}$$

It can be shown that a non-negative solution $\{\lambda^+(i), \lambda^-(i), \ i = 1, ..., n\}$ exists to the above equations. If the $q_i < 1, \ i = 1, ..., n$, then the steady-state joint probability distribution of network state is given by [9]:

$$p(k) = \prod_{i=1}^{n} (1 - q_i) q_i^{|k_i|}, \tag{15}$$

where the quantity q_i is the steady-state probability that node i is "excited". Note the $|k_i|$ exponent in the above product form, since the $k_i's$ can be positive or negative, depending on the polarity of the $i - th$ neuron.

3 Feedforward GNN Models for Function Approximation

All feedforward models considered in this section are guaranteed to have an unique solution for the $q_i, i = 1, ..., n$ as a result of Theorems 2 and 3 of [8]. Thus from now on we do not revisit this issue. Consider a continuous function $f : [0, 1]^s \mapsto R$ of an input vector $X = (x_1, ..., x_s)$. Since an $[0, 1]^s \mapsto R^w$ function can always be separated into a group of w $[0, 1]^s \mapsto R$ functions, we will only consider outputs in one dimension.

For most of this section we will concentrate on networks with a single input $x \in [0, 1]$. Our purpose is to make sure that the various technical steps we take, which are essential simple but which require attention to detail, are clearly understood by the reader. The case where the input to the network is $X = (x_1, ..., x_s)$ and we are approximating a continuous function $f : [0, 1]^s \mapsto R$ is addressed in Section 3.1.

To approximate f, we will construct s-input, 1-output, L-layer feedforward GNN's. We will use the index (l, i) for the $i - th$ neuron at the $l - th$ layer. Furthermore, when we need to specify this, we will denote by M_l the number of neurons in the $l - th$ layer.

The network under consideration is organized as follows:

- In the first layer, i.e. the input layer we set $\Lambda(1, i) = x_i$, $\lambda(1, i) = 0$, $r(1, i) = 1$, so that $q_{1,i} = x_i$, for $i = 1, ..., s$.

- In the l-th layer ($l = 2, ..., L$), $\Lambda(l,i)$, $\lambda(l,i)$, and $r(l,i)$ are adjustable parameters, and $q_{l,i}$ is given by

$$q_{l,i} = \frac{\Lambda(l,i) + \sum_{h<l} \sum_{j \leq M_h} q_{h,j} \omega^+((h,j),(l,i))}{\lambda(l,i) + r(l,i) + \sum_{h<l} \sum_{j \leq M_h} q_{h,j} \omega^-((h,j),(l,i))} \tag{16}$$

  where the connection "weights" $\omega^+(\cdot,\cdot)$ and $\omega^-(\cdot,\cdot)$ are also adjustable parameters.

- In the $L-th$ or output layer there is only one neuron. As suggested in [6] we can use the output function

$$A_{L,1} = \frac{q_{L,1}}{1 - q_{L,1}} \tag{17}$$

  whose physical meaning is that it is the average potential of the output neuron as the output of the network. In this manner, we will have $A_{L,1} \in [0, +\infty)$, rather than just $q_{L,1} \in [0,1]$.

Before we proceed with the developments concerning GNN approximations we will need some technical results.

Lemma 1. *For any continuous function $f : [0,1] \mapsto R$ and for any $\epsilon > 0$, there exists a polynomial*

$$P(x) = c_0 + c_1(\frac{1}{1+x}) + ... + c_m(\frac{1}{1+x})^m, \ 0 \leq x \leq 1, \tag{18}$$

such that the bound $\sup_{x \in [0,1]} |f(x) - P(x)| < \epsilon$ is satisfied.

Proof: This is a direct consequence of Weierstrass' Theorem (see [2], p. 61) which states that for any continuous function $h : [a,b] \mapsto R$, and some $\epsilon > 0$, there exists a polynomial $P(u)$ such that $\sup_{u \in [a,b]} |h(u) - P(u)| < \epsilon$. Now let $u = 1/(1+x)$, $u \in [1/2, 1]$ and select $x = (1-u)/u$ with $h(u) = f(\frac{1-u}{u}) = f(x)$. If $f(x)$ is continuous, then so is $h(u)$ so that there exists an algebraic polynomial of the form

$$P(u) = c_0 + c_1 u + ... + c_m u^m, \ 1/2 \leq u \leq 1, \tag{19}$$

such that $\sup_{u \in [1/2,1]} |h(u) - P(u)| < \epsilon$. Therefore $P(x)$ is given by (18), and $\sup_{x \in [0,1]} |f(x) - P(x)| < \epsilon$. **Q.E.D.**

We now turn to a second technical result concerning the relationship between polynomials of the form (18) and the GNN.

