SCCS report: 774

Some Issues of Reducibility and Equivalence in Feedforward Neural Networks

M. A. Saleh Elmohamed and Geoffrey Fox
Northeast Parallel Architectures Center
Syracuse, NY 13244-4100 , USA .
{saleh, gcf}@npac.syr.edu

ABSTRACT

We show how sigmoidal (tanh()) feedforward net reducibility and equivalence conditions proposed by Sussmann and later by Sontag can be adapted to feedforward neural structures using piecewise rational activation functions. In addition, the uniqueness results demonstrated for the tanh() nets are shown to be applicable in part to networks of piecewise rationals, but not to piecewise linear (*clipping*) networks.

Contents

List of Figures

List of Tables

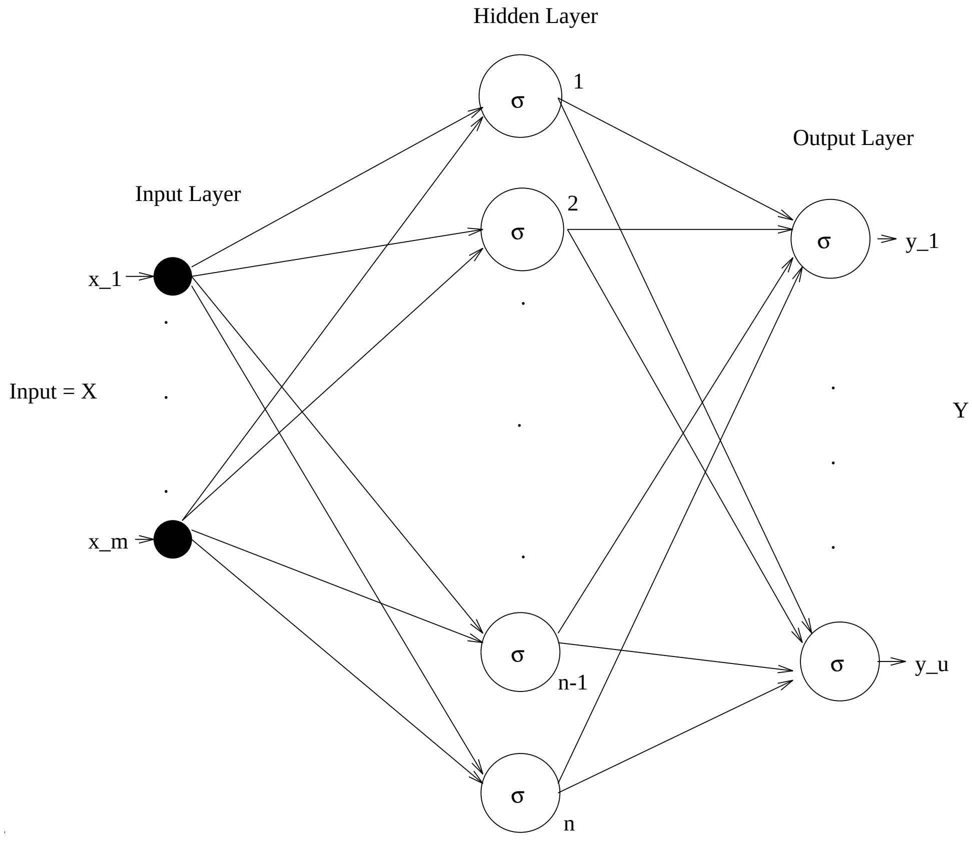

Figure 1: A feedforward neural net consisting of an input layer of m "fan-out" nodes, a single hidden layer of n computational nodes, and an output layer of u computational nodes. Each computational node carries a σ activation function and θ threshold.

1 Sketch of the background

Hector Sussmann [22] examined network reducibility and equivalence for the input-output mappings of a neural net. The net structure considered by [22] is a feedforward net with single hidden layer whose nodes use a hyperbolic tangent activation function (tanh). The units of the output layer also use tanh() activations.

In addition, Sussman proved what is called the *uniqueness theorem* in the framework of feedforward net structures. It is stated that *two irreducible feedforward nets with the same input-output mappings are related by a transformation in $\mathcal{G}$*, where $\mathcal{G}$ is a discrete symmetric group of nets as originally proposed by Hecht-Nielsen [10].

The main point in [22] is *to show that two input-output equivalent nets must necessarily be the same net, up to some simple internal symmetries* (more on this later). Also, it was pointed out that the irreducibility condition stated in the *uniqueness theorem* is needed because there is another source of non-uniqueness coming from the fact that a net may contain nodes that make no contribution whatsoever to the output. This means that there are cases when a net can be reduced without changing the I-O map, so the above theorem can hold only for irreducible nets. One major result of Sussmann's theorem is that the I-O mapping of an irreducible net can not be obtained from another net with fewer hidden nodes.

2 The main objectives

In this paper we address two issues/questions pertaining to the above work:

1. Do the *uniqueness* and *reducibility* results of Sussmann apply to nets with piecewise rational activation functions ? If so, discuss how this may come about, otherwise give an example.

2. Do the *uniqueness* and *reducibility* results of [22] apply to nets with piecewise linear activation functions? If so, outline this applicability, otherwise give an example.

3 Some Background on Feedforward Networks

3.1 Notation and Definitions

A feedforward network such as that of Figure-1 consists of an input layer whose nodes are merely fan-outs (no processing takes place in these nodes), one or more hidden layer(s) of computational nodes, and an output layer, also with one or more computational nodes. For the rest of the discussion, we will assume that both the hidden and output layer nodes use the same activation function.

It will be convenient to have a uniform notation for referring to the node input/output and weights in a feedforward network. Let $y_i^{[r-1]}$ the output of the i*th* node in the $(r-1)th$ layer of the net, $w_{ij}^{[r]}$ is the connection weight between i*th* node in the $(r-1)th$ layer and j*th* node of the r*th* layer (the current layer), and $\theta_j^{[r]}$ is the threshold of the j*th* node in the r*th* layer. Each computational node of layer r yields an output of $\sigma(\sum_i w_{ij}^{[r]} y_i^{[r-1]} + \theta_j^{[r]})$.

Definition 3.1 The function σ is any function satisfying the property that both $\lim_{u\to-\infty}\sigma(u)$ and $\lim_{u\to+\infty}\sigma(u)$ exist and are distinct. In this context, σ is often called a squashing function [11]. □

Typically, σ is one of the following:

1. For a $(-1,+1)$ output range, the activation function is:

$$\sigma_h(x) = \tanh(x) \quad . \tag{1}$$

2. For the output range of $(0,1)$, the activation function is:

$$\sigma_s(x) = \frac{1}{1+e^{-x}} \quad . \tag{2}$$

Observation 3.2 $\sigma_s(x)$ is equivalent to $\sigma_h(x)$, up to translations and change of coordinates. The relation between these two functions is:

$$\sigma_h(x) = 2\sigma_s(x) - 1 \ . \tag{3}$$

□

Definition 3.3 We say that σ is a *smooth activation function* if and only if

- σ is a squashing function,
- σ has a continuous derivative everywhere (hence, σ is continuous),
- σ is odd (hence, $\sigma(0) = 0$).

