NEURAL NETWORKS FOR CONTROL*

Eduardo D. Sontag
Department of Mathematics, Rutgers University
New Brunswick, NJ 08903, USA

Abstract

This paper starts by placing neural net techniques in a general nonlinear control framework. After that, several basic theoretical results on networks are surveyed.

1 Introduction

The field commonly referred to as "neuro-" or "connectionist" control has been the focus of a tremendous amount of activity in the past few years. As emphasized in [20], work in this and related areas represents to a great extent a return to the ideas of Norbert Wiener, who in his 1940s work on "cybernetics" drew no boundaries between the fields now called artificial intelligence, information theory, and control theory.

In this presentation, requested by the conference organizers, the goal is definitely *not* to provide an exhaustive or even a representative survey. Most recent major control conferences have had introductory courses devoted to the topic, and, in addition, many good overviews are available in the general literature; see for instance the papers [5], [20], [7], [16], and other papers in the books [27] and [39]. Rather, the first objective is to explain the general context in which this work has taken place, namely the resurrection of some very old ideas dealing with numerical and "learning" techniques for control —a rebirth due more than anything else to the availability of raw computational power, in amounts which were unimaginable when those ideas were first suggested. Starting from a very general control paradigm, the framework is progressively specialized until the particular case of neural networks is arrived at. After this, the paper deals with an aspect which is perhaps not adequately covered by other expository efforts: the study of the question *what is special about neural nets?*, understood in the sense of the search for mathematical properties which are not shared by arbitrary parametric models.

In keeping with the program outlined above, there will not be any discussion of implementation details, simulation results, or any other "practical" issues; once more, the reader is referred to the above survey papers for pointers to the relevant literature. Moreover, the availability of these papers and their extensive bibliographies has the advantage of allowing in-

*Research described here partially supported by US Air Force Grant AFOSR-91-0343. This paper was written in part while visiting Siemens Corporate Research, Princeton.

direct citations; a reference such as: [20:175] will mean "item [175] in the bibliography given in the paper [20]."

1.1 Learning and Adaptive Control

One of the main advantages claimed for neurocontrollers is their alleged ability to "learn" and "generalize" from partial data. What is meant by this, roughly, is that after being exposed to the "correct" control actions in several situations, the learning system should be able to react appropriately, by interpolation or extrapolation, to new situations. (Later the paper reviews one of the ways in which this idea can be framed, using the terminology of computational learning theory, but almost nothing has been done in applying such a formalization to control systems, so for now the use of terms such as learning is done in a completely informal manner.) The "learning" may be thought of as being performed "off-line," by a numerical algorithm during a training phase, or "on-line," during actual closed-loop operation. But the distinction between off-line and on-line is essentially one of modeling rather than a mathematical one; it mainly reflects the time span considered and the information available from the environment. Especially in the latter case, learning is closely related to adaptation in the usual sense of the word in control theory. (Actually, the term "adaptive control" is itself ill-defined; when studying nonlinear systems, adaptive control can often be seen simply as a form of control of partial states.)

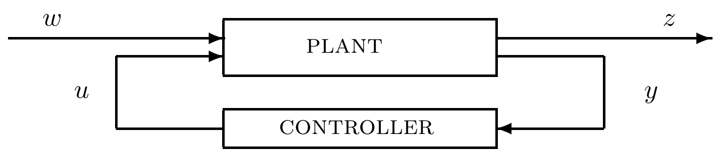
Still, when studying very specific types of systems, and using particular controller structures, many authors differentiate between adaptation and on-line learning. In practice, adaptive control tends to refer to the control of slowly changing systems (after a modeling distinction is made between state variables and parameters), and sudden changes in parameters can lead to transient behavior during which adaptation occurs and performance is degraded. Learning control, on the other hand, is a term that tends to denote controllers that adapt to various regions of the state or the parameter space, and which store the control laws that have been found to be appropriate for that region, to be later retrieved when the same operating circumstances are encountered, with no need for readaptation (but requiring "pattern recognition" capabilities to classify such previously learned situations in order to allow later for fast recognition). Clustering algorithms, neural networks, and large amounts of memory are usually associated with learning controllers.