Lemma 2. *Consider a term of the form*

$$(\tfrac{1}{1+x})^v,$$

for $0 \leq x \leq 1$*, and any* $v = 1, ..., m$*. There exists a feedforward GNN with a single output neuron* $(v+1, 1)$ *and input* $x \in [0, 1]$ *such that*

$$q_{v+1,1} = (\frac{1}{1+x})^v. \tag{20}$$

Proof: Construct a feedforward GNN with $v+1$ neurons numbered $(1,1), ...(v+1,1)$. Now set:

- $\Lambda(1,1) = x$, $\Lambda(2,1) = 1/v$, and $\Lambda(j,1) = 0$ for $j = 3, ..., v+1$,
- $\lambda(j,1) = 0$ for all $j = 1, ..., v+1$, and $d(j,1) = 0$ for $j = 1, ..., v$,
- $\omega^-((1,1),(j,1)) = 1/v$, and $\omega^+((1,1),(j,1)) = 0$ for $j = 2, ..., v+1$,
- $r(j,1) = \omega^+((j,1),(j+1,1)) = 1/v$ for $j = 2, ..., v$,
- Finally $d(v+1,1) = 1$.

It is easy to see that $q_{1,1} = x$, and that

$$q_{j+1,1} = (\frac{1}{1+x})^j, \tag{21}$$

for $j = 1, ..., v$ so the Lemma follows. **Q.E.D.**

The next lemma is meant to illustrate a construction process for algebraic expressions using the feedforward GNN.

Lemma 3. *If there exists a feedforward GNN with a single output neuron* $(L, 1)$*, and a function* $g : [0, 1] \mapsto [0, 1]$ *such that:*

$$q_{L,1} = g(x), \tag{22}$$

then there exists an $L+1$ *layer feedforward GNN with a single output neuron* (Q) *such that:*

$$q_O = \frac{g(x)}{1+g(x)}. \tag{23}$$

Proof: The simple proof is by construction. We simply add an additional neuron (Q) the original GNN, and leave all connections in the original GNN unchanged except for the output connections of the neuron $(L, 1)$. Let the firing rate of neuron $(l, 1)$ be $r(L, 1)$. Then:

- $(L,1)$ will now be connected to the new neuron $(L+1,1)$ by $\omega^+((L,1),Q) = r(L,1)/2$, $\omega^-((L,1),Q) = r(L,1)/2$,
- $r(Q) = r(L,1)/2$.

This completes the proof of the Lemma. **Q.E.D.**

The following Lemma shows how an arbitrary polynomial of the form (18) with non-negative coefficients can be realized by a feedforward GNN.

Lemma 4. *Let $P^+(x)$ be a polynomial of the form (18) with the restriction that $c_v \geq 0$, $i = 1, ..., m$. Then there exists a feedforward GNN with a single output neuron (O) such that:*

$$q_O = \frac{P^+(x)}{1 + P^+(x)}, \tag{24}$$

so that the average potential of the output neuron is $A_O = P^+(x)$.

Proof: The proof is by construction. Let C_{MAX} be the largest of the coefficients in $P^+(x)$ and write $P^*(x) = P^+(x)/C_{MAX}$. Let $c_j^* = c_j/C_{MAX} \leq 1$ so that now each term $c_j^* \frac{1}{(1+x)^j}$ in $P^*(x)$ is no greater than 1, $j = 1, ..., m$. We now take m networks of the form of Lemma 2 with $r(j,1) = 1$, $j = 1, ..., m$ and output values

$$q_{j,1} = (\frac{1}{1+x})^j, \tag{25}$$

and connect them to the new output neuron (O) by setting the probabilities $p^+((j,1),O) = c_j^*/2$, $p^-((j,1),O) = c_j^*/2$. Furthermore we set an external positive and negative signal arrival rate $\Lambda(O) = \lambda(O) = c_0^*/2$ and $r(O) = 1/(2C_{MAX}$ for the output neuron. We now have:

$$q_O = \frac{\frac{P^*(x)}{2}}{\frac{1}{2C_{MAX}} + \frac{P^*(x)}{2}}. \tag{26}$$

