Proceedings of the 31st Conference
on Decision and Control
Tucson, Arizona • December 1992

Some Recent Results on Computing With "Neural Nets"

Hava T. Siegelmann*
Department of Computer Science, Rutgers University, New Brunswick, NJ 08903
Eduardo D. Sontag*
Department of Mathematics, Rutgers University, New Brunswick, NJ 08903

Abstract

Neural networks, understood here as systems consisting of linearly interconnected dynamical elements (integrators in continuous-time, or delay lines in discrete-time) and scalar nonlinearities, provide an appealing model of analog computation. We describe some recent results on: (1) approximation properties and (2) computational capabilities, of such dynamical systems.

1. Introduction

"Neural networks" provide an appealing model of analog computation. These are discrete- or continuous-time systems built by linearly combining dynamic elements —delay lines or integrators respectively— with memory-free scalar elements, each of which performs the same nonlinear transformation $\sigma : \mathbb{R} \to \mathbb{R}$ on its input. In this paper we concentrate on "recurrent" nets.

The function σ could be $\text{sign}(x) = x/|x|$ (zero for $x = 0$). Often one wants a differentiable saturation, and for this, especially in the neural network field, it is customary to consider the hyperbolic tangent $\tanh(x)$, which is close to the sign function when the "gain" γ is large in $\tanh(\gamma x)$. Also common in practice is a piecewise linear function, $\pi(x) := x$ if $|x| < 1$ and $\pi(x) = \text{sign}(x)$ otherwise; this is sometimes called a "semilinear" or "saturated linearity" function. See Figure 1.

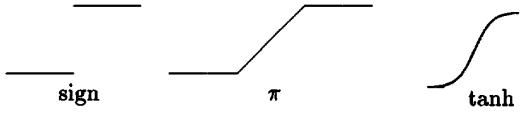

Figure 1: Different Functions σ

The type of system that we consider in this paper is given by the model

$$\boxed{\Delta x = \sigma_n(Ax + Bu), \quad y = Cx} \qquad (1)$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $C \in \mathbb{R}^{p \times n}$, for some integers n (the dimension of the system), m (number of inputs) and p (number of outputs). Here and later, we use the following notational convention. For each $\sigma : \mathbb{R} \to \mathbb{R}$, we let $\sigma_r : \mathbb{R}^r \to \mathbb{R}^r$ be the map that lets σ act coordinatewise, that is,

$$\sigma_r((x_1, \ldots, x_r)') = (\sigma(x_1), \ldots, \sigma(x_r))',$$

and we drop the subscript r when clear from the context. Similarly, the symbol Δ indicates respectively time-shift $(\Delta x)(t) = x^+(t) = x(t+1)$ or time-derivative $(\Delta x)(t) = \dot{x}(t) = \frac{d}{dt}x(t)$ depending on the context (discrete- or continuous-time), and we denote by Δx the application of Δ to each coordinate of the vector x. We call these *σ-systems* or *recurrent neural nets*. See Figure 2.

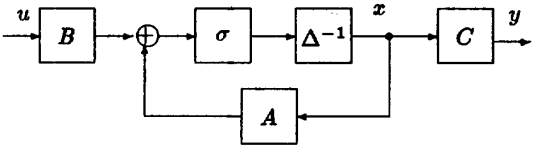

Figure 2: Recurrent Network

In the continuous-time case, we assume that σ is globally Lipschitz, so that solutions are defined for all time, for any measurable essentially bounded control. (This is a reasonable assumption for all applications, and in any case it could be relaxed as long as one keeps track of domains of definition.)

The study of recurrent networks has many different motivations. They constitute a very powerful model of computation, as we review below, and are capable of approximating —in a restricted sense— rather arbitrary behaviors. Such systems have been proposed as models of large scale parallel computation, since they are built of potentially many simple processors or "neurons". Electrical circuit implementations of recurrent networks, employing resistively connected networks of n identical nonlinear amplifiers, with the resistor characteristics used to reflect the desired weights, have been proposed as models of analog computers, in particular in the context of constraint satisfaction problems and in content-addressable memory applications. Recent papers have explored the computational and dynamical properties of several different variations of recurrent networks; see for instance [5], [3] and references there.