□

Observation 3.4 σ_h is a smooth activation function. □

3.2 Network Structure and the I-O Map

Using the notation introduced above and Figure-1, given an input vector $x = (x_1 \ldots x_m) \in \mathbb{R}^m$, the input to the jth node of the hidden layer is:

$$\upsilon_j(x) = \sum_{i=1}^{m} w_{ij}^{[2]} y_i^{[1]} \quad . \tag{4}$$

The output of the j^{th} hidden node is:

$$y_j^{[2]} = \sigma(\upsilon_j(x) + \theta_j^{[2]}) \quad . \tag{5}$$

Next, the input of the l^{th} node of the output layer is:

$$\mu_l(x) = \sum_{j=1}^{n} w_{ij}^{[3]} y_j^{[2]} \quad , \tag{6}$$

and the output of this l^{th} output node is:

$$y_l^{[3]} = \sigma(\mu_l(x) + \theta_l^{[3]}) \quad . \tag{7}$$

The I-O mapping of the network is the function from $\mathbb{R}^m \longrightarrow \mathbb{R}^u$ computed by the net.

4 Synopsis of Sussmann's Results

4.1 Definitions

Definition 4.1 $\mathcal{N}_{m,n,u}$ is the set of feedforward nets with activation functions σ where m denotes the size of the input layer, n is the size of the hidden layer, and u the size of the output layer. □

4.2 The Reducibility Conditions

Given a feedforward network , we associate a linear affine function with each node of the hidden layer as follows:

$$\gamma_j : \mathbb{R}^m \longrightarrow \mathbb{R}, \;\; \gamma_j(x) = \upsilon_j(x) + \theta_j, \quad \text{for} \;\; j = 1, \ldots, n. \tag{8}$$

Definition 4.2 Let α and β on $\mathbb{R}^m$ be linear affine functions; if either $\alpha(x) = -\beta(x)$ or $\alpha(x) = \beta(x)$, for all $x \in \mathbb{R}^m$, then α and β are sign-equivalent. □

Let b_{ij} be the weight between node i of the input layer and node j of the hidden layer, and c_{jk} be the weight between node j of the hidden layer and node k of the output layer. As outlined in [22], a feedforward network is reducible if one or more of the following conditions hold:

1. For some $j = 1, \ldots, n$ and all $k \in \{1, \ldots, u\}$, $c_{jk} = 0$.
2. There exist two different indices $j_1, j_2 \in \{1, \ldots, n\}$ such that the functions γ_{j_1} and γ_{j_2} are sign-equivalent (i.e. $\gamma_{j_1} = \pm\gamma_{j_2}$).
3. One of the functions γ_j , for $j \in \{1, \ldots, n\}$, is constant.

Next we briefly examine each of the above three cases.

- *(First case)*: Node j, of the hidden layer, and all its connections can be removed without any effect on the network output.
- *(Second case)*: Let $\rho \in \{-1, +1\}$ and $\gamma_{j_1} = \rho\gamma_{j_2}$. Since σ is an odd function, $\sigma(\rho s) = \rho\sigma(s)$. Remove node j_1 and all its connections, then change the output weights for j_2 to:
$$c_{j_2k}^{new} = \rho c_{j_1k}^{old} + c_{j_2k}^{old} \quad . \tag{9}$$
The above modification will not alter the original I-O mapping.
- *(Third case)*: In this case all the weights between the input layer and node j are zero, so node j is removed and thresholds of the output nodes are properly adjusted.

4.3 The Uniqueness Condition

The main point is to show that two irreducible, I-O equivalent nets must necessarily be the same net, up to some simple *internal symmetries*. Clearly, if a net N is reducible (using one or more of the above three reducibility conditions), then it is I-O equivalent to another net with fewer hidden nodes. Some simple *internal symmetries* create various transformations that can be applied to a network N without changing its I-O map. These symmetries are:

- **Sign-flipping**: Take a network N, pick a hidden node j and change the sign of all the fan-in connection weights b_{ij} for $i = 0, 1, \ldots, m$, to node j, change the sign of the fan-out connection weight c_{jk} for $k = 1, \ldots, u$, and also change the sign of its threshold (θ_j). Since σ is odd, this will not alter the contribution of this node j to the output layer. Now let us call the resulting net $\Upsilon_j(N)$.

- **Node permutation/interchange**: Using the same original net N, interchange two hidden nodes. That is, take two hidden nodes (in any order), j_1 and j_2, and relabel j_1 as j_2 and j_2 as j_1, taking care to also relabel the corresponding fan-in and fan-out weights and the two nodes thresholds. (Via this interchange, consider a new net with weights b_{ij}^{new}, c_{jk}^{new}, such that $b_{ij_1}^{new} = b_{ij_2}$, $b_{ij_2}^{new} = b_{ij_1}$, $c_{j_1k}^{new} = c_{j_2k}$, $c_{j_2k}^{new} = c_{j_1k}$, and thresholds $\theta_{j_1}^{new}$, $\theta_{j_2}^{new}$, such that $\theta_{j_1}^{new} = \theta_{j_2}$, and $\theta_{j_2}^{new} = \theta_{j_1}$). Let us call the resulting net $\Phi_{j_1j_2}(N)$.

The maps Υ_j, $\Phi_{j_1j_2}$ generate a finite group $\mathcal{G}_{m,n,u}$ of transformations of the set $\mathcal{N}_{m,n,u}$ of all nets with m input nodes, n hidden nodes, and u output units. We can call two nets in $\mathcal{N}_{m,n,u}$ *equivalent* if they are related by a transformation in the group $\mathcal{G}_{m,n,u}$. It is then clear that two equivalent nets are I-O equivalent.

Now, Sussmann's major result is:

Theorem 1 *Let N_1, N_2 be irreducible I-O equivalent nets in $\mathcal{N}_{m,n_1,u}$, $\mathcal{N}_{m,n_2,u}$, with σ_h as activation functions. Then (i) $n_1 = n_2$ and (ii) N_1 and N_2 are $\mathcal{N}_{m,n,u}$-equivalent.*

This theorem implies the following corollary:

Corollary 4.3 An irreducible σ_h-net is minimal. □

5 Piecewise Rational Functions, Uniqueness, and Reducibility

The class of piecewise rational functions is quite large, and for the rest of the paper we will confine ourselves to two types of functions in this class. Definitions of these two piecewise rational functions will be carried out in this section. Also, from this section on we will refer to the set of piecewise rational functions which are also smooth activations (see definition 3.3) as $\mathcal{PWR}$, and to networks using them as $\mathcal{PWR}$ nets.