We now multiply the numerator and the denominator on the right hand side of the above expression by $2C_{MAX}$ to obtain

$$q_O = \frac{P^+(x)}{1 + P^+(x)}. \tag{27}$$

so that which completes the proof of the Lemma. **Q.E.D.**

We now prove another technical lemma which will be of use in proving the approximating power of the "clamped GNN' discussed below.

Lemma 5. *Consider a term of the form*

$\frac{x}{(1+x)^v}$,

for $0 \leq x \leq 1$, *and any* $v = 1, ..., m$. *There exists a feedforward GNN with a single output neuron* $(v+1, 1)$ *and input* $x \in [0, 1]$ *such that*

$$q_{v+1,1} = \frac{x}{(1+x)^v}. \tag{28}$$

The proof is very similar to that of Lemma 2 and will be omitted.

Finally, we state without proof another lemma, very similar to Lemma 4, but which uses terms of the form $x/(1+x)^v$ to construct polynomials. It's proof uses Lemma 5, and follows exactly the same lines as Lemma 4.

Lemma 6. *Let* $P^o(x)$ *be a polynomial of the form*

$$P^o(x) = c_0 + c_1 \frac{x}{1+x} + ... + c_m \frac{x}{(1+x)^m}, \ 0 \leq x \leq 1, \tag{29}$$

with non-negative coefficients, i.e. $c_v \geq 0$, $i = 1, ..., m$. *Then there exists a feedforward GNN with a single output neuron* $(O, +)$ *such that:*

$$q_O = \frac{P^o(x)}{1 + P^o(x)}, \tag{30}$$

so that the average potential of the output neuron is $A_O = P^o(x)$.

The technical results given above pave the way for the use of the Bipolar GNN (BGNN) for approximating continuous functions.

Theorem 3. *For any continuous function* $f : [0, 1] \mapsto R$ *and any* $\epsilon > 0$, *there exists a BGNN with one positive output neuron* $(O, +)$, *one negative output neuron* $(O, -)$, *the input variable* x, *and the output variable* $y(x)$ *such that:*

$$y(x) = A_{[O,+]} + A_{[O,-]}, \tag{31}$$

$$A_{[O,+]} = \frac{q_{[O,+]}}{1 - q_{[O,+]}}, \tag{32}$$

$$A_{[O,-]} = \frac{-q_{[O,-]}}{1 - q_{[O,-]}}, \tag{33}$$

and $\sup_{x \in [0,1]} |f(x) - y(x)| < \epsilon$. *We will say that the BGNN's output uniformly approximates* $f(x)$.

The result is a direct application of Lemmas 1 and 4 and will be omitted.

We can also demonstrate the approximating power of a normal feedforward GNN by just adding a "clamping constant" to the average potential of the output neuron, and using Lemma 4. We will call this extension the Clamped GNN (CGNN) since the additive constant c resembles the clamping level in an electronic clamping circuit.

Theorem 4. *For any continuous function $f : [0,1] \mapsto R$ and any $\epsilon > 0$, there exists a GNN with two output neurons $(O,1)$, $(O,2)$, and a constant c called the clamping constant, resulting in a function $y(x) = A_{O,1} + A_{O,2} + c$ which approximates f uniformly on $[0,1]$ with error less than ϵ.*

Proof: Use Lemma 1 to construct the approximating polynomial of (18), which we write as $P(x) = P^+(x) + P^-(x)$ where $P^+(x)$ only has non-negative coefficients c_v^+, while $P^-(x)$ only has non-positive coefficients c_v^-:

$$c_v^+ = max\{0, c_v\},$$
$$c_v^- = min\{0, c_v\}.$$

Notice that

$$-\frac{1}{(1+x)^i} = 1 - \frac{1}{(1+x)^i} - 1 = \sum_{j=1}^{i} \frac{x}{(1+x)^j} - 1,$$

so that

$$P^-(x) = \sum_{v=1}^{m} |c_v^-| \sum_{j=1}^{v} \frac{x}{(1+x)^j} + \sum_{v=1}^{m} c_v^-. \tag{34}$$

