Repr. from *Proc. Seventh Yale Workshop on Adaptive and Learning Systems*, pp. 73-79, May 1992.

NEURAL NETS AS SYSTEMS MODELS AND CONTROLLERS*

Eduardo D. Sontag
Dept. of Mathematics, SYCON - Rutgers Center for Systems and Control
Rutgers University, New Brunswick, NJ 08903
E-mail: sontag@hilbert.rutgers.edu

Abstract

This paper briefly surveys some recent results relevant to the suitability of "neural nets" as models for dynamical systems as well as controllers for nonlinear plants. In particular, it touches upon questions of approximation, identifiability, construction of feedback laws, classification and interpolation, and computational capabilities of nets. No discussion is included of "learning" algorithms, concentrating instead on representational issues.

1. Introduction

The basic paradigm for control is that of a "plant" or physical device P interconnected with a controller C. The controller uses measurements from P in order

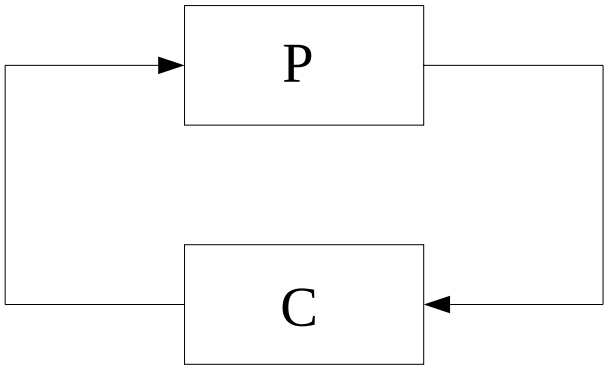

Figure 1: *Basic Paradigm*

to compute signals, which are then fed back into the plant so as to attain a given regulation objective. (This description can be extended to incorporate the effect of external disturbances, the specification of desired trajectories, and so forth.)

The plant P represents an existing system, and it is essential to have a mathematical model of P which is accurate and useful both for theoretical analysis and controller synthesis. Sometimes, a good model can be derived from physical principles involving equations of motion, fluid dynamics, or chemical balance considerations. Often, however, only behavioral —also called *input/output*— data is available. In that case, one typically postulates a general parametric model for P, its values to be fit using a numerical procedure. "Neural nets" have been proposed as one such source of parameterizations.

A related but different issue to that of deciding upon an appropriate model for P is that of choosing the type of controller C to be employed. While in the classical —linear— case it is reasonable to use a linear C, the choice is far less clear in general. Neural nets have been suggested, too, in this dual mode as controllers.

What is meant precisely by the term neural net varies, depending on the author. The general consensus is that this means a simple, often linear, interconnection of static nonlinear scalar processors, supplemented sometimes by memory elements (integrators, or delay lines). The coefficients of the interconnections, called "weights", play a role vaguely analogous to the concentrations of neurotransmitters in biological neuronal systems, while the nonlinear processors in this over-simplified analogy correspond to the neurons themselves. In engineering practice, one decides a priori on the class of "neurons" to be used, that is, the type of nonlinearity allowed —typically of the "sigmoidal" type reviewed below— and the weights are the parameters that are numerically adjusted in order to either model the plant, in the identification application, or to obtain a controller.

Motivating the use of nets is the belief —as of yet not fully theoretically formalized— that in some sense such nets are an especially appropriate family of parameterized models, from a numerical and practical point of view.

In this presentation we briefly survey some theoretical results, especially those obtained by the author and his coworkers, dealing with the use of nets for identification and control. As a general rule, we ignore numerical questions and algorithms, concentrating instead on the ultimate capabilities of nets for the above purposes. The discussion is mainly intuitive, and most details will not be included, but references to the literature are given for the precise definitions and proofs.

We return now to the issue of what a neural net —from now on, just a "net"— is. As explained above, by a net one typically means a system which is built by linearly combining memory-free scalar elements, each of which performs the same nonlinear transformation $\sigma : \mathbb{R} \to \mathbb{R}$ on its input.