5.1 Why are we interested in piecewise rational functions?

Here are few reasons:

1. When properly set over intervals such as $(-1, 1)$, piecewise rational functions can approximate the function $tanh()$ over the same interval.

2. Evaluation of $\mathcal{PWR}$ functions may be much faster than evaluation of $tanh()$.

3. If training and convergence are not significantly worse for $\mathcal{PWR}$ nets then their overall training will be faster.
4. For sequential implementations, the time to evaluate σ (the activation function) and its first derivative, σ', may not be very important in large networks; but parallel implementations can be more sensitive to the effort required to evaluate σ and σ'. Thus, we expect a faster parallel implementation when $\mathcal{PWR}$ functions are used as activation functions.

5.2 Definitions and Observations

Definition 5.1 For an integer $k > 0$, and real value $x_0 = \frac{k}{k+1}$, we define:

$$pwr_k(x) = 1.0 - \frac{(1.0 - x_0)x_0^k}{x^k} \ , \quad \text{for } x \neq 0 \ . \tag{10}$$

The first $\mathcal{PWR}$ activation function used is:

$$\sigma_k(x) = \begin{cases} -pwr_k(-x) & \text{if } x < -x_0 \\ pwr_k(x) & \text{if } x > x_0 \\ x & \text{otherwise.} \end{cases} \tag{11}$$

□

Observation 5.2 The first derivative of the function in (10) is:

$$pwr_k'(x) = k\frac{(1.0 - x_0)x_0^k}{x^{k+1}} \ , \tag{12}$$

and the second derivative is:

$$pwr_k''(x) = -k(k+1)\frac{(1.0 - x_0)x_0^k}{x^{k+2}} \ . \tag{13}$$

Now the first derivative of $\sigma_k(x)$ is:

$$\sigma_k'(x) = \begin{cases} pwr_k'(|x|) & \text{if } x < -x_0 \text{ or } x_0 < x \\ 1 & \text{otherwise.} \end{cases} \tag{14}$$

□

Observation 5.3 From previous definitions and observations, we observe the following:

- $pwr_k(x_0) = x_0$, and
- $pwr_k'(x_0) = k\frac{(1.0 - x_0)x_0^k}{x_0^{k+1}} = k\frac{(1.0-x_0)}{x_0} = k\frac{1.0 - \frac{k}{(k+1)}(k+1)}{k} = 1$.

Also we observe some properties of σ_k that are shared with tanh:

- For all x, $-1 \leq \sigma_k(x) \leq 1$.
- σ_k is odd.
- σ_k and $\sigma_k^{'}$ are continuous.
- For all x, $\sigma_k^{'} > 0$.
- $\sigma_k(0) = 0$ and $\sigma_k^{'}(0) = 1$.

□

The second $\mathcal{PWR}$ function, due to Elliott [8], is defined next.

Definition 5.4

$$\sigma_p(x) = \frac{x}{1 + |x|} \ . \tag{15}$$

□

Observation 5.5 The function stated in equation (15) is differentiable everywhere to:

$$\sigma_p^{'}(x) = \frac{1}{(1 + |x|)^2} \ \ , \tag{16}$$

and satisfies a "general" logistic differential equation as follows:

$$\sigma_p^{'} = (1.0 - |\sigma_p|)^2 \ . \tag{17}$$

□

Observation 5.6 The steps leading to equation (17) from function (15) are as follows:

$$\text{Let} \quad p_1(x) = 1 - \frac{1}{1 + x} = \frac{x}{1 + x} \quad \text{for } x \geq 0 \ , \tag{18}$$

differentiating to:

$$p_1{'}(x) = \frac{(1 + x) - x}{(1 + x)^2} = \frac{1}{(1 + x)^2} = \frac{1}{(1 + |x|)^2} \ , \tag{19}$$

and a second derivative of:

$$p_1{''}(x) = \frac{-2}{(1 + x)^3} \ . \tag{20}$$

Also,

$$\text{let} \quad p_2(x) = -1 + \frac{1}{1 - x} = \frac{x}{1 - x} \quad \text{for } x < 0 \ , \tag{21}$$

differentiating to:

$$p_2{'}(x) = \frac{(1 - x) + x}{(1 - x)^2} = \frac{1}{(1 - x)^2} = \frac{1}{(1 + |x|)^2} \ , \tag{22}$$

and a second derivative of:

$$p_2''(x) = \frac{2}{(1-x)^3} \ . \tag{23}$$

Now, from equations (19, 22) we derive the following:

$$\frac{1}{(1+|x|)^2} = \frac{(|x|-|x|+1)^2}{(1+|x|)^2} = (\frac{|x|+1}{|x|+1} - \frac{|x|}{1+|x|})^2 = (1 - \frac{|x|}{1+|x|})^2 = (1-|\sigma_p(x)|)^2 \ . \tag{24}$$

□

Remark 5.7 It is worth pointing out that σ_p and σ_h satisfy an "autonomous" differential equation, so that calculating σ_p' or σ_k' requires very little effort beyond the calculation of the function itself. The aforementioned "autonomy" concept can be illustrated by the following example:

To evaluate functions $\sigma_h' = 1 - \sigma_h^2$ and $\sigma_p' = (1 - |\sigma_p|)^2$, the only values needed are σ_h and σ_p, respectively.

Regarding the activation function σ_k, it seems not to satisfy such a differential equation, but even so calculation of σ_k' requires negligible additional effort. Also, networks using these three types of activation functions, can be trained using backpropogation algorithm. □

Observation 5.8 We observe some properties of σ_p shared with σ_h:

- σ_p and σ_p' are continuous.
- For all x, $\sigma_p'(x) > 0$.
- $\sigma_p(0) = 0$ and $\sigma_p'(0) = 1$.
- σ_p is odd.
- For all x, $-1 \leq \sigma_p(x) \leq 1$.

□

5.2.1 On the continuity of the piecewise rational functions

As in observation (5.3), when $x = x_0$, we obtain $pwr_k'(x_0) = 1$. From this and the $pwr_k'(x)$ of equation (12), we saw that the first derivative of function $pwr_k(x)$ is continuous.