Call $c = c_0 + \sum_{v=1}^{m} c_v^-$ and for some $d_v \geq 0$ write:

$$P(x) = c + \sum_{v=1}^{m} [c_v^+ \frac{1}{(1+x)^v} + d_v \frac{x}{(1+x)^v}]. \tag{35}$$

Let us write $P(x) = c + P^*(x) + P^o(x)$ where both $P^*(x)$ and $P^o(x)$ are polynomials with non-negative coefficients, and

$$P^*(x) = \sum_{v=1}^{m} c_v^+ \frac{1}{(1+x)^v},$$
$$P^o(x) = \sum_{v=1}^{m} d_v \frac{x}{(1+x)^v}.$$

Then by Lemma 6 there are two GNN's whose output neurons $(O,1)$, $(O,2)$ take the values:

$$q_{O,1} = \frac{P^+(x)}{1 + P^+(x)},$$
$$q_{O,2} = \frac{P^o(x)}{1 + P^o(x)}.$$

Clearly, we can consider that these two GNN's constitute one network with two output neurons, and we have $y(x) = c + P^*(x) + P^o(x) = P(x)$, completing the proof. **Q.E.D.**

3.1 Function Approximation for s Input Variables

Now that the process for approximating a one-dimensional continuous function with the BGNN or the CGNN is well understood, let us consider the case of multiple inputs and the continuous function $f : [0,1]^s \mapsto R$. Let us admit our version of the Weierstrass theorem (Lemma 2) in the case for $s-$inputs stating that there is a polynomial:

$$P(x) = \sum_{m_1 \geq 0, ..., m_s \geq 0, \sum_{v=1}^{s} m_v = m} c(m_1, ..., m_s) \Pi_{v=1}^{s} \frac{1}{(1 + x_v)^{m_v}}, \tag{36}$$

with coefficients $c(m_1, ..., m_s)$ which approximates f uniformly. The basic result for approximating f using a BGNN then calls upon the following technical result which is simply an extension of Lemma 2.

Lemma 7. *Consider a term of the form*

$$\frac{1}{(1 + x_{z1})^{m_{z1}}} ... \frac{1}{(1 + x_{zK})^{m_{zK}}}$$

for $0 \leq x_{zj} \leq 1$, *positive integers* $m_{zj} > 0$ *and* $j = 1, ..., K$. *There exists a feedforward GNN with a single output neuron* $(\mu + 1, 1)$ *and input* $x \in [0,1]$ *such that*

$$q_{\mu+1,1} = \frac{1}{(1 + x_{z1})^{m_{z1}}} ... \frac{1}{(1 + x_{zK})^{m_{zK}}}. \tag{37}$$

Proof: Without loss of generality set $m_{z1} \leq m_{z2} \leq ... \leq m_{zK}$. The resulting network is a cascade connection of a set of networks. The first network is identical in structure to the one of Lemma 2, and has $m_{z1} + 1$ neurons numbered $(1,1), ..., (1, m_{z1} + 1)$. Now set:

- $\Lambda(1,1) = x_{z1}$, $\Lambda(1,2) = 1/m_{z1}$, and $\Lambda(1,j) = 0$ for $j = 3, ..., m_{z1} + 1$,
- $\lambda(1,j) = 0$ for all $j = 1, ..., m_{z1} + 1$, and $d(1,j) = 0$ for $j = 1, ..., m_{z1}$,
- $\omega^-((1,1),(1,j)) = 1/m_{z1}$, and $\omega^+((1,1),(1,j)) = 0$ for $j = 2, ..., m_{z1} + 1$,
- $r(1,j) = \omega^+((1,j),(1,j+1)) = 1/m_{z1}$ for $j = 2, ..., m_{z1} + 1$,

- Finally the connection from the first network into the second network is made via $p^+((1, m_{z1}+1), (2,2)) = m_{z1}/m_{z2} \leq 1$, with $d(1, m_{z1}+1) = (1 - m_{z1}/m_{z2})$.