One of the functions σ that we will focus on is $\text{sign}\,(x) = x/|x|$ (zero for $x = 0$), or its relative,

*Research supported in part by US Air Force Grant 91-0343

the hardlimiter, threshold, or *Heaviside* function $\mathcal{H}(x)$, which equals 1 if $x > 0$ and 0 for $x \leq 0$ (one could define the value at zero differently; results do not change much). 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 $\tanh(\gamma x)$. (Equivalently, up to translations and change of coordinates, one usually finds the *standard sigmoid* $\sigma_s(x) = \frac{1}{1+e^{-x}}$.) 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.

Whenever time behavior is of interest, one also includes in nets dynamic elements, namely delay lines if dealing with discrete-time systems, or integrators in the continuous-time case.

In the static case, we consider nets formed by interconnections *without loops*, as otherwise the behavior is not always well-defined; we call these "feedforward" nets, in contrast to the terms "feedback," "recurrent," or "dynamic" nets used in the general case.

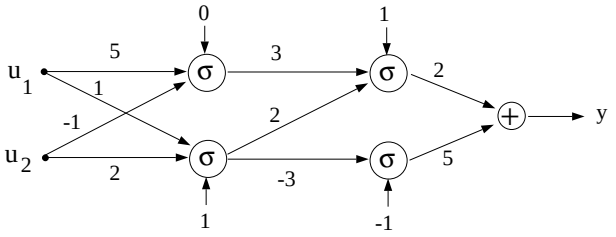

Figure 2: *Static Net Computing the Function* $2\sigma(3\sigma(5u_1-u_2)+2\sigma(u_1+2u_2+1)+1)+5\sigma(-3\sigma(u_1+2u_2+1)-1)$

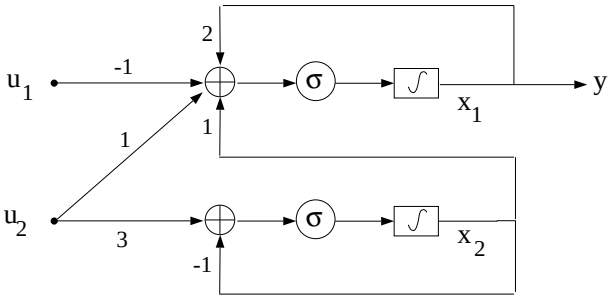

Figure 3: *Dynamic Net Representing the System* $\dot{x}_1=\sigma(2x_1+x_2-u_1+u_2), \dot{x}_2=\sigma(-x_2+3u_2), y=x_1$

1.1. What Do Nets Do?

In the purely static case, nets can be thought of as merely computing functions. Such memory-free nets are natural in at least two control applications:

— as direct implementations of static feedback laws;
— as "pattern recognition" devices which choose among various controllers.

In the latter case, different —typically simple linear— controllers are employed depending on which region of the state space or output-value space the current measurement is in; see Figure 4. This is known sometimes as "gain scheduling" or, if switching between controllers is replaced by a soft switching or blending of control laws, "fuzzy control." It is a useful control architecture when dealing with plants exhibiting qualitatively different modes of operation, such as reconfigurable aircraft or robotic manipulators under variable loads. In the first role, nets typically compute functions with continuous values, while in the second case nets are used as classifiers, producing a binary or more generally a finite-valued output. Mathematically, the first case gives rise to the study of questions of function approximation and interpolation, while the second focuses on classification. We will discuss results relating with both of these aspects. One of the conclusions is that static nets are capable of controlling fairly general nonlinear plants, if state feedback is available.

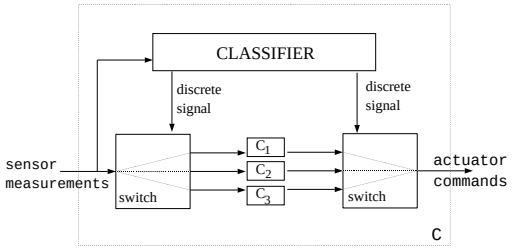

Figure 4: *Hierarchical Controller*

When memory elements are present, it is natural to view nets as nonlinear dynamical systems with inputs and outputs, in the sense of control theory. Dynamic nets play a role as identification models, as described above. In this context, theoretical results on approximation are relevant: To what extent can a general nonlinear system P be modeled by a net? Once that we agree to use a net as a model, a discussion of parameter identifiability is important: To what extent are the parameters describing the model —in other words, the weights— uniquely determined by the input/output behavior to be matched?