Regarding the second derivative $pwr_k''(x)$, at $x = x_0$, we obtain the following derivation:

$$pwr_k''(x_0) = -k(k+1)\frac{(1.0-x_0)}{x_0^2} = -k(k+1)[\frac{(k+1)^2}{k^2} - \frac{(k+1)}{k}] = -\frac{(k+1)^3}{k} + (k+1)^2 \ , \tag{25}$$

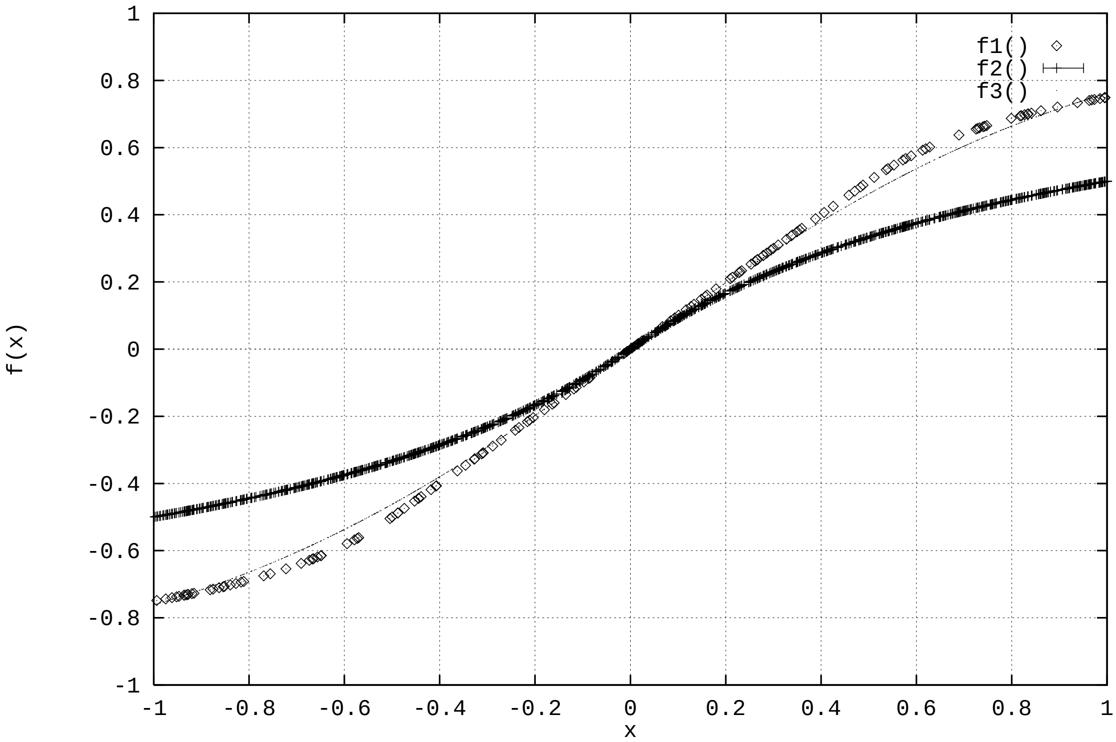

Figure 2: f1() $= \sigma_1$, f2() $= \sigma_p$, and f3() $=$ tanh().

and

$$-\frac{(k+1)^3}{k} + (k+1)^2 = -\frac{(k+1)^2}{k} \neq 0 \ , \tag{26}$$

which is not continuous. For a pictorial representation, see Figure-(3).

On the issue of first and second derivatives continuity of the function defined in equation (15), the first derivative of equation (19) is clearly continuous as well as that of equation (22).

About the second derivatives of equations (20 and 23) we get the following:

$$\text{At } x = 0 \ , \quad {p_1}''(0) = -2 \neq {p_2}''(0) = 2 \ . \tag{27}$$

Therefore, as shown in Figure-(4), the function's graph is broken at 0. Hence, σ_p'' is not a continuous function.

5.3 Network Equivalence and Uniqueness

Let N_1 and N_2 stand for 2 feedforward nets, for example, see Figure-1. Also, let the symbol ($\sim$) stands for $\mathcal{N}_{m,n,u}$-equivalence in behavior, and ($\equiv$) for $\mathcal{N}_{m,n,u}$-equivalence in structure.

Albertini, Sontag and Maillot [1] have stated that given two networks N_1 and N_2, one says that they are I-O equivalent if the behavior of N_1 equal to the behavior of N_2. In

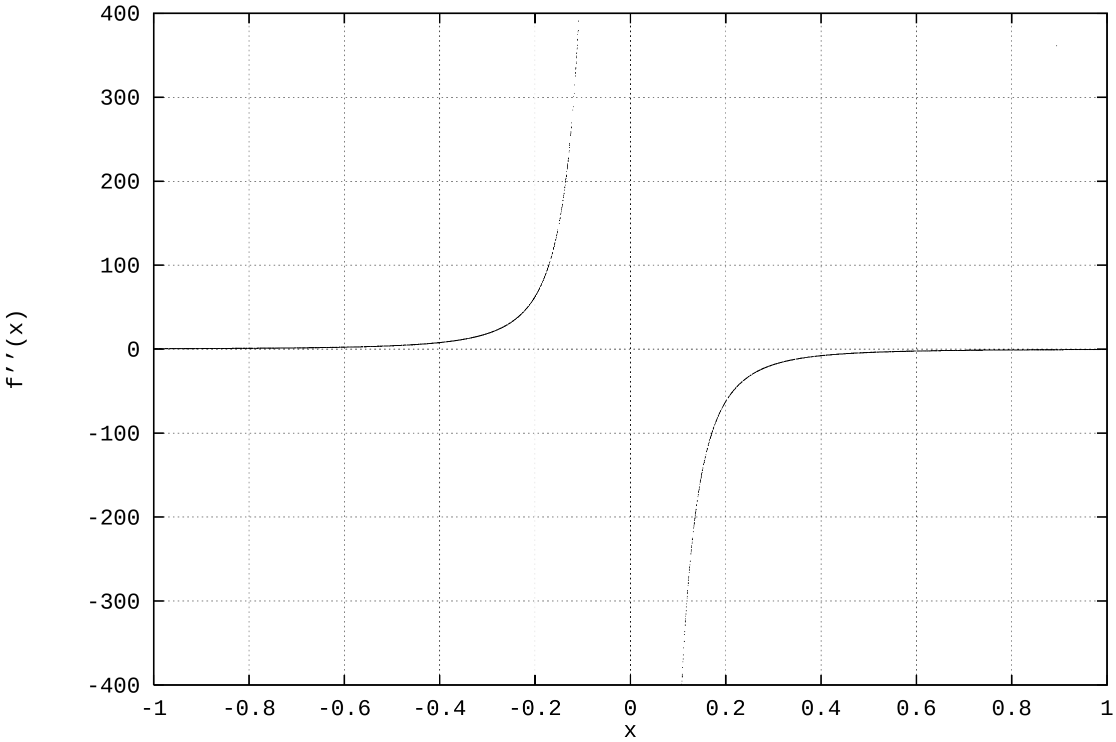

Figure 3: A plot showing the 2^{nd} derivative (f'') of pwr_k at $k = 1$, which is pwr_k'' .

other words, applying the same set of inputs to both networks will yield the same output. Clearly, networks with identical structures and identical internal representations (*weight sets*) have the same I-O mappings. Also, nets with different internal representations can have the same I-O mappings.