It is easy to see that $q_{1,1} = x_{z1}$, and that

$$q_{1,m_{z1}+1} = \frac{1}{(1+x_{z1})^{m_{z1}}}. \tag{38}$$

The second network has $m_{z2}+1$ neurons numbered $(2,1), ..., (2, m_{z2}+1)$. Now set:

- $\Lambda(2,1) = x_{z2}$ and $\Lambda(1,j) = 0$ for $j = 2, ..., m_{z2}+1$,
- $\lambda(2,j) = 0$ for all $j = 1, ..., m_{z2}+1$, and $d(2,j) = 0$ for $j = 1, ..., m_{z2}$,
- $\omega^-((2,1),(2,j)) = 1/m_{z2}$, and $\omega^+((2,1),(2,j)) = 0$ for $j = 2, ..., m_{z2}+1$,
- $r(2,j) = \omega^+((2,j),(2,j+1)) = 1/m_{z2}$ for $j = 2, ..., m_{z2}+1$,
- The connection from the second network into the third network is made via $p^+((2, m_{z2}+1), (3,2)) = m_{z2}/m_{z3} \leq 1$, with $d(2, m_{z2}+1) = (1 - m_{z2}/m_{z3})$.

It is easy to see that $q_{2,1} = x_{z2}$, and that

$$q_{2,m_{z2}+1} = \frac{1}{(1+x_{z1})^{m_{z1}}} \frac{1}{(1+x_{z2})^{m_{z2}}}. \tag{39}$$

The remaining construction just pursues the same scheme. **Q.E.D.**

Theorem 5. *For any continuous function $f : [0,1]^s \mapsto R$ and any $\epsilon > 0$, there exists a BGNN with one positive output neuron $(O,+)$, one negative output neuron $(O,-)$, s input variables $X = (x_1, ..., x_s)$, and the output variable $y(X)$ such that:*

$$y(X) = A_{[O,+]} + A_{[O,-]}, \tag{40}$$

$$A_{[O,+]} = \frac{q_{[O,+]}}{1 - q_{[O,+]}}, \tag{41}$$

$$A_{[O,-]} = \frac{-q_{[O,-]}}{1 - q_{[O,-]}}, \tag{42}$$

and $\sup_{x \in [0,1]} |f(X) - y(X)| < \epsilon$. We will say that the BGNN's output uniformly approximates $f(X)$.

Proof: The proof follows the proof of Theorem 3, using the polynomial of (36). Lemma 7 establishes that the terms of such a polynomial can be realized by a GNN. We then construct

two polynomials, one with non-negative coefficients only, and the other with negative coefficients, and show how they are realized with the BGNN. We will not go through the steps of the proof since it is a step by step duplicate of the proof of Theorem 3. **Q.E.D.**

In a similar manner we have the following result, which generalizes Theorem 4.

Theorem 6. *For any continuous function* $f : [0,1]^s \mapsto R$ *and any* $\epsilon > 0$*, there exists a GNN with two output neurons* $(O,1)$*,* $(O,2)$*, and a constant* c *called the clamping constant, resulting in a function* $y(X) = A_{O,1} + A_{O,2} + c$ *which approximates* f *uniformly on* $[0,1]^s$ *with error less than* ϵ*.*

4 Conclusions

The neural network model [6, 8, 12] discussed in this paper is a novel spiked model which represents more closely the manner in which signals are transmitted in a biophysical neural network where they mostly travel as voltage (action potential) spikes rather than as fixed analog levels. The model has a closed form solution for network state even in the recurrent case, which leads to efficient computational techniques. It has been applied to many real-world problems [9, 11, 12, 13, 17, 21, 20, 22, 23], and has been shown to be able to efficiently carry out useful tasks such as optimization, learning, associative memory, pattern recognition. However, the general approximating properties of this network had not been previously established. In this paper we have studied the approximation of arbitrary continuous functions on $[0,1]^s$ using this network model. We have shown that two extended versions of the model (the clamped GNN and the bipolar GNN) have this property. The proofs are by construction of appropriate GNN's which realize the approximating polynomials based on Weierstrass' Theorem. We do not address here the minimal representations of these approximating networks, although we present straightforward canonical constructions of the appropriate GNN's in each case based upon sub-networks which implement the terms of the approximating polynomial. Future work will address the design of other canonical structures, such as a three-layer GNN's, for function approximation.

