Function Approximation by Random Neural Networks with a Bounded Number of Layers

Erol Gelenbe, *Fellow IEEE* *
School of Electrical Engineering & Computer Science
University of Central Florida
Orlando, FL 32816
erol@cs.ucf.edu Fax: 407 823 5419

Zhi-Hong Mao and Yan-Da Li †
Department of Automation
Tsinghua University
Beijing 100084
P.R. China
maozh@jerry.au.tsinghua.edu.cn

February 14, 2001

Abstract

This paper discusses the function approximation properties of the "Gelenbe" random neural network (GNN) [5, 6, 9]. We use two extensions of the basic model: the bipolar GNN (BGNN) [7] and the clamped GNN (CGNN). We limit the networks to being feedforward and consider the case where the number of hidden layers does not exceed the number of input layers. With these constraints we show that the feedforward CGNN and the BGNN with s hidden layers (total of $s + 2$ layers) can uniformly approximate continuous functions of s variables.

1 Introduction

A novel neural network model – the GNN or "Gelenbe's Random Neural Network" [5, 6, 9] – has had significant applications in various engineering areas [7, 8, 10, 11, 12, 13, 14, 15], using the network's learning algorithm [9] or its ability to act as an optimizing network. These random neural network differ significantly from standard connexionist models in that information travels between neurons in this model in the form of random spike trains, and network state is represented by the probability distributions that the n neurons in the network are excited. These

*This author's research was supported by the Office of Naval Research under Grant No. N00014-97-1-0112, and is currently supported by NAWC Grants No. N61339-97-K-005 and N61339-00-K-002.

†These authors' work was supported by the National Science Foundation of the P.R. of China under Grant No. 69682010.

Proceedings 17th European Simulation Multiconference
(c) SCS Europe BVBA, 2003 ISBN 3-936150-25-7

models have a mathematical structure which is significantly different from that of the sigmoidal connexionist model, the Hopfield model, or the Boltzman machine [3]. Thus the approximation capability of these networks also needs to be established in a manner distinct from that of previous models [4]. In particular, the "Gelenbe" random neural network model [5, 6, 9] does not use sigmoid functions which are basic to the standard models' approximation capabilities.

In recent work [16] we have studied the approximation of arbitrary continuous functions on $[0,1]^s$ using the GNN. We have shown that the clamped GNN and the bipolar GNN [7] have the universal approximation property, using a constructive method which actually exhibits networks constructed from a polynomial approximation of the function to be approximated. However the constructions in [16] place no restrictions on the structure of the networks except for limiting them to being feedforward. For instance, in [16] we were not able to limit the size of the network as a function of other meaningful characteristics such as the number of input variables or the number of layers.

In this paper we will discuss the design of GNN approximators with a bounded number of layers. In Section 2, a brief introduction to the GNN and the bipolar GNN (BGNN) is given. In Section 3, we establish the technical premises for our main results. Then in Section 4 we prove the universal approximation capability of the feedforward BGNN and CGNN when the number of hidden layers does not exceed the number of input variables. The last section presents conclusions.

2 The GNN and its Extensions

Consider a GNN [5, 6, 9] with n neurons in which "positive" and "negative" signals circulate. The $i-th$ neuron's state is represented at any time t by its "potential" $k_i(t)$, which is a non-negative integer. In the RNN (Gelenbe (1989,90) [5, 6]) signals in the form of spikes of unit amplitude circulate among the neurons. Positive signals represent excitation and negative signals represent inhibition. Each neuron's state is a non-negative integer called its potential, which increases when an excitation signal arrives to it, and decreases when an inhibition signal arrives. An excitatory spike is interpreted as a "$+1$" signal at a receiving neuron, while an inhibitory spike is interpreted as a "-1" signal. Neural potential also decreases when the neuron fires. Thus a neuron i emitting a spike, whether it be an excitation or an inhibition, will lose potential of one unit, going from some state whose value is k_i to the state of value $k_i - 1$. In general, this is a "recurrent network" model, *i.e.* a network which is allowed to have feedback loops of arbitrary topology.