Now, what we would like to answer is the following question: When does $N_1 \sim N_2$ imply $N_1 \equiv N_2$?

Sontag [20] pointed out that Sussmann's argument takes into account the "independence property" (IP) (defined below), but he did not explicitly state this property in his work [22]. Also, they state that to show uniqueness of the connection weights of a given network, one needs to demonstrate that the given activation functions satisfy IP. Sussmann showed that σ_h and the related sigmoid σ_s satisfy IP.

5.3.1 On the Independence Property

Definition 5.9 A function σ satisfies IP *if only if, for every positive integer l, every set of non-zero real numbers $A = \{a_1, \ldots, a_l\}$, and every set of real numbers $B = \{b_1, \ldots, b_l\}$ which satisfy*

$$\forall (i,j) \text{ such that } i \neq j, \text{ neither } (a_i, b_i) = (a_j, b_j) \text{ nor } (a_i, b_i) = (-a_j, -b_j) \ , \qquad (28)$$

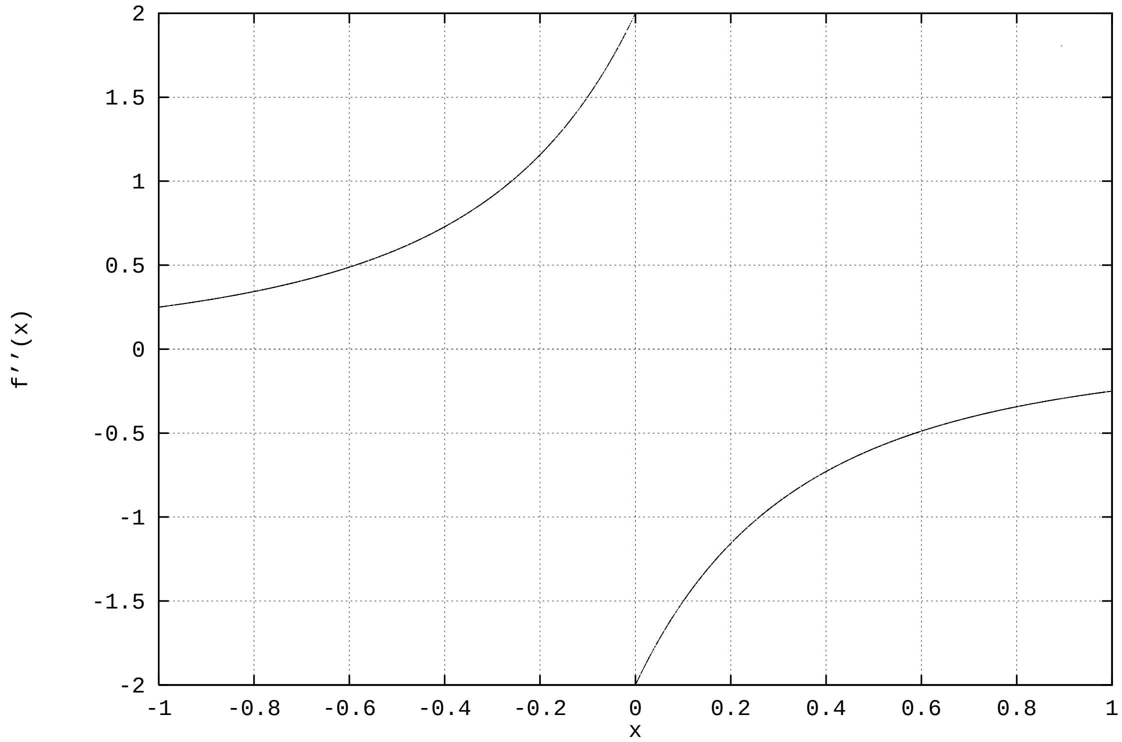

Figure 4: A plot showing the 2^{nd} derivative (f'') of σ_p, which is σ_p'' .

the functions:

$$\mu_0(u) = 1, \mu_1(u) = \sigma(a_1 u + b_1), \ldots, \mu_l(u) = \sigma(a_l u + b_l) \tag{29}$$

are linearly independent. This linear independence means that unless

$$c_0 = c_1 = c_2 = \ldots = c_l = 0 \ , \tag{30}$$

it must be that, for some u :

$$c_0 + c_1\sigma(a_1 u + b_1) + \ldots + c_l\sigma(a_l u + b_l) \neq 0 \ . \tag{31}$$

Elements of equation (29) are of the linear space (over $\mathcal{R}$), consisting of all functions defined on $\mathcal{R}$:

$$(c_0\mu_0 + c_1\mu_1 + \ldots + c_l\mu_l)(x) = c_0\mu_0(x) + \ldots + c_l\mu_l(x) \ . \tag{32}$$

Otherwise, they are linearly dependent and one of them can be formed as a linear combination of the others. □

In addition, σ satisfies the **Weak Independence Property** (WIP), if the above IP is only required to hold for all pairs with $b_i = 0, i = 1, \ldots, l$.

There are many nonlinear maps that do not satisfy IP. An example of functions that belongs to this category are **polynomials**, **periodic functions**, and the **exponential function** [2].

5.3.2 Uniqueness and the Independence Property

Observation 5.10 Albertini, Sontag and Maillot [2] observe that in fact Sussmann showed the following:

If N_1 and N_2 are irreducible nets with activation σ and (i) $N_1 \sim N_2$, (ii) σ is odd, and (iii) σ satisfies IP, then $n_1 = n_2$ and $N_1 \equiv N_2$ (see Theorem 1). □

5.3.3 The Independence Property and Piecewise Rational Functions

Assume our network uses $\mathcal{PWR}$ functions (see definitions 5.1 and 5.4) as the activations of the nodes. Now, we need to address the issue of whether these types of activation functions satisfy the IP.

First: Assume the activation function is σ_k.

Choose an integer $l > 0$, non-zero real numbers $a_1, \ldots, a_l$ and real numbers $b_1, \ldots, b_l$.

$$\sigma_k(a_i x + b_i) = a_i x + b_i \quad \text{if} \ \ |a_i x + b_i| \leq x_0 \ , \tag{33}$$

otherwise

$$\sigma_k(a_i x + b_i) = sgn(a_i x + b_i)(1 - \frac{(1-x_0)x_0^k}{(|a_i x + b_i|)^k}) \ . \tag{34}$$

Select a set of coefficients $C = \{c_0, \ldots, c_l\}$. We need to show:

$$c_0 + \sum_{i=1}^{l} c_i \sigma_k(a_i x + b_i) \neq 0 \quad \text{for some} \ \ x \ . \tag{35}$$

By confining one's attention to x for which $|x|$ is sufficiently large that $|a_i x + b_i| > x_0$, for all i, one can see that it is sufficient to show that pwr_k satisfies the independence property. A sketch of the proof follows:

<u>Proof</u>

We show by *contradiction* that pwr_k satisfies the IP. Assume any reference to σ is a reference to σ_k.