2 Plants, Controllers, and Models

In order to provide some unity to the description of neurocontrol, the paper starts with a general paradigm, which is progressively specialized. This discussion should be understood as an informal one. It is *not* the purpose to give rigorous mathematical definitions, but rather to establish a language in which to frame the various applied issues that are being considered. It is far too early, given the current poor state of knowledge and level of results obtained, to attempt to provide a general theoretical setting for neurocontrol. So this section should be read as a philosophical, informal discussion. In addition, the reader will note that the term "neural network" will be used several times but is left undefined. The discussion depends on *nothing more* than the fact that nets provide a certain particular class of dynamical systems (or, in static situations, maps). By not defining the term, this fact is emphasized.

2.1 The Initial Paradigm

Take as a starting point the following basic control paradigm. The object to be controlled, called the "plant," interacts with its environment by means of two types of input and output channels.



The signal w contains all inputs that cannot be directly affected by the controller. This includes disturbances, measured or unmeasured, as well as signals to be tracked and other reference information. If the controller is to have access to some of these inputs, a pass-through connection is assumed to be part of the box labeled "plant" so that the respective information appears again in the measured output variable y; this variable contains as well all other data which is immediately available to the controller, such as values of state variables. The signal u is the part of the input that can be manipulated, and the signal z encodes the control objective. The objective may be for instance to drive this variable (which might be the difference between a certain function of the states and a desired reference function) to zero. If needed, one may assume that the control values themselves appear in y, and this is useful to keep in mind when the controller will consist of separate subsystems, as done later.

This setup can be formalized in various ways, using abstract definitions of control systems as in [33], but the current discussion will be completely intuitive.

2.2 Learning Control Structure

The first refinement of the basic control paradigm that one may consider is the splitting of the controller into a fixed part and a "learning" or adaptable part; see the Figure (the signal flow is drawn from left to right, for ease of reference).

$$\begin{tikzcd}
{} \arrow[r, "y"'] \arrow[rrr, bend right] & \text{PF} \arrow[r, "\eta"'] & \text{LC} \arrow[r, "v"'] & \text{PC} \arrow[r, "u"'] & {}
\end{tikzcd}$$

The box labeled LC represents a "learning controller" as discussed later. The other two parts —which may be missing— are used to indicate that part of the controller which does not change. The box labeled PF represents a "prefiltering" device; it may perform for instance one of the following (not necessarily distinct) functions:

Feature Extraction. This may be based on Fourier, wavelet, or other numeric transforms, or on symbolic procedures such as edge detection in images. One may also include in this category the computation of basic operations on measured signals y; for example, obtaining all pairwise products $y_i y_j$ of coordinates allows correlations to be used as inputs by the subsequent parts of the controller.

Sampling. Quantization, time-sampling, and general A/D conversion.

Template Matching. The output of PF may be a binary signal which indicates if y equals a specific value, or more generally, if this signal belongs to a predetermined set. More generally, its output may be a discrete signal that indicates membership in various, possibly overlapping, receptive fields, that is, a vector of the form $(\chi_{S_1}(y), \ldots, \chi_{S_k}(y))$, where each S_l is a subset of the space where instantaneous values of y lie. When these sets consist of just one point each, pure template matching results. More interestingly, one may have spheres centered at various points in y-space, which allows a certain degree of interpolation. Instead of characteristic functions, a more "fuzzy" weighting profile may be used, corresponding to degrees of membership. The possible overlap of regions of influence allows for distributed representations of the input; CMAC-type controllers (see e.g. [7:2] for many references) employ such representations together with hashing memory addressing systems for associative recall. Localized receptive fields play a role quite analogous, for approximation problems, to splines, and this connection is explored in [20:108].