Dynamic nets also arise as controllers, in the case in which direct state feedback is impossible and only partial measurements are available. (A general introduction to dynamic feedback is given in the textbook [13], which should also be consulted for all undefined terms from control theory.) Often dynamic controllers may be viewed as static controllers for augmented systems: adding parallel integrators, or delay lines, to the plant P and then controlling the enlarged system by static feedback can be interpreted in this fashion. Sometimes, however, new issues arise, as discussed in the paper [10] which dealt with piecewise linear regulation, an area which is very closely related to the one of using dynamic nets as controllers.

1.2. Some Definitions

To fix terminology, we provide some definitions and notations.

The symbol σ will always denote a fixed function $\sigma : \mathbb{R} \to \mathbb{R}$. For each positive integer r, we let $\sigma_r : \mathbb{R}^r \to \mathbb{R}^r$ be the map that lets σ act coordinatewise, that is, $\sigma_r((x_1, \ldots, x_r)') = (\sigma(x_1), \ldots, \sigma(x_r))'$ (prime indicates transpose), but we drop the subscript r when clear from the context. When dealing with continuous-time systems containing σ, we implicitely assume that σ is globally Lipschitz, so that solutions are defined for all times, though far less is needed. Usually, σ is taken to be one of the sigmoidal-type maps discussed earlier.

Feedforward Nets

We say that a a function $f : \mathbb{R}^m \to \mathbb{R}$ is *computable by a zero-hidden-layer net* if it is an affine function, that is, there exist a vector $v \in \mathbb{R}^m$ and a scalar $\tau \in \mathbb{R}$ such that $f(u) = v.u + \tau$, where the dot indicates inner product. For any integer $d \geq 1$, the function $f : \mathbb{R}^m \to \mathbb{R}$ is *computable by a d-hidden-layer net (with processsors of type σ)* if there exist an integer l, constants $w_1, \ldots, w_l \in \mathbb{R}$, and functions $f_1, \ldots, f_l$ so that $f(u) = \sum_{i=1}^{l} w_i \sigma(f_i(u))$ and each f_i is computable by a $(d-1)$-hidden-layer net.

In other words, the functions computable by nets with no hidden layers are those in the span of the coordinate functions and the constants, and those computable by d layers constitute the span of the functions $\sigma(f(x))$, for f computable with one less layer. Note that constant terms (or "biases" in neural net terminology) can always be included in the above sum, as one could take one of the f_i's to be a constant. A d-hidden-layer net is sometimes called a "$(d+2)$-layer net" if one counts the inputs and outputs as a layer. We prefer the hidden-layer terminology, as it is less ambiguous. Note that a function f is computable by a one-hidden-layer net (it is a "1HL" function) if there are real numbers $w_0, w_1, \ldots, w_k$, $\tau_1, \ldots, \tau_k$, and vectors $v_1, \ldots, v_k \in \mathbb{R}^m$ such that, for all $u \in \mathbb{R}^m$, $f(u) = w_0 + \sum_{i=1}^{k} w_i \, \sigma(v_i.u + \tau_i)$. We also say in this case that f is a net "with k hidden neurons of type σ" or just that f is a "(k, σ)-net."

The results to be given on the feedforward case will deal with $d = 1$ or $d = 2$.

A *function computable by a net with possible direct input to output connections (and d hidden layers)* is by definition a function $g : \mathbb{R}^m \to \mathbb{R}$ of the form $Fu + f(u)$, where F is linear and f is computable by a d-hidden-layer net as above. For multivariable maps $f : \mathbb{R}^m \to \mathbb{R}^p$, "computable by a d-hidden-layer net" means by definition that each coordinate function $f_i : \mathbb{R}^m \to \mathbb{R}$, $i = 1, \ldots, m$ is so computable, and similarly if direct connections are allowed. For instance, computable by a one-hidden-layer net means that F factors as

$$F(u) = F_1(\sigma_r(F_2(u))) \, , \tag{1}$$

where $F_1 : \mathbb{R}^r \to \mathbb{R}^p$ and $F_2 : \mathbb{R}^m \to \mathbb{R}^r$ are affine maps.