The state of the n-neuron network at time t, is represented by the vector of non-negative integers $k(t) = (k_1(t), \ldots, k_n(t))$, where $k_i(t)$ is the potential or integer state of neuron i. We will denote by k and k_i arbitrary values of the state vector and of the i-th neuron's state. Neuron i will "fire" (i.e. become excited and send out spikes) if its potential is *positive*. The spikes will then be sent out at a rate $r(i) \geq 0$, with independent, identically and exponentially distributed inter-spike intervals. Spikes will go out to some neuron j with probability $p^+(i,j)$ as excitatory signals, or with probability $p^-(i,j)$ as inhibitory signals. A neuron may also send signals out of the network with probability $d(i)$, and $d(i) + \sum_{j=1}^{n} [p^+(i,j) + p^-(i,j)] = 1$. Figure **??** shows the representation of a neuron in the RNN.

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 $\lambda(i)$. These different

Poisson streams for $i = 1, ..., n$ are independent of each other. To simplify the notation, in the sequel we will write:

$$\omega^+(i,j) = r(i)p^+(i,j), \tag{1}$$
$$\omega^-(i,j) = r(i)p^-(i,j). \tag{2}$$

The state transitions of the network are represented by Chapman-Kolmogorov equations [1] for the probability distribution:

$$p(k,t) = Prob[k(t) = k], \tag{3}$$

where $k = (k_1, \ldots, k_n)$ denotes a particular value of the state vector.

Let $\lambda^+(i)$ and $\lambda^-(i)$ denote the average arrival rates of positive and negative signals to each neuron i. The key results about the GNN developed in [5, 6, 9] are summarized below. **Theorem 1.** (Proposition 1 in the Appendix of [9]) *There always exists a non-negative solution* $(\lambda^+(i) \geq 0,\ \lambda^-(i) \geq 0)$ *to the equations:*

$$\lambda^+(i) = \Lambda(i) + \sum_{j=1}^{n} q_j \omega^+(j,i), \tag{4}$$

$$\lambda^-(i) = \lambda(i) + \sum_{j=1}^{n} q_j \omega^-(j,i), \tag{5}$$

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

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

The next important result concerns the stationary joint probability distribution of network state:

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

Theorem 2. (Theorem 1 of [5]) *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 Theorem 2 are satisfied then the stationary probability distribution of the state of neuron i denoted by $p(k_i) = \lim_{t \to \infty} p(k_i(t) = k_i)$, 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 BGNN Model

In order to represent bipolar patterns taking values such as $\{+1, -1\}$, and to strengthen the associative memory capabilities of the GNN, in some early work Gelenbe, Stafylopatis and Likas [7] extended the original model by introducing the artifact of "positive and negative" neurons. The resulting 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 arriving to a negative neuron 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 the specific form taken by the stationary solution (Theorems 1 and 2). However the use of both positive and negative neurons allows the BGNN to become convenient universal approximator for continuous functions because of the possibility of using both positive and negative valued functions of the input variables. Let P and N denote, respectively, the indeces of the positive and negative neurons in the network. In the BGNN the state of the network is represented by the vector $k(t) = (k_1(t), \ldots, k_n(t))$ so that $k_i(t) \geq 0$ if $i \in P$ and $k_i(t) \leq 0$ if $i \in N$.

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 will 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:

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

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

and

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

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

Using a direct extension of the results for the conventional GNN, 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 [7]:

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

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. In the sequel we will consider how the BGNN, as well as a simpler extension of the feedforward (i.e. non-recurrent) GNN, can be used to approximate arbitrary continuous functions.

Another extension of the GNN – the clamped GNN (CGNN) – will be introduced in Section 3.3.

3 Approximation of Functions of One Variable by the GNN with a Bounded Number of Layers

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 [6, 9]. 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 distinct functions $[0, 1]^s \mapsto R$, we will only consider outputs in one dimension. The sequel of this paper is therefore devoted to how a continuous function $f : [0, 1]^s \mapsto R$ can be approximated by neural networks derived from the GNN model. 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
$$q_{l,i} = \frac{\Lambda(l,i) + \sum_{1\le h<l} \sum_{1\le j\le M_h} q_{h,j}\omega^+((h,j),(l,i))}{\lambda(l,i) + r(l,i) + \sum_{1\le h<l} \sum_{1\le j\le M_h} q_{h,j}\omega^-((h,j),(l,i))} \tag{16}$$
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 [5] 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 $q_{L,1} \in [0, 1]$.

3.1 Technical Premises

Before we proceed with the developments concerning GNN approximations we need some technical results. They are similar to some technical results used in [16] concerning continuous and

bounded functions $f : [0,1] \mapsto R$ for a scalar variable x. The generalization to $f : [0,1]^s \mapsto R$ is direct and will be examined in Section 4. The proofs are given in the Appendix.