The box labeled PC represents a "precomputed (part of the) controller" which also is invariant in structure during operation. Included here are, as examples:

Gain Scheduled Control Laws. A finite number of precomputed compensators, to be switched on according to the value of a discrete signal v, or to be blended according to weights indicated by a continuous v ("fuzzy" control applications).

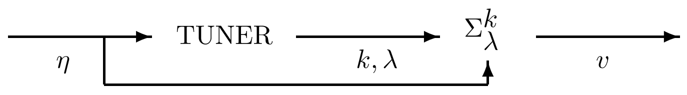
A State Feedback Law. For instance, v may be a state estimate produced by LC (which then plays the role of an observer), and PC may be a state feedback law. Recall that the observation y may be assumed to include, by making the "plant" larger, information on the input signal u; or one may include a backward path from PC into LC, for use by the observer.

Sampling. As with the prefiltering box, one may include here smoothers, D/A conversion, hold devices, and so forth.

This structure is only given for expository purposes; in practice the boundaries between prefiltering, learning, and precomputed controller are often blurred. (For instance, one may use various clustering algorithms for the PF part, but these algorithms may in turn be "learned" during operation of the controller.)

2.3 Refining the LC Block

Refining further the general control paradigm in order to arrive to neurocontrol techniques, assume that the block LC splits into two parts, a "tuner" and a box representing a system (dynamical, or a static mapping) with parameters determined by the tuner.



The block labeled Σ_λ^k corresponds to a system or static mapping parameterized by a vector λ of real parameters and operating on the input signal η. There is as well a discrete parameter k, so that the size of the vector λ is a function of k. For instance, k may determine a choice of number of state variables in a controller, or number of neurons —or more generally an encoding of an "architecture"—for a neural network, and λ encompasses all free parameters in the corresponding model. The tuner (see arrow) selects the parameters of Σ_λ^k according to some "learning procedure" which uses all available information (included as part of η).

In adaptive control, one might include as models Σ_λ^k linear systems of order k, parameterized by λ. The tuner would then include an adaptation rule for this controller. Or, the tuner might be used to set the parameters in an identification model Σ_λ^k, and the result of identification could be a signal used to drive a precomputed controller. The term "precomputed" could be taken to mean, in this last case, "to be computed by a standard algorithm" such as LQG design.

This is the place where neural nets are most often used in applications. Typically the family Σ_λ^k will include a class of systems and/or controllers that is sufficiently rich to represent a wide variety of dynamical behaviors (if used as an identification model, or for representing controllers with memory), or static maps (for pattern recognition applications, or for implementing static controllers). This motivates the search for such rich classes of models, and the study of their representational and approximation properties. Many choices are possible. For dynamical systems, one may pick linear or bilinear systems, or more generally systems of the form

$$\dot{x} = f(x, \lambda, u),\ y = h(x, \lambda) \tag{1}$$

(or analogously in discrete time) where the functional form of f and h is predetermined, for each state space dimension n. This functional form may simply be a linear combination of a set of basic functions (polynomials, splines with fixed nodes, and so forth), with the λ variables providing the coefficients of the combination. Or, the parameters may appear nonlinearly, as in rational parameterizations, splines with free nodes, and neural nets (whatever this last term means). Instead of state-space models, one may use parameterized classes of input/output behaviors; in that case, the parameters might correspond to coefficients in transfer functions or in coprime factorizations, or kernels in Volterra or Chen-Fliess series expansions.