[1]: σ is C^∞ on $|x| < x_0$, and $|x| > x_0$.

Let $\sigma^{(n)}$ to be the n^{th} derivative of σ. Also note that:

$$\sigma_-^{(n)}(x_0) = \lim_{x \to x_0^-} \sigma^{(n)}(x_0) \tag{36}$$

$$\sigma_+^{(n)}(x_0) = \lim_{x \to x_0^+} \sigma^{(n)}(x_0) \tag{37}$$

Let $\Delta_n = \sigma_+^{(n)}(x_0) - \sigma_-^{(n)}(x_0) \quad$ for $n \in \mathcal{N}^*$.

By straightforward calculations, for $n \geq 2$, we get:

$$\sigma_-^{(n)}(x_0) = 0 \tag{38}$$

$$\sigma_+^{(n)} = (-1)^{n+1} \frac{(k+n-1)!}{(k-1)!} \cdot \frac{(1-x_0)}{{x_0}^n} \tag{39}$$

Therefore,

$$\Delta_n = (-1)^{n+1} \frac{(k+n-1)!}{(k-1)!} \cdot \frac{(1-x_0)}{{x_0}^n} \neq 0 \tag{40}$$

(The main point here is that $\Delta_n \neq 0$ for $n \geq 2$, of course $\Delta_0 = 0$ and $\Delta_1 = 0$ because of the continuity of σ and $\sigma^{'}$).

[2]: Now suppose there exist:

$(c_0, c_1, \ldots, c_l \in \mathcal{R}^*)$ and $(b_1, b_2, \ldots, b_l \in \mathcal{R}^*)$ and $(B_1, B_2, \ldots, B_l \in \mathcal{R})$

such that:

$$(b_i, B_i) \neq \pm(b_j, B_j) \quad \forall i \neq j \tag{41}$$

and

$$c_0 + \sum_{i=1}^{l} c_i \sigma(b_i x + B_i) = 0 \quad (\forall x \in \mathcal{R}) \tag{42}$$

Changing the sign of some of the c_i, we also can assume all the $b_i > 0$.

(This assumption is not actually necessary but more convenient for the proof).

Now, let $x_1 \in \mathcal{R}$ be the *largest* $x \in \mathcal{R}$ such that for one of the b_i, we have:

$x \longmapsto \sigma(b_i x + B_i)$ is not C^∞ at x_1. This means that $b_i x_1 + B_i = x_0$ because $b_i > 0$.

Consider all the b_j such that: $x \longmapsto \sigma(b_j x + B_j)$ is not C^∞ at x_1.

Rearranging the b_j, we can denote them by $b_1, \ldots, b_p$ for $p \leq l$. Consider the function:

$$\Theta(x) + \sum_{i=p+1}^{l} c_i \sigma(b_i x + B_i) = -c_0 \tag{43}$$

We get that Θ is C^∞ at x_1. Of course, we must have $p > 1$. Now, for x near x_1, but with $x \neq x_1$, we get:

$$\Theta^{(n)}(x) = \sum_{i=1}^{p} c_i b_i^n \sigma^{(n)}(b_i x + B_i) \tag{44}$$

and because Θ is C^∞ at x_1,

$$\lim_{x \to x_1^-} \Theta^{(n)}(x) = \lim_{x \to x_1^+} \Theta^{(n)}(x) \tag{45}$$

So, because:

$$\lim_{x \to x_1^-} \sigma^{(n)}(b_i x + B_i) = \sigma_-^{(n)}(x_0) \tag{46}$$

$$\lim_{x \to x_1^+} \sigma^{(n)}(b_i x + B_i) = \sigma_+^{(n)}(x_0) \tag{47}$$

(Remember that $b_i x_1 + B_i = x_0$). Now, we get:

$$\left(\sum_{i=1}^{p} c_i b_i^n \right) \left[\sigma_+^{(n)}(x_0) - \sigma_-^{(n)}(x_0) \right] = 0 \tag{48}$$

$\Longrightarrow \left(\sum_{i=1}^{p} c_i b_i^n \right) \Delta_n = 0$.
Then, we choose $n \geq 2$, and $\Delta_n \neq 0$, $\quad \Longrightarrow \sum_{i=1}^{p} c_i b_i^n = 0 \quad \forall n \geq 2$.
Considering this quality for $2 \leq n \leq (p+2)$, we get:

$$\begin{pmatrix} b_1^2 & \dots & b_p^2 \\ \dots & \dots & \dots \\ b_1^{p+2} & \dots & b_p^{p+2} \end{pmatrix} \begin{pmatrix} c_1 \\ \dots \\ c_p \end{pmatrix} = 0 \tag{49}$$

Now, the $c_i \neq 0$, we can conclude that:

$$\det \begin{vmatrix} b_1^2 & \dots & b_p^2 \\ \dots & \dots & \dots \\ b_1^{p+2} & \dots & b_p^{p+2} \end{vmatrix} = 0 \tag{50}$$

$\Longrightarrow (b_1 - b_p)^2 \prod_{1 \leq i < j \leq p} (b_i - b_j) = 0$, because the $b_i > 0$, we obtain:

$$\prod_{1 \leq i < j \leq p} (b_i - b_j) = 0 \tag{51}$$

This means that we can find i and j such that: $i \neq j$ and $b_i = b_j$. But,

$$b_i x_1 + B_i = x_0 = b_j x_1 + B_j \tag{52}$$

Then $B_i = B_j$, hence, we get: $i \neq j$ and $(b_i, B_i) = (b_j, B_j)$.
Which is a clear *contradiction*. **QED**.

Certainly the above model (and reasoning) can be applied in more general situations. For example, whenever we consider piecewise rational functions, (because such a type of functions cannot be C^∞ on $\mathcal{R}$. Now, the theorem becomes:

Theorem 2 *Let f be an odd or even piecewise rational function, then f has (or satisfies) the strong I.P. property.* ∎

Anyway, the odd or even hypothesis is kind of strong, and taking care of what one needs to do, it is possible to prove the same theorem for "almost" piecewise rational functions.

Second: Let the activation function be σ_p. Sontag [20] indicated in a private communication that σ_p satisfies the independence property.