Some authors have been motivated by a loose analogy to neural systems —hence the terminology— and for this reason the entries of the matrices A, B, and C are sometimes called "weights" or "synaptic strengths" and σ —taken to be of a sigmoidal type— is called the "activation function" (which represents how each neuron x_i responds to its aggregate stimulus). In this context, the outputs $y(t)$ can be thought of as measurements recorded by probes that average the activation values of many neurons.

*This research supported in part by grant AFOSR-91-0343

In speech processing applications and language induction, recurrent net models are used as identification models, and they are fit to experimental data by means of a gradient descent optimization (the so-called "back-propagation" technique) of some cost criterion.

In this paper, we will mention some recent results on the computational power of σ-systems, and we review their approximation capabilities. A number of recent papers explore related issues and provide further results. In the paper [18], other structures somewhat less general than recurrent networks are also studied, and connections are made to various system theoretic questions of observability and controllability. The paper [17] includes more details on the use of these systems as well as other neural-net based models for identification and control, in particular in the role of discontinuous feedback (when a discontinuous activation is used) for stabilization of nonlinear systems. The paper [1], also in these Proceedings, discusses the problem of parameter identifiability of recurrent networks.

2. Approximation

In a restricted sense, with recurrent networks one may approximate a wide class of nonlinear plants. Approximations are *only* valid on compact subsets of the state space and for finite time, so interesting dynamical characteristics are not reflected. However, there are many instances in control and signal processing where recurrent networks may play a role analogous to that of bilinear systems, which had been proposed previously (see e.g. [19]) as universal models. Before showing how a universal approximation property arises, we recall some standard facts from neural network theory.

2.1. Approximation, Interpolation, Classification

Given a function $\sigma : \mathbb{R} \to \mathbb{R}$, let $\mathcal{F}_\sigma$ be the affine span of the set of all the maps $\sigma_{a,b}(x) := \sigma(ax+b)$, with $a, b \in \mathbb{R}$. That is, the elements of $\mathcal{F}_\sigma$ are those functions $\mathbb{R} \to \mathbb{R}$ that are finite linear combinations $c_0 + \sum_i c_i \sigma(a_i x + b_i)$. We say that the mapping σ is a *universal nonlinearity* if for each $-\infty < \alpha < \beta < \infty$ the restrictions to the interval $[\alpha, \beta]$ of the functions in $\mathcal{F}_\sigma$ constitute a dense subset of $C^0[\alpha, \beta]$, the set of continuous functions on $[\alpha, \beta]$ endowed with the metric of uniform convergence.

Not every nonlinear function is universal in the above sense, of course; for instance, if σ is a polynomial of degree k then $\mathcal{F}_\sigma$ is the set of all polynomials of degree $\leq k$, hence closed and not dense in any C^0. But most nonlinear functions are universal. Indeed, Hornik proved in [6] that any σ which is continuous, nonconstant, and bounded is universal (see also [4] for related results). (M. Leshno has recently shown that universality holds for any continuous function which is not a polynomial.) In the rather general case of "sigmoidal" functions, that is, nondecreasing functions with the property that both $\lim_{x \to -\infty} \sigma(x)$ and $\lim_{x \to +\infty} \sigma(x)$ exist (without loss of generality, assume the limits are -1 and $+1$ respectively), universality is not hard to prove, as follows. Take any continuous function f on $[\alpha, \beta]$. One can first approximate f uniformly by a piecewise constant function, i.e. by an element of $\mathcal{F}_{\mathbf{sign}}$; then each sign function is approximated by $\sigma(\gamma x)$, for large enough positive γ.