It is by now well-known that functions computable by nets with a *single* hidden layer can approximate continuous functions, uniformly on compacts, under only weak assumptions on σ. We review below this fact, together with some results relating to the capabilities of such nets for solving interpolation and classification problems. We also explain that for many problems, including those typically found in nonlinear control, *two* hidden layers are sometimes necessary.

Feedback or Recurrent Nets

Consider recurrent nets in discrete or continuous time. In the first case, there are n dynamic units whose values at time t, $x_i(t) \in \mathbb{R}, i = 1, \ldots, n$, evolve according to difference equations $(\Delta x_i)(t) = \sigma(w_i(t))$, where Δ indicates time-shift: $(\Delta x)(t) = x^+(t) = x(t+1)$. In the continuous time case, these are differential equations, and Δ stands for the time-derivative operator: $(\Delta x)(t) = \dot{x}(t) = \frac{d}{dt}x(t)$. In either case, $w_i(t)$ is the input at time t to the ith device, which is a linear combination of the states of all the units as well as external signals. When σ= identity, we are dealing with ordinary linear control systems. As usual in control theory, we also assume that a particular set of measurements is of interest, and this is modeled by specifying an ouput map $y(t) = Cx(t)$.

Thus, we are led to consider systems that can be described by vector equations of the following type (deleting the time argument t):

$$\Delta x \; = \; \sigma_n(Ax + Bu) \, , \;\; y = Cx \, , \tag{2}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $C \in \mathbb{R}^{p \times n}$, and $\Delta = x^+$ or $= \dot{x}$ in discrete or continuous time respectively. We call these *σ-systems* or *recurrent nets*.

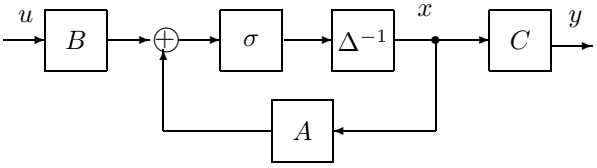

Figure 5: Recurrent Net

A constant vector c could be added to the right-hand side of equation (2), to represent what are usually called "biases" in the neural net literature, that is, in order to allow affine rather than just linear combinations of states and inputs, leading to equations $\Delta x = \sigma_n(Ax + Bu + c)$. It is often simpler to ignore this term by first adding a constant coordinate $x_0(t) \equiv \mu$, where $\mu \neq 0$ and then replacing each entry c_i by the linear term $(c_i/\mu)x_0$, which can then be absorbed into the A matrix. (In continuous-time, assuming that $\sigma(0) = 0$, then x_0 evolves according to $\dot{x}_0 = \sigma(0)$; in discrete-time, we may take any nonzero element in the image of σ, $\mu = \sigma(\nu)$, and then use the update $x_0^+ = \sigma((\nu/\mu)x_0)$.)

Mathematically, σ-systems are interesting, among other reasons, because:

— they are a powerful model of computation;

— they can approximate rather arbitrary plants.

Such systems have been proposed as models of large scale parallel computation, since they are built of potentially many simple processors. Electrical circuit

implementations of σ-systems, employing resistively connected networks of n identical nonlinear amplifiers, with the resistor characteristics used to reflect the desired weights, have been suggested as analog computers, in particular for solving constrained optimization problems and for implementing content-addressable memories. 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 the gradient-descent optimization (the so-called "dynamic backpropagation" procedure) of some cost criterion.

2. 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. If σ is k-times continuously differentiable, we say that it is *k-universal* if these restrictions of functions of $\mathcal{F}_\sigma$ are dense in $C^k[\alpha, \beta]$, the set of C^k functions with the metric of uniform convergence of all derivatives up to order k.

Note that, at this level of generality, we are not requiring anything besides density. In practical applications, it may be desirable to consider special classes of functions σ, such as those used in Fourier analysis or wavelet theory, for which it is possible to provide "reconstruction" algorithms for finding, given an $f : [\alpha, \beta] \to \mathbb{R}$, a set of coefficients a_i, b_i, c_i that result in an approximation of f within a desired tolerance.