Lemma 1. *For any continuous and bounded $f : [0,1] \mapsto R$ 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 $\sup_{x\in[0,1]} |f(x) - P(x)| < \epsilon$ is satisfied.

The second technical result concerns the relationship between polynomials of the form (18) and the GNN.

Lemma 2. *Consider a term of the form*

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

for $0 \leq x \leq 1$, and any $v = 1, 2, \cdots$. There exists a feedforward GNN with a single output neuron $(v+1, 1)$ and input $x \in [0,1]$ such that

$$q_{v+1,1} = (\frac{1}{1+x})^v. \tag{19}$$

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

Lemma 3. *Let $P^+(x)$ be a polynomial of the form (18) with the restriction that $c_v \geq 0$, $v = 1, ..., m$. Then there exists a feedforward GNN with a single output neuron (O) such that:*

$$q_O = \frac{P^+(x)}{1 + P^+(x)}, \tag{20}$$

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

The fourth technical result will be of use in proving the approximating power of the "clamped GNN' discussed below.

Lemma 4. *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

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

We state without proof another lemma, very similar to Lemma 3, but which uses terms of both forms of $1/(1+x)^v$ and $x/(1+x)^v$ to construct polynomials. It's proof uses Lemma 3 and 4, and follows exactly the same lines as Lemma 3.

Lemma 5. *Let $P^o(x)$ be a polynomial of the form*

$$P^o(x) = c_0 + \sum_{v=1}^{m}[c_v \frac{1}{(1+x)^v} + d_v \frac{x}{(1+x)^v}],\ 0 \le x \le 1, \tag{22}$$

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

$$q_O = \frac{P^o(x)}{1 + P^o(x)}, \tag{23}$$

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

The next lemma is a technical premise of Lemma 7.

Lemma 6. *For any $(\frac{1}{1+x})^i$ $(0 \le x \le 1, i = 1, 2, \cdots)$ and for any $\epsilon > 0$, there exists a function*

$$P_1(x) = b_0 + \frac{b_1}{x + a_1} + \frac{b_2}{x + a_2} + \cdots + \frac{b_r}{x + a_r},\ 0 \le x \le 1, \tag{24}$$

where $a_k > 0$, $k = 1, ..., r$, such that $\sup_{x\in[0,1]} |(\frac{1}{1+x})^i - P_1(x)| < \epsilon$ is satisfied.

Proof: We proceed by induction. For $i = 1$, the conclusion obviously holds. Now assume it is true for $i = j$, i.e., for any $\epsilon > 0$, there exists a

$$P^{(j)}(x) = b_0^{(j)} + \frac{b_1^{(j)}}{x + a_1^{(j)}} + \frac{b_2^{(j)}}{x + a_2^{(j)}} + \cdots + \frac{b_m^{(j)}}{x + a_m^{(j)}},\ 0 \le x \le 1, \tag{25}$$

where $a_k^{(j)} > 0$, $k = 1, ..., m$, such that $\sup_{x\in[0,1]} |(\frac{1}{1+x})^j - P^{(j)}(x)| < \epsilon$.

Then for $i = j + 1$,

$$(\frac{1}{1+x})^{j+1} = (\frac{1}{1+x})^j (\frac{1}{1+x}) = b_0^{(j)} \frac{1}{1+x} + \sum_{k=1}^{m} \frac{b_k^{(j)}}{x + a_k^{(j)}} \frac{1}{1+x}. \tag{26}$$

When $a_k^{(j)} \ne 1$,

$$\frac{b_k^{(j)}}{x + a_k^{(j)}} \frac{1}{1+x} = \frac{b_k^{(j)}}{a_k^{(j)} - 1} (\frac{1}{1+x} - \frac{1}{x + a_k^{(j)}}) \tag{27}$$

which is in the form of (24). When $a_k^{(j)} = 1$,

$$(\frac{1}{1+x})^2 = \lim_{\eta\to 0} \frac{1}{(1 - \eta + x)(1 + \eta + x)} = \lim_{\eta\to 0} \frac{1}{2\eta} (\frac{1}{1 - \eta + x} - \frac{1}{1 + \eta + x}) \tag{28}$$

which can be arbitrarily approximated by a function of the form (24).

Therefore $(\frac{1}{1+x})^{j+1}$ can also be approximated by a function in the form of (24). Through mathematical induction, the conclusion holds for any $i = 1, 2, \cdots$. **Q.E.D.**

The following lemma is the preparation for the construction of a single-hidden-layered BGNN for the approximation of one dimensional continuous function.