Clearly, networks using σ_p, and assuming the above conjecture, networks using σ_k, satisfy the property that if N_1 and N_2 are irreducible in $\mathcal{N}_{m,n,u}$ and the $N_1 \sim N_2$ then $N_1 \equiv N_2$.

6 Piecewise Linear Nets

6.1 Why are we interested in them?

1. Piecewise linear functions ($\mathcal{PLF}$) are simple to evaluate.

2. Smooth activation functions, such as σ of the previous section, were originally introduced as smooth versions of certain piecewise linear activation functions (in this case, the clipping type).

3. Clipping plays a quite useful role as outlined recently by Hertz et. al [11] (see the section on discretization and clipping).

4. It is stated in Sontag's work[18] that the conventional squashing function (see previous section) as well as the $\mathcal{PLF}$ satisfy the criteria of continuity and differentiability almost everywhere (see conditions S_1 and S_2 of definition 3.2 in [18]).

6.2 Definitions and Observations

Definition 6.1 (see [16]) A function φ on the interval $[a, b]$ is a piecewise linear ($\mathcal{PLF}$) if there is a subdivision $a = x_0 < x_1 < \ldots < x_n = b$ such that φ is linear on each interval $[x_i, x_{i+1}]$, for $0 \leq i < n$. □

In this paper, we only consider the class of $\mathcal{PLF}$ that meets the continuity criteria. This class will be referred to as the *clipping* type or the *clip* functions.

Definition 6.2 The *clip* function (*clip*) is of one of the following two forms:

$$clip_0(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } 0 < x < 1 \\ 1 & \text{if } x \geq 1 \end{cases} \tag{53}$$

$$clip_1(x) = \begin{cases} -1 & \text{if } x \leq -1 \\ x & \text{if } -1 < x < 1 \\ 1 & \text{if } x \geq 1 \end{cases} \tag{54}$$

□

Observation 6.3 The relationship between $clip_0$ and $clip_1$ is:

$$clip_1(x) = 2 \cdot clip_0(\frac{x+1}{2}) - 1 \tag{55}$$

□

From this point on, any reference to *clip* is intended to refer to $clip_1$.

Definition 6.4 A piecewise linear network ($\mathcal{PLN}$) is a feedforward multilayer net using $\mathcal{PLF}$ as activation functions. □

6.3 Examples

The following example shows a network in which the concepts of irreducibility and uniqueness do not hold. Again, the hidden and output layers of Figure-5(a) are composed of *clip* functions. Clearly n has a value greater than 1, hence $clip(n) = 1$.

Now, the net of Figure-5(a) has the following output:

$$y_1 = clip(\frac{1}{n}clip(n \cdot x) + \frac{1}{n}clip(n \cdot x) + \ldots + \frac{1}{n}clip(n \cdot x) + clip(x)), \tag{56}$$

$$= clip(\frac{n-1}{n}clip(n \cdot x) + clip(x)), \tag{57}$$

$$= clip(\frac{n-1}{n}clip(n) \cdot clip(x) + clip(x)), \tag{58}$$

$$= clip(\frac{n-1}{n}clip(x) + clip(x)), \tag{59}$$

$$= clip((1 + \frac{n-1}{n})clip(x)), \tag{60}$$

$$= \frac{clip(2n-1)}{clip(n)}clip(clip(x)). \tag{61}$$

Since $n > 1$ we get $(2n-1) > 1$, thus $clip(2n-1) = 1$.

$$y_1 = clip(clip(x)), \tag{62}$$

$$= clip(x) . \tag{63}$$

The net of Figure-5(b) gives an output of:

$$y_2 = clip(n \cdot clip(1 \cdot x)) = clip(n) \cdot clip(clip(x)) = clip(x) . \tag{64}$$

While the output of Figure-5(c) net is:

$$y_3 = clip(1 \cdot clip(n \cdot x)) = clip(clip(n \cdot x)) = clip(clip(n) \cdot clip(x)), \tag{65}$$

$$= clip(clip(x)) = clip(x) . \tag{66}$$

Since $y_1 = y_2 = y_3$, the three networks are I-O equivalent. Clearly, the nets are not structurely equivalent, which contradicts the concepts of irreducibility and uniqueness discussed in the last section. Hence, piecewise linear nets using *clip* activation functions, do not satisfy the properties of irreducibility and uniqueness.

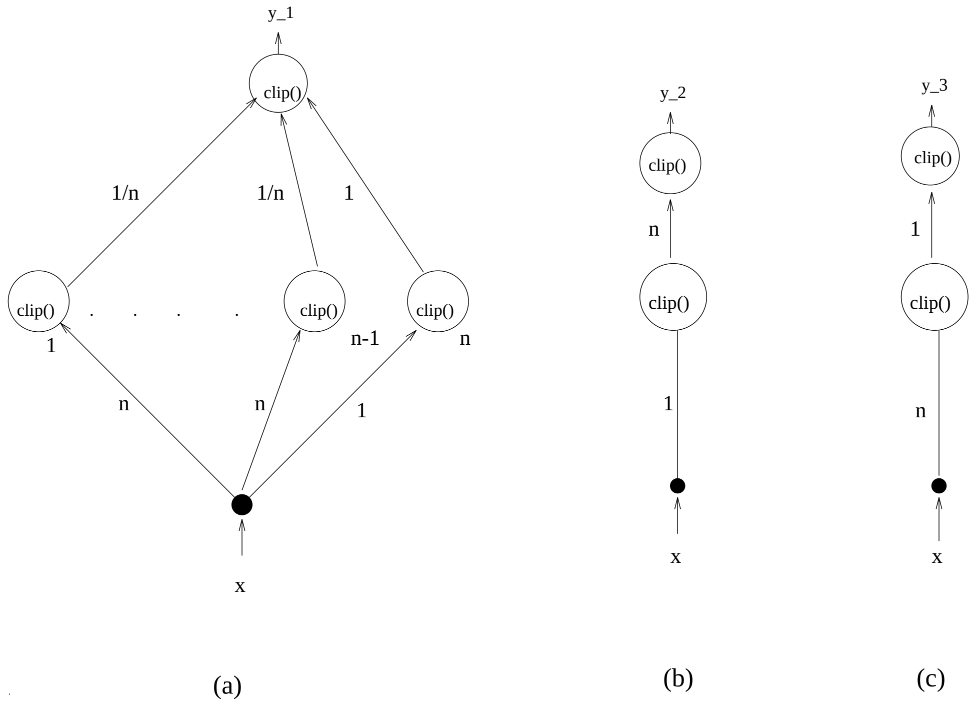

Figure 5: Three piecewise linear nets.