We recall that to say that a function F is *computable by a one-hidden-layer net* means that F factors as

$$F(u) = F_1(\sigma_r(F_2(u))) \,, \tag{2}$$

where $F_1 : \mathbb{R}^r \to \mathbb{R}^p$ and $F_2 : \mathbb{R}^m \to \mathbb{R}^r$ are affine maps. That is to say, each coordinate of F is in $\mathcal{F}_\sigma$. It is a standard fact that universality of σ implies that, for each m, p and each compact subset K of $\mathbb{R}^m$, the set of functions $F : \mathbb{R}^m \to \mathbb{R}^p$ computable by single hidden layer nets is dense in $C^0(K)$.

2.2. Approximations of Nonlinear Systems

We now explain how the above translates into the fact that recurrent nets provide universal identification models, in a suitable sense. Consider a continuous- or discrete-time, time-invariant, control system Σ:

$$\begin{aligned} \dot{x} \ [\text{ or } x^+ \ ] &= f(x,u) \\ y &= h(x) \end{aligned} \tag{3}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, and $y(t) \in \mathbb{R}^p$ for all t, and f and h are continuously differentiable. For any measurable essentially bounded control $u(\cdot) : [0,T] \to \mathbb{R}^m$, we denote by $\phi(t, x_0, u)$ the solution at time t of (3) with initial state $x(0) = x_0$; this is defined at least on a small enough interval $[0, \varepsilon)$, $\varepsilon > 0$. For recurrent networks, when σ is bounded or globally Lipschitz with respect to x, it holds that $\varepsilon = T$; we will assume here that we are dealing with controls for which solutions exist globally, at least for the states on some compact set of interest. For each control, we let $\lambda(u) = \lambda_{\Sigma, x_0}(u)$ be the output function corresponding to the initial state $x(0) = x_0$, that is, $\lambda(u)(t) := h(\phi(t, x_0, u))$. We wish to see that, on compacts, and for finite time intervals, this system can be approximately simulated by a recurrent network. We first define what is meant by approximate simulation.

Assume given two systems Σ and $\widetilde{\Sigma}$, as in (3), where we use tildes to denote data associated to the second system, and with same number of inputs and outputs (but possibly $\tilde{n} \neq n$). Suppose also that we are given compact subsets $K_1 \subseteq \mathbb{R}^n$ and $K_2 \subseteq \mathbb{R}^m$, as well as an $\varepsilon > 0$ and a $T > 0$. Supposed further (this simplifies definitions, but can be relaxed) that for each initial state $x_0 \in K_1$ and each control $u(\cdot) : [0,T] \to K_2$ the solution $\phi(t, x_0, u)$ is defined for all $t \in [0,T]$. We'll say that the system $\widetilde{\Sigma}$ *simulates* Σ *on the sets* K_1, K_2 *in time* T *and up to accuracy* ε if there exist two continuous mappings $\alpha : \mathbb{R}^n \to \mathbb{R}^{\tilde{n}}$ and $\beta : \mathbb{R}^{\tilde{n}} \to \mathbb{R}^n$ so that the following property holds: For each $x_0 \in K_1$ and each $u(\cdot) : [0,T] \to K_2$, denote $x(t) := \phi(t, x_0, u)$ and $\tilde{x}(t) :=$

$\widetilde{\phi}(t, \beta(x_0), u)$; then this second function is defined for all $t \in [0, T]$, and

$$\|x(t) - \alpha(\widetilde{x}(t))\| < \varepsilon\,, \quad \|h(x(t)) - \widetilde{h}(\widetilde{x}(t))\| < \varepsilon$$

for all such t.

One may ask for more regularity properties of the maps α and β as part of the definition; in any case the maps to be constructed below can be taken to be at least differentiable.

Assume that σ is a universal nonlinearity in the sense defined earlier. Then, for each system Σ and for each K_1, K_2, ε, T as above, there is a recurrent network $\widetilde{\Sigma}$ that simulates Σ on the sets K_1, K_2 in time T and up to accuracy ε. Some variations of this result were given earlier and independently in [10] and [8], under more restrictive assumptions and with somewhat different definitions. As we haven't found it in this manner in the literature, we next sketch a proof.