REFERENCES

[1] A. L. Hodgkin and A. F. Huxley "A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes", *J. Physiology (London)*, vol. 117, pp. 500-544, 1952.

[2] M. J. D. Powell, *Approximation Theory and Methods*. Cambridge University Press, 1981.

[3] V. Vapnik, "Estimation of Dependencies based on Empirical Data", Springer Verlag, 1982.

[4] K. Funahashi, "On the approximate realization of continuous mapping by neural network," *Neural Networks*, vol. 2, pp. 183-192, 1989.

[5] E. Gelenbe "Réseaux stochastiques ouverts avec clients négatifs et positifs, et réseaux neuronaux", *Comptes-Rendus Acad. Sci. Paris*, t. 309, Série II, pp. 979-982, 1989.

[6] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Computation*, vol. 1, no. 4, pp. 502-511, 1989.

[7] E. Gelenbe "Réseaux neuronaux aléatoires stables", *Comptes-Rendus Acad. Sci.*, t. 310, Série II, pp. 177-180, 1990.

[8] E. Gelenbe, "Stability of the random neural network model," *Neural computation*, vol. 2, no. 2, pp. 239-247, 1990.

[9] E. Gelenbe, A. Stafylopatis, and A. Likas, "Associative memory operation of the random network model," in *Proc. Int. Conf. Artificial Neural Networks*, Helsinki, pp. 307-312, 1991.

[10] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, pp. 246-257, 1991.

[11] E. Gelenbe, F. Batty, "Minimum cost graph covering with the random neural network," in *Computer Science and Operations Research*, New York, Pergamon, pp. 139-147, 1992.

[12] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154-164, 1993.

[13] E. Gelenbe, V. Koubi, F. Pekergin, "Dynamical random neural network approach to the traveling salesman problem," in *Proc. IEEE Symp. Syst., Man, Cybern.*, pp. 630-635, 1993.

[14] H. Leung and S. Haykin, "Rational function neural network," *Neural Computation*, vol. 5, pp. 928-938, 1993.

[15] M. Abeles "Firing rates and well-timed events in the cerebral cortex", in *E. Domany, J.L. van Hemmen and K. Schulten (eds.), "Models of Neural Networks II"*, pp. 121-138, Springer Verlag, 1994.

[16] W. Gerstner and J. L. van Hemmen "Coding and information processing in neural networks", in *E. Domany, J.L. van Hemmen and K. Schulten (eds.), "Models of Neural Networks II"*, pp. 1-118, Springer Verlag, 1994.

[17] A. Ghanwani, "A qualitative comparison of neural network models applied to the vertex covering problem," *Elektrik*, vol. 2, no. 1, pp. 11-18, 1994.

[18] J. J. Buckley and Y. Hayashi, "Can fuzzy neural nets approximate continuous fuzzy functions?" *Fuzzy Sets and Systems*, vol. 61, pp. 43-51, 1994.

[19] J. Zhang, G. G. Walter, Y. Miao, and W. N. W. Lee, "Wavelet neural networks for function learning," *IEEE Trans. Signal Processing*, vol. 43, no. 6, pp. 1485-1496, 1995.

[20] E. Gelenbe, C. Cramer, M. Sungur, P. Gelenbe "Traffic and video quality in adaptive neural compression", *Multimedia Systems*, Vol. 4, pp. 357–369, 1996.

[21] C. Cramer, E. Gelenbe, H. Bakircioglu "Low bit rate video compression with neural networks and temporal subsampling," *Proceedings of the IEEE*, Vol. 84, No. 10, pp. 1529–1543, October 1996.

[22] E. Gelenbe, T. Feng, K.R.R. Krishnan "Neural network methods for volumetric magnetic resonance imaging of the human brain," *Proceedings of the IEEE*, Vol. 84, No. 10, pp. 1488–1496, October 1996.

[23] E. Gelenbe, A. Ghanwani, V. Srinivasan "Improved neural heuristics for multicast routing", *IEEE Journal of Selected Areas of Communications*, Vol. 15, No. 2, pp. 147-155, February 1997.

[24] E. Gelenbe, G. Pujolle "Introduction to Networks of Queues", J. Wiley & Sons, Chichester and New York, 1998.