7 Experimental Results for Piecewise Rational Nets

Using the Aspirin/MIGRAINES [13] software system, a neural net example is used to show the evaluation time of σ_k and σ_p. First, we substituted σ_k for σ_h (the net's default activation function), then ran it on the problem and recorded the results. Next, σ_p is used in place of σ_k and the results were also recorded. These two sets of results are compared with those obtained from running the network on the same problem using σ_h.

7.1 The Spiral Problem

Briefly, this is an example of a very hard pattern recognition problem. The data are the x and y coordinates of two spirals. In the actual data file the coordinates for each spiral have an additional number associated with them which denotes to which spiral the coordinate belongs to (i.e., 1.0 for one spiral and -1.0 for the other). The two spirals coil three times around the origin and around one another. The goal of this example is to train the network to map an (x, y) coordinate into the proper spiral. The network is trained by giving it x and y and the target classification for some of the points along these intertwined spirals. For a detailed description of this problem, see Lang and Withbrock [12].

7.2 The Experiment

7.2.1 Problem Description

Two sets of points are generated, each with 97 members (three complete revolutions at 32 points per revolution, plus endpoints). Each point is represented by two floating-point numbers, the x and y coordinates, plus an output value of -1.0 for one set, $+1.0$ for the other set.

7.2.2 Methodology

The task is to train on the 194 (97 x 2) I/O pairs until the learning system can produce the correct output for all of the inputs. The time required for learning is then recorded. For back-propagation systems, the time is typically recorded in "epochs". Each epoch includes one training presentation for each of the 194 points in the training set. An output is considered to be a logical -1 if it is in the lower 40% of the output range, a +1 if it is in the upper 40%, and indeterminate (and therefore incorrect) if it is in the middle 20% of the range.

7.2.3 Results

Lang and Withbrock [12] report results obtained on this problem using $2 \rightarrow 5 \rightarrow 5 \rightarrow 5 \rightarrow 1$ (a two-node input layer, five-node first hidden layer, five-node second hidden layer, five-node third hidden layer, and one-node output layer) network with short-cut connections (each unit is connected to every unit in all earlier layers, not just to units in the previous layer). For this experiment, we use a $2 \rightarrow 5 \rightarrow 10 \rightarrow 15 \rightarrow 1$ network with also short-cut connections.

In the three runs of the network over the same input data, the first run uses σ_h, the second uses σ_1, and the third uses σ_p activation functions. For each run, the network weights are initialized with a uniformly distributed random values in the range -0.1 to $+0.1$. Learning rate and momentum are set at 0.1 and 0.95, respectively. The error bound is set to 0.4. The network is periodically (every 100000 iterations) saved to a file. Table-(1) summarizes the various testing and learning times. Note that the elapsed compute and system times are measured in *seconds*, and average performance on the task is measured in *connections per second.*

8 Conclusion

We have looked at the issues of reducibility and uniqueness, pertaining to feedforward neural nets composed of tanh() units. In the course of this analysis, two types of piecewise rational functions and one type of piecewise linear activation have been examined to determine whether they also satisfy reducibility and uniqueness. Our results show that neither reducibility nor uniqueness holds when the functions involved are piecewise linear (the *clip* type functions), but they do hold when piecewise rational activations are used.

	Using σ_h		Using σ_1		Using σ_p	
	Learning	Testing	Learning	Testing	Learning	Testing
Elapsed Compute Time	916.75	1.2400	579.70	1.7000	1967.0	1.4000
Elapsed System Time	1.4800	0.2100	0.3500	0.2200	0.9400	0.1800
Avg. Performance on the task	86828.5	62267.7	171827	72160.7	171988	74242.3
Total iterations	200000	194	250000	194	850000	194
Total epochs	1030.93	1.000	1288.66	1.000	4381.44	1.000
Total number of connection	398	398	398	398	398	398

Table 1: Performance Figures for Spirals

Acknowledgment

Many thanks to Professor Ernie Sibert, for his useful comments and suggestions while writing this report.

References

[1] Albertini, F., E. Sontag, and V. Maillot, "For Neural Nets, Function Determines Form", Rutgers Center for Systems and Control, report SYSCON-92-03. 1992.

[2] Albertini, F., E. Sontag, and V. Maillot, "Uniqueness of weights for neural networks", in *Artificial Neural Networks with Applications in Speech and Vision*, edited by R. Mammone, Chapman and Hall, London, 1993.

[3] Chester, Dan, *DIMACS Neural Nets Workshop presentation*, May 1991.

[4] Chester, Dan, "The generalization capabilities of feedforward neural nets", Unpublished Draft.

[5] Chester, Dan, private communication.

[6] Cybenko, George, "Continuous valued neural networks with two hidden layers are sufficient," CSRD-935, Univ. of Illinois, Urbana,IL (March 1988).

[7] Cybenko, George, "Complexity Theory of Neural Networks and Classification Problems," Univ. of Illinois, Urbana, IL (April 1990).

[8] Elliot, David L. "A Better Activation Function for Artificial Neural Networks", TR 93-8, Institute for Systems Research, University of Maryland, College Park, MD 20742, January 1993.

[9] Elliot, David, private communication.

[10] Hecht-Nielsen, Robert "Theory of Backprop neural nets",in proc. of Int. Joint Conf. on Neural Networks, Washington, 1989, IEEE Publications, NY, pp. 593-605.

[11] Hertz, John, Andres Krogh, and Richard Palmer, *Intro. to the theory of neural computation*, Addison-Wesley, Redwood City, 1991.

[12] Lang, Kevin, and Michael Withbrock, "Learning To Tell Two Spirals Apart", in the proceedings of the 1988 Connectionists Summer School, Morgan Kaufmann (San Mateo, 1988).

[13] Leighton, Russell, The Aspirin/MIGRAINES Neural Network Software system. Release V6.0, the Mitre Corp. 1992.

[14] Maillot, V, private communication.

[15] Minsky, M and S. Papert, *Perceptron* , MIT Press, Cambridge, MA (1988 edition)

[16] Royden, H. L., *Real Analysis*, 3rd edition, The Macmillan Company, New York, 1988.

[17] Rumelhart, D , J McClelland, and the PDP Research Group. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol I, 1986. MIT Press, Cambridge.

[18] Sontag, E, “Feedforward nets for interpolation and classification,” SYCON Report, Rutgers Univ., 1991.

[19] Sontag, E, “Feedback stabilization using two-hidden-layer nets,” IEEE Trans. Neural Nets 3 (1992): 981-990.

[20] Sontag, E, private communication.

[21] Sussmann, Sontag, and Kuhn, (editors) Proceedings of the DIMACS workshop on theoretical neural nets. May 1991.

[22] Sussmann, H, “Uniqueness of the weights for minimal feedforward nets with a given a given input-output map,” Report SYCON-91-06a, Rutgers Center for Systems and Control, Rutgers University, May 1991.