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 [3] that any σ which is continuous, nonconstant, and bounded is universal (see also [2] for related results). Moreover, he also proved there (Theorem 3 in the paper) that if in addition to the above, σ is of class C^k, then it is also k-universal, for each positive integer k. (M. Leshno has recently notified this author that he has a new result showing 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 $\mathcal{F}_{\text{sign}}$; then each sign function is approximated by $\sigma(\gamma x)$, for large enough positive γ.

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, that is, those that can be written in the factored form (1), is dense in $C^0(K)$. A similar situation holds for k-universality.

One can also establish density results for L^q spaces, $q < \infty$, (use density of continuous functions,) but not in L^∞, which causes serious problems in nonlinear feedback control (see below).

2.3. 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 σ-systems, 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 σ-system. 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 $\widetilde{n} \neq n$). 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 $\alpha : \mathbb{R}^{\widetilde{n}} \to \mathbb{R}^n$ and $\beta : \mathbb{R}^n \to \mathbb{R}^{\widetilde{n}}$ 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 $\widetilde{x}(t) := \widetilde{\phi}(t, \beta(x_0), u)$; 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 σ-system $\widetilde{\Sigma}$ 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 [6] and [5], 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 1HLN function as in equation (1) 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 [13], 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 σ-system. 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

$$\begin{aligned} \dot{x}_1 \text{ [or } x_1^+ \text{]} &= T\sigma_r(A_1x_1 + A_2x_2 + \alpha + Bu) + \beta_1 \\ \dot{x}_2 \text{ [or } x_2^+ \text{]} &= \beta_2 \end{aligned}$$

and the output function is $y = C_1x_1 + C_2x_2$ in these coordinates. Write $n_2 = n - \operatorname{rank} T$ for the size of the x_2 variable. This is essentially a σ-system 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: $\dot{z}_1 \text{ [or } z_1^+ \text{]} = \sigma_r(A_1Tz_1 + A_2x_2 + \alpha + Bu) + \widetilde{\beta}_1$ and output $y = C_1Tz_1 + C_2x_2$. 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 approximations to signal processing see [5], and [6] for applications to control and identification.

2.4. Number of Units

As discussed earlier, pattern recognition is involved in the use of nets as selectors of lower-level controllers. In this context, it is interesting to ask about the number units that is needed in order to solve a given classification or interpolation task. We formalize this as follows. A *labeled sample* is a finite set $S = (u_1, y_1), \ldots, (u_s, y_s)$, where the $u_1, \ldots, u_s \in \mathbb{R}^m$ are distinct and $y_1, \ldots, y_s \in \mathbb{R}$ are the "labels". (The labels are *binary* if $y_i \in \{0, 1\}$.) A *classifier* is a function $F : \mathbb{R}^m \to \mathbb{R}$. The *error* of F on the labeled sample S is defined as $E(F, S) := \sum_{i=1}^s \|F(u_i) - y_i\|^2$.

A set $\mathcal{F} = \{F_{\vec{w}} : \mathbb{R}^m \to \mathbb{R}, \vec{w} \in R^r\}$ of functions parameterized by $\vec{w} \in R^r$, $r = r(\mathcal{F})$, is an *architecture*. An example is that of nets with no direct i/o connections and $m{=}1$ inputs, k hidden units; here the parameter set has dimension $r{=}3k + 1$. The sample S is *loadable into* $\mathcal{F}$ iff $\inf_{\vec{w}\in\mathbb{R}^r} E(F_{\vec{w}}, S) = 0$. The *capacity* $c(\mathcal{F})$ of $\mathcal{F}$ is defined by requiring that $c(\mathcal{F}) \geq k$ iff every $|S| = k$ is loadable. (This was called the "testing dimension" in [7].) That is, $c(\mathcal{F}){=}\infty$ means that all finite S are loadable, and $c(\mathcal{F}){=}k <\infty$ means that each S of cardinality $\leq k$ is loadable but some S of cardinality $k + 1$ is not. In [11], we studied the relation between capacity and number of neurons, for nets with one hidden layer and Heaviside or sigmoidal activations; note that large capacity $\approx$ memorization, so this is especially relevant for learning issues. Next we review some results from [11].