Lemma 7. *For any continuous function $f : [0, 1] \mapsto R$ and for any $\epsilon > 0$, there exists a function $P_1(x)$ in the form of (24) such that $\sup_{x\in[0,1]} |f(x) - P_1(x)| < \epsilon$ is satisfied.*

Proof: This is a direct consequence of Lemma 1 and Lemma 6. **Q.E.D.**

3.2 BGNN Approximation of Continuous Functions of One Variable

The technical results given above now pave the way for the use of the Bipolar GNN (BGNN) with a bounded number of layers. Specifically in Theorem 4, we show that a BGNN with a single hidden layer can uniformly approximate functions of one variable. The multivariable case is discussed in Section 4.

Let us first recall a result from [16] concerning the case when the number of layers is not bounded.

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:*

$$y(x) = A_{O,+} + A_{O,-}, \tag{29}$$

$$A_{O,+} = \frac{q_{O,+}}{1 - q_{O,+}}, \tag{30}$$

$$A_{O,-} = \frac{-q_{O,-}}{1 - q_{O,-}}, \tag{31}$$

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 result is a direct application of Lemmas 1 and 3. Apply Lemma 1 to f and express the approximating polynomial as $P(x) = P^+(x) + P^-(x)$ so that the coefficients of $P^+(x)$ are nonnegative, while the coefficients of $P^-(x)$ are negative:

$$P^+(x) = \sum_{i=1}^{m} \max\{0, c_i\}(\frac{1}{1+x})^i, \tag{32}$$

$$P^-(x) = \sum_{i=1}^{m} \min\{0, c_i\}(\frac{1}{1+x})^i. \tag{33}$$

Now simply apply Lemma 3 to obtain the feedforward GNN with an output neuron $(O,+)$ whose value is

$$q_{O,+} = \frac{P^+(x)}{1 + P^+(x)}, \tag{34}$$

and the average potential of the output neuron is $A_{O,+} = P^+(x)$. Similarly, using the nonnegative polynomial $|P^-(x)|$ construct a feedforward BGNN which has positive neurons throughout, except for its output neuron, along the ideas of Lemma 4. It's output neuron $(O,-)$ however is a negative neuron, yet all the parameter values are the same as those prescribed in Lemma 4 for the output neuron, as they relate to the polynomial $|P^-(x)|$. Thus the output neuron takes the value

$$q_{O,-} = \frac{|P^-(x)|}{1 + |P^-(x)|}, \tag{35}$$

and the average potential of the output neuron is: is $A_{O,-} = -|P^-(x)|$, completing the proof. **Q.E.D.**

The next theorem shows the approximation capability of a BGNN with a single hidden layer.

Theorem 4. *For any continuous function $f : [0,1] \mapsto R$ and any $\epsilon > 0$, there exists a BGNN of three layers (only one hidden layer), one positive output neuron $(O,+)$, one negative output*

neuron $(O,-)$*, the input variable* x*, and the output variable* $y(x)$ *determined by (29) such that* $\sup_{x\in[0,1]}|f(x)-y(x)|<\epsilon$.

Proof: The result is obtained by using Lemma 7. Applying Lemma 7 to f we express the approximating function as $P_1(x)=P_1^+(x)+P_1^-(x)$ so that the coefficients of $P_1^+(x)$ are non-negative, while the coefficients of $P_1^-(x)$ are negative:

$$P_1^+(x)=\max\{0,b_0\}+\sum_{k=1}^{r}\max\{0,b_k\}\frac{b_k}{x+a_k}, \tag{36}$$

$$P_1^-(x)=\min\{0,b_0\}+\sum_{k=1}^{r}\min\{0,b_k\}\frac{b_k}{x+a_k}. \tag{37}$$

Now construct a BGNN of three layers: one output layer with one positive output neuron $(O,+)$ and one negative output neuron $(O,-)$ in it, one input layer with one input neuron $(1,1)$ in it, and one hidden layer with r neurons $(2,1)$,...,$(2,r)$ in it. Now set:

- $\Lambda(1,1)=x,\ \lambda(1,1)=0,\ r(1,1)=1,\ d(1,1)=0,$
- $\omega^+((1,1),(2,k))=0,\ \omega^-((1,1),(2,k))=1/r,$
  $r(2,k)=a_k/r,\ \Lambda(2,k)=a_k/r,\ \lambda(2,k)=0,$ for $k=1,\cdots,r,$
- $p^+((2,k),(O,+))=p^-((2,k),(O,+))=(\max\{b_k,0\}r)/(2a_k^2C_{MAX}),$
  $p^+((2,k),(O,-))=p^-((2,k),(O,-))=(|\min\{b_k,0\}|r)/(2a_k^2C_{MAX}),$
  for $k=1,\cdots,r,$ where $C_{MAX}=\max\{1,|b_0|,\frac{|b_k|r}{a_k^2},k=1,...,r\},$
- $\Lambda(O,+)=\lambda(O,+)=\max\{b_0,0\}/(2C_{MAX}),\ r(O,+)=1/(2C_{MAX}),$
  $\Lambda(O,-)=\lambda(O,-)=|\min\{b_0,0\}|/(2C_{MAX}),\ r(O,-)=1/(2C_{MAX}).$

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

$$q_{2,k}=\frac{a_k}{a_k+x},\ k=1,...,r, \tag{38}$$

$$q_{O,+}=\frac{\frac{P^+(x)}{2C_{MAX}}}{\frac{1}{2C_{MAX}}+\frac{P^+(x)}{2C_{MAX}}}=\frac{P^+(x)}{1+P^+(x)}, \tag{39}$$

$$q_{O,-}=\frac{\frac{|P^-(x)|}{2C_{MAX}}}{\frac{1}{2C_{MAX}}+\frac{|P^-(x)|}{2C_{MAX}}}=\frac{|P^-(x)|}{1+|P^-(x)|}. \tag{40}$$

Therefore, $A_{O,+}=P^+(x)$, $A_{O,-}=-|P^-(x)|$, and $y(x)=P_1(x)$, completing the proof. **Q.E.D.**

3.3 CGNN Approximation of Continuous Functions of One Variable

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. We call this extension

the "clamped GNN (CGNN)" since the additive constant c resembles the clamping level in an electronic clamping circuit. Let us first see the corresponding result from our previous work [16].

Theorem 5. *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, 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 (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

$$P^-(x) = \sum_{v=1}^{m} |c_v^-| \sum_{j=1}^{v} \frac{x}{(1+x)^j} + \sum_{v=1}^{m} c_v^-. \tag{41}$$

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

$$P(x) = c + \sum_{v=1}^{m} [c_v^+ \frac{1}{(1+x)^v} + d_v \frac{x}{(1+x)^v}]. \tag{42}$$

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 5 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.**

This result can be extended to the CGNN with only one output neuron by applying Lemma 5. However let us first consider the manner in which a positive "clamping constant" $c > 0$ can be added to the average potential of an output neuron of a GNN using the ordinary structure of the network.

Remark 1 (Adding a Positive Clamping Constant). *Consider a GNN with an output neuron $\hat{q}$ and an input vector x which realizes the function $\hat{q}(x) = P(x)$. Then there is another GNN with output neuron $Q(x)$ which, for real $c > 0$ realizes the function:*

$$Q(x) = \frac{P(x) + c}{1 + P(x) + c} \tag{43}$$

and hence whose average potential is $P(x) + c$. More generally we can exhibit a GNN with output neuron $Q_1(x)$ whose average potential is $bP(x) + c$, for $b > 0, c > 0$.

Proof: The proof is by construction. We first take the output of the neuron of the original network (whose firing rate is denoted $2r$), and feed it into a new neuron with probability 0.5 as an excitatory signal and with probability 0.5 as an inhibitory signal. We set the firing rate of the new neuron to r, and introduce additional exogenous inhibitory and excitatory arrivals to the new neuron, both of rate rc. As a result we have:

$$\begin{aligned} Q(x) &= \frac{rP(x) + rc}{r + rP(x) + rc}, \\ &= \frac{P(x) + c}{1 + P(x) + c}. \end{aligned}$$

As a result, the new neuron's average potential is:

$$\frac{Q(x)}{1 - Q(x)} = P(x) + c.$$

and we have been thus able to obtain a new neuron with an added positive "clamping constant" c with respect to the average potential $P(x)$ of the original neuron. The extension to a neuron with average potential $bp(x) + c$ is straightforward. Let the additional neurons firing rate be $R > 0$ rather than r and take its exogenous excitatory and inhibitory arrival rates to be Rc. We then obtain:

$$\begin{aligned} Q(x) &= \frac{rP(x) + Rc}{R + rP(x) + Rc}, \\ &= \frac{\frac{r}{R}P(x) + c}{1 + \frac{r}{R}P(x) + c}, \end{aligned}$$

so that if we call $b = \frac{r}{R}$ this leads to an average potential of $bP(x) + c$. **Q.E.D.**