The final maps α and β will be built up out of several elementary maps. The first step is to add, if necessary, the equation

$$\dot{y} = \frac{\partial h(x)}{\partial x} . f(x, u)$$

in the continuous case, or

$$y^+ = h(f(x, u))$$

in discrete-time, so that from now on one may take without loss of generality h linear, that is, $y = Cx$ for some matrix C. The enlarged system simulates the original one via $\alpha(x, y) = x$ and $\beta(x) = (x, h(x))$.

Next one finds a function as in equation (2) that uniformly approximates $f(x, u)$ (for the extended system) close enough on $K_1 \times K_2$; this will imply the desired approximation of solutions, by any standard well-posedness result, as discussed e.g. in [16], Theorem 37. This new function is specified by matrices and vectors

$$T_1 \in \mathbb{R}^{n \times r}, A \in \mathbb{R}^{r \times n}, B \in \mathbb{R}^{r \times m}, \alpha, \beta \in \mathbb{R}^n$$

and

$$f(x, u) \approx T_1 \sigma_r(Ax + \alpha + Bu) + \beta.$$

Finally, one needs to show that a system with such a right-hand side and output $y = Cx$ can be itself simulated by some recurrent network. Changing coordinates in $\mathbb{R}^n$ if necessary, one may assume that T_1 has the form $(T'0)'$, where T is of full row rank. Thus the equations take the form

$$\begin{aligned} \dot{x}_1 \text{ [or } x_1^+ \text{]} &= T\sigma_r(A_1 x_1 + A_2 x_2 + \alpha + Bu) + \beta_1 \\ \dot{x}_2 \text{ [or } x_2^+ \text{]} &= \beta_2 \end{aligned}$$

and the output function is $y = C_1 x_1 + C_2 x_2$ in these coordinates. Write $n_2 = n - \operatorname{rank} T$ for the size of the x_2 variable. This is essentially a recurrent network after a change of variables $x = Tz$ and elimination of the constant ("bias") vectors α, β_1, β_2. More precisely, we proceed as follows.

First, there is a $\widetilde{\beta}_1$ so that $T\widetilde{\beta}_1 = \beta_1$ (since T has full row rank). Consider the system of dimension $r + n_2$ consisting of the above equation for x_2 together with:

$$\dot{z}_1 \text{ [or } z_1^+ \text{]} = \sigma_r(A_1 T z_1 + A_2 x_2 + \alpha + Bu) + \widetilde{\beta}_1$$

and output $y = C_1 T z_1 + C_2 x_2$. Given any initial condition $(\xi_1, \xi_2)' \in \mathbb{R}^n$, and any control $u(\cdot)$, pick the solution of the (z_1, x_2) system that has $z_1(0) = \zeta$ and $x_2(0) = \xi_2$, where ζ is any vector so that $T\zeta = \xi_1$ (again use that T is onto). Write $x_1(t) := Tz_1(t)$ along this solution. Then $(x_1(t), x_2(t))'$ satisfies the original equations, and has the initial value $(\xi_1, \xi_2)'$, so it is the state trajectory corresponding to the given control. In conclusion, each trajectory of the original system can be simulated by some trajectory of the z_1, x_2-system. Let $\sigma_{n_2}(0) = \gamma$; then the equation for x_2 can be written with right-hand side $\sigma(0x + 0u) + (\beta_2 - \gamma)$; thus, redefining n as $r + n_2$, one is reduced to studying systems of the following special form:

$$\dot{x} \text{ [or } x^+ \text{]} = \sigma_n(Ax + Bu + \alpha) + \beta,$$

with linear output $y = Cx$. One is only left to eliminate the bias terms α and β. Consider first treat the continuous-time case. Pick any real numbers μ, ν so that

$$\mu\sigma(\nu) = 1$$

and consider these equations in dimension $n + 2$: $\dot{z} = \sigma_n(Az + \mu z_{n+1} A\beta + \mu z_{n+2}\alpha + Bu)$, $\dot{z}_{n+1} = \sigma(\nu \mu z_{n+2})$, $\dot{z}_{n+2} = 0$, with output $y = C(z + \mu z_{n+1}\beta)$, where $z(t) \in \mathbb{R}^n$. Given any initial $\xi \in \mathbb{R}^n$ and any control $u(\cdot)$, pick the solution of this extended system for which $z(0) = \xi$, $z_{n+1}(0) = 0$, and $z_{n+2}(0) = 1/\mu$. Consider