Consider first the case of input dimension $m{=}1$, that is, nets with one input, and k hidden units of type σ

in one layer. Observe that there are $3k+1$ parameters (appearing nonlinearly) —or $3k+2$ if allowing direct connections,— though for $\mathcal{H}$, effectively only $2k+1$ matter. (In the case of the standard sigmoid, a Jacobian computation shows that these parameters are independent.) Let $\mathcal{F}_{\mathrm{INTP},k,\theta}$ be the set of all functions like this, with no direct connections, and use a superscript "d" if direct connections are allowed. Finally, let $\mathcal{F}_{\mathrm{CLSF},k,\theta}$ be the set of $\{0,1\}$-valued functions of the form $\mathcal{H}(f(x))$, for f a (k,σ)-net.

We study scaling properties as $k \to \infty$. Let, for each σ: $\mathrm{CLSF}(\sigma) := \liminf_{k\to\infty} c(\mathcal{F})/r(\mathcal{F})$, for $\mathcal{F} = \mathcal{F}_{\mathrm{CLSF},k,\theta}$, and $\mathrm{INTP}(\sigma) := \liminf_{k\to\infty} c(\mathcal{F})/r(\mathcal{F})$, for $\mathcal{F} = \mathcal{F}_{\mathrm{INTP},k,\theta}$. Define similarly $\mathrm{CLSF}^{\mathrm{d}}, \mathrm{INTP}^{\mathrm{d}}$ when allowing direct connections.

For any map σ, we consider the property (S1): $\exists\ \lim_{x\to+\infty}\sigma(x) \neq \lim_{x\to-\infty}\sigma(x)$ and the property (S2): $\exists\, c$ s.t. θ is differentiable at c and $\theta'(c) = \eta \neq 0$.

Then, for classification, we have:

$$\boxed{\mathrm{CLSF}(\mathcal{H}) = 1/3,\ \mathrm{CLSF}^{\mathrm{d}}(\mathcal{H}) = 2/3,\ \mathrm{CLSF}(\sigma) \geq 2/3}$$

assuming that σ is so that properties (S1)-(S2) are satisfied. The last bound is best possible, in the sense that for the piecewise linear π we have $\mathrm{CLSF}(\pi) = 2/3$, while it is the case that $\mathrm{CLSF}(\sigma) = \infty$ for some "nice" (even, real-analytic) functions σ satisfying (S1)-(S2).

Regarding continuous-valued interpolation, we have the results:

$$\boxed{\mathrm{INTP}(\mathcal{H}) = 1/3,\ \mathrm{INTP}^{\mathrm{d}}(\mathcal{H}) = 1/3,\ \mathrm{INTP}(\sigma) \geq 2/3}$$

assuming in the last case that (S1)-(S2) hold and σ is continuous. Again here, $\mathrm{INTP}(\pi) = 2/3$, and we can also show that $2/3 \leq \mathrm{INTP}(\sigma_{\mathrm{s}}) \leq 1$ for the standard sigmoid (the proof of the upper bound in this latter case involves some algebraic geometry; the value may be infinite for more general sigmoids). Furthermore, the inequality $\mathrm{INTP}(\sigma) \geq 1/3$ holds for any universal nonlinearity.

The paper [11] includes also results when the number of inputs $m > 1$. Other closely related concepts were studied there as well, such as notions of "capacity" defined via the shattering of random sets, and Vapnik-Chervonenkis (VC) dimension measures, again comparing the power of Heaviside nets and sigmoidal nets, as well as the effect of direct i/o connections. In the multi-input case, it is interesting to study the particular case of binary inputs and the scaling with respect to input size, for natural Boolean functions; this gives a connection with theoretical computer science issues in the area of "circuit complexity" as discussed in [4].