Theorem 6. *For any continuous function $f : [0, 1] \mapsto R$ and any $\epsilon > 0$, there exists a GNN with one output neuron (O), and a constant c, resulting in a function $y(x) = A_O + c$ 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^-:

$$\begin{aligned} c_v^+ &= max\{0, c_v\}, \\ c_v^- &= min\{0, c_v\}. \end{aligned}$$

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

$$P^-(x) = \sum_{v=1}^{m} |c_v^-| \sum_{j=1}^{v} \frac{x}{(1+x)^j} + \sum_{v=1}^{m} c_v^-. \tag{44}$$

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

$$P(x) = c + \sum_{v=1}^{m} [c_v^+ \frac{1}{(1+x)^v} + d_v \frac{x}{(1+x)^v}]. \tag{45}$$

Let us write $P(x) = c + P^o(x)$ where $P^o(x)$ is a polynomial with non-negative coefficients. Then by Lemma 5 there is a GNN whose output neurons (O) takes the value:

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

Clearly, we can consider that this GNN constituts one network with only one output neuron, and we have $y(x) = c + P^o(x) = P(x)$, completing the proof. **Q.E.D.**

The next theorem shows that a CGNN with a single hidden layer is also a universal approximator to continuous functions on $[0, 1]$. We omit the proof, which follows closely the approach used in the proofs of Theorems 4 and 6.

Theorem 7. *For any continuous function $f : [0, 1] \mapsto R$ and any $\epsilon > 0$, there exists a GNN of three layers (only one hidden layer), one output neuron (O), and a constant c called the clamping constant, resulting in a function $y(x) = A_O + c$ which approximates f uniformly on $[0, 1]$ with error less than ϵ.*

4 Approximation of Continuous Functions of s Variables

Now that the process for approximating a one-dimensional continuous functions with the BGNN or the CGNN having a single hidden layer is well understood, consider the case of continuous functions of s variables, i.e. $f : [0, 1]^s \mapsto R$. As a starting point, consider the straightforward extension of Lemma 1 to the case of $s-$inputs such that there is a polynomial:

$$P(x) = \sum_{m_1 \geq 0, ..., m_s \geq 0, \sum_{v=1}^{s} m_v = m} c(m_1, ..., m_s) \Pi_{v=1}^{s} \frac{1}{(1+x_v)^{m_v}}, \tag{46}$$

with coefficients $c(m_1, ..., m_s)$ which approximates f uniformly. We now extend Lemma 2 to Lemma 8 and Theorem 8 which are given below.

Lemma 8. *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

$$q_{\mu+1,1} = \frac{1}{(1 + x_{z1})^{m_{z1}}} ... \frac{1}{(1 + x_{zK})^{m_{zK}}}. \tag{47}$$

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

$$q_{1,m_{z1}+1} = \frac{1}{(1+x_{z1})^{m_{z1}}}. \tag{48}$$

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

$$q_{2,m_{z2}+1} = \frac{1}{(1+x_{z1})^{m_{z1}}} \frac{1}{(1+x_{z2})^{m_{z2}}}. \tag{49}$$

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

Theorem 8. *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:*

$$y(X) = A_{O,+} + A_{O,-}, \tag{50}$$

$$A_{O,+} = \frac{q_{O,+}}{1 - q_{O,+}}, \tag{51}$$

$$A_{O,-} = \frac{-q_{O,-}}{1 - q_{O,-}}, \tag{52}$$

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 (46). 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.**

We now extend Lemma 7 to the case of s-inputs.

Lemma 9. *For any continuous function $f : [0,1]^s \mapsto R$ and for any $\epsilon > 0$, there exists a function of the form*

$$P_s(x) = \sum_{i=1}^{r} \sum_{0 \le m_1 \le 1, ..., 0 \le m_s \le 1} b(m_1, ..., m_s, i) \Pi_{v=1}^{s} \frac{1}{(a_{v,i} + x_v)^{m_v}}, \tag{53}$$

where $a_{v,i} > 0$, $v = 1, ..., s$, $i = 1, 2, ...$, such that $\sup_{x \in [0,1]} |f(x) - P_s(x)| < \epsilon$ is satisfied.

Proof: This is simply an extension of Lemma 7. **Q.E.D.**

As a consequence we can now establish the following general result.