$$x(t) := z(t) + \mu z_{n+1}(t)\beta.$$

Observe that $z_{n+2} \equiv 1/\mu$ and $\dot{z}_{n+1} \equiv \sigma(\nu\mu\frac{1}{\mu}) = 1/\mu$. Therefore $\dot{x}(t) = \sigma_n(Ax + Bu + \alpha) + \beta$, and $x(0) = z(0) + \mu z_{n+1}(0) = \xi$, so the z, z_{n+1}, z_{n+2} system provides the desired simulation. In discrete time, the only modification needed consists of replacing the z_2 equation by $z_2^+ = \sigma(\nu\mu z_{n+2})$. This completes the sketch of the proof of the approximation result.

Thus, recurrent nets approximate a wide class of nonlinear plants. Note, however, that approximations are only valid on compact subsets of the state space and for finite time, so that many interesting dynamical characteristics are not reflected. This is analogous to the role of bilinear systems, which had been proposed previously (work by Fliess and Sussmann in the mid-1970s) as universal models. As with bilinear systems, it is obvious that if one imposes extra stability assumptions ("fading memory" type) it will be possible to obtain global approximations, but this is probably not very useful, as stability is often a goal of control rather than an assumption.

For applications of these types of approximations to signal processing see [8], and [10] for applications to control and identification.

3. Computing Power

In the work [11], [12], and [13] we dealt with the computational capabilities of recurrent networks, seen from the point of view of classical formal language theory. There we studied discrete-time systems with $\sigma = \pi$. Though more general nonlinearities as well as continuous-time systems are of interest, note that using π = sign would give no more computational power than finite automata. Our main results —after suitable definitions, see below— are: (1) with rational matrices A, B, and C, recurrent networks are computationally equivalent, up to polynomial time, to Turing machines; (2) with real matrices, all possible binary functions, recursive or not, are "computable" (in exponential time), but when imposing polynomial-time constraints, an interesting class results.

Restricting for simplicity to language recognition, the (extremely elegant) situation can be summarized as follows, where "(usual) P" stands for the standard class of polynomial-time computable problems, and "circuit P" is discussed further later.

Weights	Capability	Polytime
integer	regular	regular
rational	recursive	(usual) P
real	arbitrary	circuit P

Computational universality, both in the rational and real cases, is due to the unbounded precision of state variables, in analogy to the potentially infinite tape of a Turing machine. In the cellular-automata literature, other Turing and super-Turing models have been proposed, but they involve an unbounded number of state variables (processor units), as opposed to a finite number fixed in advance as in our work.

An immediate consequence of the universality results is the fact that the problem of determining if a recurrent network ever reaches an equilibrium, from a given initial state, is effectively undecidable; note that this is precisely the question relevant when using such systems for content-addressable retrieval, where the initial state is the "input pattern" and the final state is interpreted as the retrieved output or classification. This consequence follows from an immediate reduction to the halting problem of Turing machines.

We now state precisely the simulation results (the forms are slightly different, but equivalent to, those in the papers cited above, which should be consulted for details). We deal with recurrent networks with $\sigma = \pi$, the piecewise-linear saturation, and having just one input and output channel ($m = p = 1$). A pair consisting of a recurrent network Σ and an initial state $\xi \in \mathbb{R}^n$ is *admissible* if the following property holds: Given any input of the special form

$$u(\cdot) = \alpha_1, \ldots, \alpha_k, 0, 0, \ldots, \qquad (4)$$

where each $\alpha_i = \pm 1$ and $1 \leq k < \infty$, the output that results with $x(0) = \xi$ is either $y \equiv 0$ or is a sequence of the form

$$y(\cdot) = \underbrace{0, 0, \ldots, 0}_{s}, \beta_1, \ldots, \beta_l, 0, 0, \ldots, \qquad (5)$$

where each $\beta_i = \pm 1$ and $1 \leq l < \infty$. The pair (Σ, ξ) will be called *rational* if the matrices defining Σ as well as the initial ξ all have rational entries; in that case, for rational inputs all ensuing states and outputs remain rational.