2.5. Two-Hidden Layers, Inverse Problems

Consider now the following *inversion* problem: *Given a continuous function $f : \mathbb{R}^p \to \mathbb{R}^m$, a compact subset $C \subseteq \mathbb{R}^m$ included in the image of f, and an $\varepsilon > 0$, find a function $\phi : \mathbb{R}^m \to \mathbb{R}^p$ so that $\|f(\phi(x)) - x\| < \varepsilon$ for all $x \in C$.* One wants to find a ϕ which is computable by a net, as done in global solutions of inverse kinematics problems —in which case the function f is the direct kinematics. It is trivial to see that in general discontinuous functions ϕ are needed, so continuous σ cannot be used. However, and this is the interesting part, [12] establishes that nets with just one hidden layer, even if discontinuous σ is allowed, are *not* enough to guarantee the solution of all such problems. On the other hand, it is shown there that nets with two hidden layers (using $\mathcal{H}$ as the neuron type) are sufficient, for every possible f, C, and ε. The basic obstruction is due, in essence, to the impossibility of approximating by single-hidden-layer nets the characteristic function of any bounded polytope, while for some (non one-to-one) f the only possible one-sided inverses ϕ must be close to such a characteristic function. This brings up the need to revive the study of training of nets with discontinuous activations, an area that had been left relatively aside during the recent interest in neural networks, but which can be attacked by means of modern techniques of nonsmooth optimization and set analysis.

2.6. State Feedback

We now make several remarks about nets for implementation of state-feedback controllers. The objective, given a system

$$\dot{x} = f(x,u)$$

with $f(0,0)$=0, is to find a stabilizer u=$k(x)$, $k(0)$=0, making x=0 a globally asymptotically stable state of the closed-loop system $\dot{x}$=$f(x,k(x))$. The first remark is that the existence of a *continuous* stabilizer k is essentially equivalent to the possibility of stabilizing using 1HL nets (with any desired continuous σ). (Thus the simple classes of systems studied in many neurocontrol papers, which are typically feedback-linearizable and hence continuously stabilizable, can be controlled using such 1HL nets.)

More precisely, assume that f is twice continuously differentiable, that k is also in C^2, that the origin is an exponentially stable point for $\dot{x}$=$f(x,k(x))$, and that K is a compact subset of the domain of stability. Pick any σ that is 2-universal (most interesting twice-differentiable scalar nonlinearities will do, as recalled earlier). Then, we can conclude that there is also a different k, this one a 1HL net with neurons σ, for which exactly the same stabilization property holds. (Sketch of proof: one only needs to show that if $k_n \to k$ in $C^2(K)$, with all $k_n(0)$=0 —this last property can always be achieved by simply considering $k_n(x) - k_n(0)$ as an approximating sequence— then $\dot{x}$=$f(x,k_n(x))$ has the origin as an exponentially stable point and K is in the domain of attraction, for all large n. Now, the proof of Theorem 12 in [13] shows that there is a neighborhood V of zero, independent of n, where exponential stability will hold, for all n sufficiently large, because $f(x,k_n(x))$=$A_n x + g_n(x)$, with $A_n \to A$ and with $g_n(x)$ being $o(x)$ *uniformly* on x. Now continuity of solutions on the right-hand side gives the result globally on K.

In general, continuous stabilizers fail to exist, as dis-

cussed for instance in [13], Section 4.8. Thus 1HLN feedback laws, with continuous σ, do not provide a rich enough class of controllers. This motivates the search for discontinuous feedback, and maps of the type studied in this paper provide a computational paradigm in which to pose questions of existence of such more general controllers. It is easy to provide examples where 1HL $\mathcal{H}$-nets will stabilize but no net with continuous activations (hence implementing a continuous feedback) will. More surprisingly, 1HLN feedback laws, even with $\mathcal{H}$ activations, are not in general enough —intuitively, one is again trying to solve inverse problems— but two hidden layer nets using $\mathcal{H}$ (and having direct i/o connections) are always sufficient. More precisely, [12] shows that the weakest possible type of open-loop asymptotic controllability is sufficient to imply the existence of (sampled) controllers built using such two-hidden layer nets, which stabilize on compact subsets of the state space.

3. Recurrent Nets

We saw above that recurrent nets, i.e. σ-systems (2), have certain approximation properties. This suggests their suitability as identification models, that is, models for plants P. An alternative way of characterizing the power of a class of models is in terms of the computations that they can perform. Another natural question deals with parameter (weights) identification from i/o data. We now offer some remarks on these two topics.