2.4 A (Mini-) Illustration

The following is an example of a parametric adaptation rule used in neurocontrol applications. Although very simplified and for a very special case, it illustrates some of the typical ideas. (If desired, one could formulate in various manners this example in terms of the above general paradigms, but as explained earlier, those structures have no formal meaning and are only given to help understand the literature. In this particular case, the methodology will be clear.) Assume that the plant to be controlled is described, in discrete time, by the scalar equation

$$x^+ = f(x) + u$$

(use the superscript + to indicate time shift). The mapping f is unknown but x can be observed. The objective is to track a known reference signal r; more precisely, to obtain

$$x(t) - r(t-1) \to 0 \ \text{ as } \ t \to \infty$$

by means of an appropriate control. If the mapping f were known, one would use the control law $u(t) = -f(x(t)) + r(t)$ in order to achieve perfect tracking. (Note that this system is easy to control; most applications of neurocontrol have been indeed for systems that would be easy to control if perfectly known, most often for feedback-linearizable systems. Later, the paper considers some problems that arise when the system itself is truly nonlinear, and the constraints that this imposes on neural controller structure.) Since f is unknown, one proposes a parametric model such as

$$x^{+} = F(x, \lambda) + u \tag{2}$$

where F is "rich enough" and attempts to find a $\hat{\lambda}$ so that $F(x, \hat{\lambda}) \approx f(x)$ for all x. Once such an approximation is found, the certainty equivalence controller $u(t) = -F(x(t), \hat{\lambda}) + r(t)$ is used, possibly updating $\hat{\lambda}$ during operation. For instance, F may be a generic polynomial in x of some large degree k, and λ is a vector listing all its coefficients.

How should the parameter(s) λ be estimated? One possibility is to assume that (2) is the true model but it is subject to driving and observation noise:

$$x^{+} = F(x, \lambda) + u + \xi_1, \;\; y = x + \xi_2$$

where ξ_1, ξ_2 are independent 0-mean Gaussian variables and y is being observed. This leads to an extended Kalman filtering formulation, if one assumes also a stochastic model for the parameters: $\lambda^{+} = \lambda + \xi_3$ where ξ_3 is independent from the other noise components. One may then use the a posteriori EKF estimate $\hat{\lambda}[t|t]$ as $\hat{\lambda}$. The estimate $\hat{x}[t|t]$ can be used instead of x. (If only partial observations are available, an output mapping $y = h(x) + \xi_2$ may be assumed instead of the identity, and a similar procedure may be followed.) See for instance [25] and [23] for this type of parametric nonlinear adaptive control approach.

When parameterizations are linear in λ, this procedure, or one based on Lyapunov techniques for parameter identification —possibly in conjunction with sliding mode robust control, to handle the regions where the true f differs considerably from the best possible model of the form $F(x, \lambda)$— can be proved to converge in various senses; see especially [32], as well as for instance [30].

In general, however, there are absolutely no guarantees that such a procedure solves the tracking problem, and only simulations are offered in the literature to justify the approach. (An exception are certain local convergence results; see e.g. [12].) Moreover, the control problem itself was capable of being trivially solved by feedback linearization, once that the plant was identified. Note also that the nonlinear adaptive control approach is independent of the neural nets application; by and large there have been *no theoretical results* given in the literature, for adaptive control, that would *in any way use properties particular to neural nets*. This is typical of most neural networks applications at present, and it is the main reason for concentrating below instead on the basic representation, learnability, and feedback control questions involved.

2.5 Other Techniques for Control

When systems are not feedback linearizable, nonlinear control becomes a very hard problem, even leaving aside identification issues (as an illustration, see [33], Section 4.8, and the many references given there, for stabilization questions). There are several adaptation approaches which have been popular in neurocontrol and which correspond to various combinations of tuners and parametric models. Many of these approaches are very related to each other, and they are really independent of the use of neural networks; in fact, they represent methods of numerical control that have been around since at least the early 1960s. Briefly described next are some of the ideas that have appeared in this context, with the only objective of helping the reader navigate through the literature.