Each admissible pair (Σ, ξ) defines a partial function

$$\phi : \{-1, 1\}^+ \to \{-1, 1\}^+,$$

where $\{-1, 1\}^+$ is the free semigroup in the two symbols ± 1, via the following interpretation: Given a sequence $w = \alpha_1, \ldots, \alpha_k$, consider the input in Equation (4), and the output, which is either identically zero or has the form in Equation (5). If $y \equiv 0$, then we say that $\phi(w)$ is undefined; otherwise, if Equation (5) holds, then $\phi(w)$ is defined as the sequence $\beta_1, \ldots, \beta_l$, and we say that the response to the input sequence w was computed *in time* $s + l$. One says that the (partial) function ϕ is *realized* by (Σ, ξ).

In order to be fully compatible with standard recursive function theory, we are allowing the possibility that a decision is never made, corresponding to a partially defined behavior. On the other hand, if for each input sequence w there is a well-defined $\phi(w)$, and if there is a function on positive integers $T : \mathbb{N} \to \mathbb{N}$ so that the response to each sequence w is computed in time at most $T(|w|)$, where $|\alpha_1, \ldots, \alpha_k| = k$, we say that (Σ, ξ) *computes in time* T.

In the special case when ϕ is everywhere defined and $\phi : \{-1, 1\}^+ \to \{-1, 1\}$, that is, the length of the output is always one, one can think of ϕ as the characteristic function of a subset L of $\{-1, 1\}^+$, that is, a *language* over the alphabet $\{-1, 1\}$.

Given $T : \mathbb{N} \to \mathbb{N}$, we say that the language L is computed in time T if the corresponding characteristic function is, for some admissible pair that computes in time T. For a function $T : \mathbb{N} \to \mathbb{N}$, we let $\mathrm{NET}(T)$ be the class of languages computed by admissible pairs in time T.

3.3. Arbitrary Time

Disregarding computation time, some of the main results from [12] and [13] can be summarized as follows:

Theorem. *Let $\phi : \{-1, 1\}^+ \to \{-1, 1\}^+$ be any partial function. Then ϕ can be realized by some admissible pair. Furthermore, ϕ can be realized by some rational admissible pair if and only if ϕ is a partial recursive function.* ∎

For the rational case, one shows how to simulate an arbitrary Turing machine. In fact, the proof shows how to do so in linear time, and tracing the construction results in a simulation of a universal Turing machine by a recurrent network of dimension roughly 1000. The

main idea of the proof in the real case relies in storing all information about ϕ in one weight, by a suitable encoding of an infinite binary tree. Then, π-operations are employed, simulating a chaotic mapping, to search this tree. In both the real and rational cases, the critical part of the construction is to be able to write everything up in terms of π, and the use of a Cantor set representation for storage of activation values. Cantor sets permit making binary decisions with finite precision, taking advantage of the fact that no values may appear in the "middle" range.

3.4. Polynomial Time

It is of course much more interesting to impose resource constraints, in particular in terms of computation time. We restrict to language recognition, for simplicity of exposition, but similar results can be given for computation of more general functions.

In order to present our results, we need to briefly recall some of the basic definitions of non-uniform families of circuits (see e.g. [2]), a concept which appears often in current theoretical computer science. A *Boolean circuit* is a directed acyclic graph. Its nodes of in-degree 0 are called *input nodes*, while the rest are called *gates* and are labeled by one of the Boolean functions AND, OR, or NOT (the first two seen as functions of many variables, the last one as a unary function). One of the nodes, which has no outgoing edges, is designated as the *output* node. The *size* of the circuit is the total number of gates. Adding if necessary extra gates, we assume that nodes are arranged into levels $0, 1, \ldots, d$, where the input nodes are at level zero, the output node is at level d, and each node only has incoming edges from the previous level. The *depth* of the circuit is d, and its *width* is the maximum size of each level. Each gate computes the corresponding Boolean function of the values from the previous level, and the value obtained is considered as an input to be used by the successive level; in this fashion each circuit computes a Boolean function of the inputs.