3.7. Computability

In the paper [8] with Hava Siegelmann, and related work, we investigated the computational capabilities of σ-systems, seen from the point of view of formal language theory. There we studied discrete-time σ-systems with $\sigma = \pi$. When restricted to binary inputs and outputs, the computations carried out by such systems can be viewed as transformations on languages, and hence can be compared with other, more classical, models of computation. Our main results in [8] are: (a) with rational matrices A, B, and C, (and rational initial state), such systems are equivalent, up to polynomial time, to Turing machines; (b) with real matrices, all possible binary functions, recursive or not, are "computable". Thus one may reduce questions in computability to questions about realizability by π-systems (2). When time constraints are imposed on the computation length, a rich theory arises. For instance, polynomial-time computation by π-systems with real coefficients —a form of analog computing— is equivalent to a notion of computability (nonuniform polysize circuits) that appeared in abstract computation theory. Even the "P=?NP" problem has a version for π-systems; see [9].

3.8. Identifiability

In [1], with Francesca Albertini, we were interested in studying the following issue: To what extent does the function of a σ-system, that is to say, the "black box" behavior mapping external inputs to outputs, uniquely determine the coefficients of the matrices A, B, C? A precise formulation, for continuous-time, is as follows. Assume that the network is started at $x(0) = 0$ and the corresponding state and output trajectories $x(\cdot)$ and $y(\cdot)$ are generated. In this manner to each triple (A, B, C) we associate an input-output mapping $\lambda_{(A,B,C)} : u(\cdot) \mapsto y(\cdot)$. We wish to know to what extent are the matrices A, B, C determined by the i/o mapping $\lambda_{(A,B,C)}$. If σ would have been the identity, linear systems theory tells us that, generically, the triple (A, B, C) is determined only up to an invertible change of variables in the state space. That is, except for degenerate situations that arise due to parameter dependencies —non-controllability or non-observability— if two triples (A, B, C) and $(\overline{A}, \overline{B}, \overline{C})$ give rise to the same i/o behavior then there is an invertible matrix T such that the interlacing conditions $T^{-1}AT = \overline{A}$, $T^{-1}B = \overline{B}$, $CT = \overline{C}$ hold. This is the same as saying that the two systems are equivalent under a linear change of variables $x(t) = T\overline{x}(t)$. Conversely, any such T gives rise to another system with the same i/o behavior. These classical facts apply only when σ is linear, as we discuss in [1]. There we show that for nonlinear activations —under very mild assumptions— the natural group of symmetries is far smaller than that of arbitrary nonsingular matrices, being instead just a finite group. We prove in [1] that if two nets give rise to the same i/o behavior, then (again, as for linear systems, generically,) a matrix T will exist, providing the above interlacing equations, but having the special form of a permutation matrix composed with a diagonal matrix performing at most a sign reversal at each neuron. That is, the input/output behavior uniquely determines all the weights, except for a reordering of the variables and, for odd activation functions, possible sign reversals of all incoming and outgoing weights. A consequence of this uniqueness is that a dimensionality reduction of the parameter space, as done for linear systems via canonical forms, is not possible for recurrent nets.

References

[1] Albertini, F., and E.D. Sontag, "For neural networks, function determines form," submitted.

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

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

[4] Maass, W., G. Schnitger, and E.D. Sontag, "On the computational power of sigmoid versus boolean threshold circuits," *Proc. of the 32nd Annual Symp. on Foundations of Computer Science*, 1991: 767-776.

[5] 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.

[6] 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.

[7] Romanik, K., "Approximate testing and learnability," in *Proc. Fifth ACM Workshop on Computational Learning Theory*, Pittsburgh, July 1992.

[8] 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; see also SYCON Report 91-11, Rutgers Center for Systems and Control, November 1991.

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

[10] Sontag, E.D., "Nonlinear regulation: The piecewise linear approach," *IEEE Trans. Autom. Control* **AC-26**(1981): 346-358.

[11] Sontag, E.D., "Feedforward nets for interpolation and classification," *J. Comp. Syst. Sci.*, to appear, 1992.

[12] Sontag, E.D., "Feedback Stabilization Using Two-Hidden-Layer Nets," in *Proc. Amer. Automatic Control Conference*, Boston, June 1991, pp. 815-820. To appear in *IEEE Trans. Neural Networks*.

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