Theorem 9. *For any continuous function $f : [0,1]^s \mapsto R$ and any $\epsilon > 0$, there exists a BGNN of no more than $s + 2$ layers (s hidden layers), one positive output neuron $(O, +)$, one negative output neuron $(O, -)$, s input variables $X = (x_1, ..., x_s)$, and the output variable $y(X)$ determined by (50) such that $\sup_{x \in [0,1]} |f(X) - y(X)| < \epsilon$.*

Proof: The proof is by construction. By Lemma 9, we only need to find an appropriate BGNN of the form as described in Theorem 9 to realize any function of the form (53). We construct a BGNN with s input neurons $(1,1)$,..., $(1,s)$, one positive output neuron $(O,+)$, one negative output neuron $(O,-)$, and M parallel sub-networks between the input layer and the output layer, where

$$M \equiv \sum_{i=1}^{r} \sum_{0 \le m_1 \le 1, ..., 0 \le m_s \le 1} 1(b(m_1, ..., m_s, i) \neq 0), \tag{54}$$

$1(X) = 1$ when X is true otherwise $1(X) = 0$. Each sub-network is a cascade connection of no more than s neurons. The output of the last neuron of each sub-network takes the value in proportion to each term in function (53).

Without loss of generality, we consider a term of the form

$$\frac{1}{a_{z1} + x_{z1}} ... \frac{1}{a_{zK} + x_{zK}} \tag{55}$$

where $a_{z1} \ge a_{z2} \ge \cdots \ge a_{zK}$. Now we want to construct a sub-network which has K neurons and of which the last neuron's output takes the value in proportion to the term. Number the K neurons as $(2,1)$, $(3,1)$,..., $(K+1,1)$, and set:

- $\Lambda(1,i) = x_i$, $\lambda(1,i) = 0$, $r(1,i) = 1$, for $i = 1, ..., s$,
- $\omega^+((1,z1),(2,1)) = 0$, $\omega^-((1,z1),(2,1)) = 1/M$,
- $r(2,1) = a_{z1}/M$, $\Lambda(2,1) = a_{z1}/M$, $\lambda(2,1) = 0$.

It is easy to see that

$$q_{2,1} = \frac{a_{z1}}{a_{z1} + x_{z1}}. \tag{56}$$

Then set:

- $p^+((k,1),(k+1,1)) = a_{zK}/a_{z(K-1)}$, for $k = 2, ..., K$,
- $\omega^+((1,zk),(k+1,1)) = 0$, $\omega^-((1,zk),(k+1,1)) = 1/M$, for $k = 2, .., K$,
- $r(k+1,1) = a_{zk}/M$, $\Lambda(k+1,1) = 0$, $\lambda(k+1,1) = 0$, for $k = 2, .., K$.

We will find

$$q_{3,1} = \frac{a_{z1}a_{z2}}{(a_{z1} + x_{z1})(a_{z2} + x_{z2})}, \tag{57}$$

$$\cdots,$$

$$\frac{a_{z1} \cdots a_{zK}}{(a_{z1} \ + \ x_{z1}) \cdots (a_{zK} \ + \ x_{zK})} \tag{58}$$

which is in proportion to (55).

Next we connect all the last neurons of the sub-networks to $(O, +)$ or $(O, -)$. The parameter setting follows the steps in the proof of Theorem 4 which connect the neurons in the hidden layer to the output neurons. Since the sub-networks are parallel and each sub-network contains of no more than s neurons, there are totally no more than s hidden layers in this constructed BGNN. Thus, we complete the construction. **Q.E.D.**

We can now obtain Theorems 10 and 11, which generalize Theorems 6 and 7, in a similar manner.

Theorem 10. *For any continuous function $f : [0,1]^s \mapsto R$ and any $\epsilon > 0$, there exists a GNN with one output neuron (O), and a constant c called the clamping constant, resulting in a function $y(X) = A_O \ + \ c$ which approximates f uniformly on $[0,1]^s$ with error less than ϵ.*

Theorem 11. *For any continuous function $f : [0,1]^s \mapsto R$ and any $\epsilon > 0$, there exists a GNN of no more than $s+2$ layers (s hidden layers), one output neuron (O), and a constant c called the clamping constant, resulting in a function $y(X) = A_O \ + \ c$ which approximates f uniformly on $[0,1]^s$ with error less than ϵ.*

5 Conclusions

The approximation of functions by neural networks is central the learning theory of neural networks. It is also a key to many applications of neural networks such as pattern recognition, data compression, time series prediction, adaptive control, etc..