A *family of circuits* $\mathbf{C}$ is a set of circuits

$$\{c_n, n \in \mathbb{N}\} \ .$$

These have sizes $S_{\mathbf{C}}(n)$, depth $D_{\mathbf{C}}(n)$, and width $W_{\mathbf{C}}(n)$, $n = 1, 2, \ldots$, which are assumed to be monotone nondecreasing functions. If $L \subseteq \{0,1\}^+$, we say that the language L is *computed by the family* $\mathbf{C}$ if the characteristic function of

$$L \bigcap \{0,1\}^n$$

is computed by c_n, for each $n \in \mathbb{N}$. One says, then, that L can be computed by a "circuit of size $S_{\mathbf{C}}$." (Though standard, this is not a good terminology, as $S_{\mathbf{C}}$ is really a function, and "circuit" now means a family of circuits.) Given $S : \mathbb{N} \to \mathbb{N}$, we say that the language L is computed by circuits in size S if there is a circuit of size S that computes L. For such a function S, we let $\textsc{circuit}(S)$ be the class of languages computed by (families of) circuits of size S.

The concepts of circuit size and neural network computation time can be related as follows; see [13] for a proof.

Theorem. *For each function $F(n) \geq n$:*

- $\textsc{circuit}(F(n)) \subseteq \textsc{net}(nF^2(n))$.
- $\textsc{net}(F(n)) \subseteq \textsc{circuit}(F^3(n))$. ∎

What matters most from this implication is that the two measures of complexity are polynomially related. Thus, *languages computed by polynomial size circuits are the same as those computed by networks in polynomial time*. This allows, in turn, the application of standard results from nonuniform circuit theory, in particular those results dealing with polynomial size, to networks. The class of languages computed by polynomial size circuits is often called "P/poly" and coincides with the class of languages recognized by Turing machines "with advice sequences" in polynomial time. It also coincides with the class of languages recognized in polynomial time by Turing machines which consult oracles, where the oracles are sparse sets. Sparse sets are those for which, for each length n, the number of words which are not longer than n is bounded by a polynomial. Furthermore, one knows that, in a precise sense, all languages can be recognized in exponential circuit size, and hence exponential time using networks, and that for most languages such exponential time is in fact necessary (use the result for circuits given in [2], theorem 5.11, pg 122).

3.5. Nondeterministic Neural Networks

One of the more interesting areas in computer science deals with the concept of nondeterministic computation time. Essentially, one is interested in comparing the time it takes to verify that a proposed solution to a problem is indeed a solution with the time it would take to actually find a solution. For many problems, the first (verification time) is easy to estimate, but the latter is not.

In [13] we define a *nondeterministic network* to be an admissible pair having an extra binary input line, the *Guess* line, in addition to the input line already assumed. A language L is said to be accepted by a nondeterministic admissible pair in time T if for each word w of length n which belongs to L there is a "guess" input w' of length polynomial in n so that the input (w, w') gives an accepting output.

The concept of a nondeterministic circuit family is defined in the literature, also by means of an extra input.

Our results apply in the nondeterministic case as well. That is, the class of languages accepted by a nondeterministic network in polynomial time, and the class of languages accepted by a nondeterministic non-uniform family of circuits of polynomial size, turn out to be the same.

Since the standard class NP of nondeterministic polynomial time languages is included in that of languages accepted by a nondeterministic networks in polynomial time, (one may simulate a nondeterministic Turing machine by a nondeterministic network with rational weights,) the equality "P=NP" for neural networks would imply that NP $\subseteq$ P/poly. Thus, from [7] we conclude that this equality is impossible, unless a widely believed conjecture would be false (the polynomial hierarchy collapses to Σ_2).

In summary, even though networks, as analog devices, can "compute" far more than digital computers, they still give rise to a rich theory of computation, in the same manner as the latter.

4. Remarks

One of the most exciting challenges in current control theory and signal processing is that of formulating a rich mathematical framework in which to study the interface between the continuous (analog) world and discrete (digital) computers which are capable of symbolic processing. Succesful approaches will eventually allow the interplay of modern control with automata theory and other techniques from computer science. This is needed because, although classical control techniques have proved spectacularly successful in automatically regulating relatively simple systems, in practice controllers resulting from the application of the well-developed theory are often used as building blocks of far more complex systems. The integration of these systems is often accomplished by means of ad-hoc techniques that combine pattern recognition devices, various types of switching controllers, and humans –or, more recently, expert systems,– in supervisory capabilities. The need to understand the analog/digital interface has motivated much research into areas such as discrete-event systems, supervisory control, and more generally "intelligent control systems". One would hope that the study of dynamical systems as analog computing devices may be a useful component of the general approaches that will eventually emerge.

References

[1] Albertini, F., and E.D. Sontag, "For neural networks, function determines form," *Proc. IEEE Conf. Decision and Control, Tucson, Dec. 1992*, IEEE Publications, 1992.

[2] BalCazar, J.L., J. Diaz, and J. Gabarro, *Structural Complexity*, Springer-Verlag, Berlin, 1988.

[3] Cohen, M.A., and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. Systems, Man, and Cybernetics* **13**(1983): 815–826.

[4] Cybenko, G., "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals, and Systems* **2**(1989): 303-314.

[5] Hopfield, J.J., "Neurons with graded responses have collective computational properties like those of two-state neurons," *Proc. of the Natl. Acad. of Sciences, USA* **81**(1984): 3088–3092.

[6] Hornik, K., "Approximation capabilities of multilayer feedforward networks," *Neural Networks* **4**(1991): 251-257.

[7] Karp, R.M., and R. Lipton, "Turing Machines that take advice," *Enseign. Math.* **28**(1982): 191-209.

[8] Matthews, M., "On the uniform approximation of nonlinear discrete-time fading-memory systems using neural network models," Ph.D. Thesis, E.T.H. Zurich, Diss. ETH No. 9635, 1992.

[9] McCulloch, W.S., and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.* **5**(1943): 115-133.

[10] Polycarpou, M.M., and P.A. Ioannou, "Identification and control of nonlinear systems using neural network models: Design and stability analysis," Report 91-09-01, Sept. 1991, Dept. of EE/Systems, USC, Los Angeles.

[11] Siegelmann, S., and E.D. Sontag, "Turing computability with neural nets," *Appl. Math. Lett.* **4(6)**(1991): 77-80.

[12] Siegelmann, H.T., and E.D. Sontag, "On the computational power of neural nets," in *Proc. Fifth ACM Workshop on Computational Learning Theory*, Pittsburgh, July 1992, pp. 440-449.

[13] Siegelmann, H.T., and E.D. Sontag, "Analog computation, neural networks, and circuits," submitted.

[14] Sontag, E.D., "Feedforward nets for interpolation and classification," *J. Comp. Syst. Sci.*, **45**(1992): 20-48.

[15] Sontag, E.D., "Feedback Stabilization Using Two-Hidden-Layer Nets," in *Proc. Amer. Automatic Control Conference*, Boston, June 1991, pp. 815-820.

[16] Sontag, E.D., *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, Springer, New York, 1990.

[17] Sontag, E.D., "Neural nets as systems models and controllers," in *Proc. Seventh Yale Workshop on Adaptive and Learning Systems*, pp. 73-79, Yale University, 1992.

[18] Sontag, E.D., "Systems combining linearity and saturations, and relations to "neural nets," in *Proc. Nonlinear Control Systems Design Symp., Bordeaux, June 1992* (M. Fliess, Ed.), IFAC Publications, pp. 242-247.

[19] Sussmann, H.J., "Semigroup representations, bilinear approximations of input-output maps, and generalized inputs," in *Mathematical Systems Theory, Udine 1975* (G. Marchesini, Ed.,) Springer-Verlag, New York, pp. 172-192.