The random neural network introduced and developped in [5, 6, 9, 17] differs significantly from standard connexionist models in that information travels between neurons in this model in the form of random spike trains, and network state is represented by the joint probability distributions that the n neurons in the network are excited. This model has a mathematical structure which is significantly different from that of the connexionist model, the Hopfield model, or the Boltzman machine [3]. Thus the approximation capability of these networks also needs to

be examined in a manner distinct from that of previous models [4]. In particular, the "Gelenbe" random neural network model [5, 6, 9, 17] does not use sigmoid functions which are basic to the standard models' approximation capabilities.

The most basic requirement for a neural network model is that it should be a universal function approximator: i.e. to any continuous function f on a compact set, we should be able to find a specific network which implements a mapping close enough, in some precise sense to f, to a given degree of accuracy. Furthermore, among all networks which satisfy this property, we may wish to choose the one with the "smallest" size or the most simple structure.

In [16] we showed that the BGNN and the CGNN, two simple extensions of the basic "Gelenbe" Random Neural Network model, are universal approximators of continuous real-valued functions of s real variables. However we had not previously established the specific "size" constraints for the approximating networks.

In this paper we limit the networks to being feedforward and consider the case where the number of hidden layers does not exceed the number of input variables. With these constraints we show that the feedforward CGNN and the BGNN with s hidden layers (total of $s+2$ layers) can uniformly approximate continuous functions of s variables. We also extend a theorem in [16] on universal approximation using the CGNN with two output neurons, to the CGNN with only one output neuron.

The theoretical results we report in this paper are not only needed to justify the empirically observed success obtained in a variety of applications of the "Gelenbe" random neural network [7, 8, 10, 11, 12, 13, 14, 15], and to support further applied work in spiked stochastic neural network models. We believe that these results will lead to new developments in the design of network structures which are adapted to certain specific learning or approximation tasks.

REFERENCES

[1] W.E. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. I (3rd Edition) and Vol. II, Wiley, 1968, 1966.

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

[3] J.L. McClelland, D.E. Rumelhart, D.E., et al. *Parallel Distributed Processing*, Vols. I and II, MIT Press, 1986.

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

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

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

[7] 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.

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

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

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

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

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

[13] 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.

[14] 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.

[15] E. Gelenbe, A. Ghanwani, V. Srinivasan, "Improved neural heuristics for multicast routing," *IEEE J. Selected Areas in Communications*, vol. 15, no. 2, pp. 147-155, 1997.

[16] E. Gelenbe, Z. H. Mao, and Y. D. Li, "Function approximation with the random neural network," *IEEE Trans. Neural Networks*, vol. 10, no. 1, January 1999.

[17] E. Gelenbe, J.M. Fourneau "Random neural networks with multiple classes of signals," *Neural Computation*, vol. 11, pp. 721–731, 1999.

Appendix:
Proof of Technical Lemmas

Proof of Lemma 1: 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

$$P(u) = c_0 + c_1 u + ... + c_m u^m, \ 1/2 \leq u \leq 1, \tag{59}$$

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.**

Proof of Lemma 2: 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

$$q_{j+1,1} = (\frac{1}{1 + x})^j, \tag{60}$$

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

The next result exhibits a simple a construction process for algebraic expressions using the feedforward GNN.

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

$$q_{L,1} = g(x), \tag{61}$$

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

$$q_O = \frac{g(x)}{1 + g(x)}. \tag{62}$$

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. **Q.E.D.**

Proof of Lemma 3: 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

$$q_{j,1} = (\frac{1}{1+x})^j, \tag{63}$$

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:

$$q_O = \frac{\frac{P^*(x)}{2}}{\frac{1}{2C_{MAX}} + \frac{P^*(x)}{2}}. \tag{64}$$

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

$$q_O = \frac{P^+(x)}{1+P^+(x)}. \tag{65}$$

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

Proof of Lemma 4: The proof is very similar to that of Lemma 2. Construct a feedforward GNN with $v+1$ neurons numbered $(1,1), ...(v+1,1)$. Now set:

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

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

$$q_{j+1,1} = \frac{x}{(1+x)^j}, \tag{66}$$

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

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*

$$P^o(x) = c_0 + c_1 \frac{x}{1+x} + ... + c_m \frac{x}{(1+x)^m},\ 0 \leq x \leq 1, \tag{67}$$

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:

$$q_O = \frac{P^o(x)}{1 + P^o(x)}, \tag{68}$$

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