Identifying a State Feedback Law. Here one assumes a parametric form for a state feedback law $u = k(\lambda, x)$ and the parameter λ is chosen so as to minimize some cost criterion. Obviously, the class of feedback functions that can be represented in this manner should be rich enough to contain at least one solution to whatever the ultimate control objective is. (In practice, this is simply assumed to be the case.) A cost criterion that depends on the parameter and the state is picked to be minimized, such as for instance:

$$J(\lambda, x_0) \;=\; \int_0^\infty Q(x(t), u(t))\, dt\,,$$

where $x(\cdot), u(\cdot)$ are the state and input obtained when using $u = k(\lambda, x)$ as a controller for the given system $\dot{x} = f(x, u)$ and $x(0) = x_0$. Now one may attempt to minimize $\max_x J(\lambda, x)$ over λ. This is the type of approach taken, for example, in [4], in which the viability problem (make the state stay in a desired set) is attacked using neural net controllers. (The minimax character of the problem makes it suitable for nondifferentiable optimization techniques.)

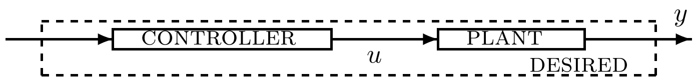
Estimate a Lyapunov or Bellman Function. A basic problem in feedback control theory, which also arises in many other areas (it is known as the "credit assignment problem" in artificial intelligence) is that of deciding on proper control actions at each instant in view of an overall, long-term, objective. Minimization of an integrated cost, as above, leads implicitly to such good choices, and this is the root of the Lagrangian or variational approach. Another possibility, which underlies the dynamic programming paradigm, is to attempt to minimize at each instant a quantity that measures the overall "goodness" of a given state. In optimal control, this quantity is known as the *Bellman* (or the cost-to-go, or value, function); in stabilization, one talks about *Lyapunov* (or energy) functions; game-playing programs use *position evaluations* (and subgoals); in some work that falls under the rubric of learning control, one introduces a *critic*. Roughly speaking, all these are variants of the same basic principle of assigning a cost (or, dually, an expectation of success) to a given state (or maybe to a state and a proposed action) in such a way that this cost is a good predictor of eventual, long-term, outcome. Then, choosing the right action reduces to a simpler, nondynamic, pointwise-in-time, optimization problem: choose a control that leads to a next state with least cost.

Finding a suitable measure in the above sense is a highly nontrivial task, and neural network research has not contributed anything conceptually new in that regard. The usual numerical techniques from dynamic programming (value or policy iteration) can be used in order to compute the Bellman function for an optimization problem. Other work has been based on posing a parametric form for a Lyapunov ("critic", etc) function $V(\lambda, x)$. In this latter mode, one attempts to fit parameters λ for both the Lyapunov function and the proposed feedback law $K(\lambda, x)$ simultaneously. This is done by adjusting λ after a complete "training" event, by means of, typically, a gradient descent step. The "adaptive critic" work by Barto and others (see [7] and the many references there) is one example of this approach, which is especially attractive when the overall goal is ill-defined in quantitative terms. For instance, the technique is often illustrated through the problem of controlling an inverted pendulum-on-a-cart system, in which the parameters are unknown and the only training signal allowed is the knowledge that the car has achieved a certain maximum displacement or that the pendulum angle has reached a maximum deviation. This type of work is also closely related to relatively old literature in learning control by researchers such as Mendel, Fu, and Sklansky in the mid to late 1960s and Narendra in the 1970s. See for instance [7:(64,74)] and [16:54].

Local Adaptive Control. One may consider several selected operating conditions, and design an adaptive (or a robust) controller for the linearization around each of them. (Operating conditions close to one of the selected ones give rise to linear models that are thought of as parameter variations.) The "learning part" consists in building an association between the current state and the appropriate controller. This is conceptually just a variant on the idea of gain-scheduling and use of a pattern recognition device to choose the appropriate gain, and it appears in many neurocontrol applications. There is nothing special in the use of "neural nets" as pattern recognizers or associative memories; any other reasonable technique would in principle apply equally well. Again, no mathematical analysis is ever given of overall performance.

Expert-Systems (Mimicking) Approach. Often a good controller may be available, but various reasons may make it worth to simulate its action with a neural net. For instance, a human expert may be good at a given control task and one may want to duplicate her behavior. In this case, one may be able to fit a parameterized class of functions, on the basis of observations of typical controller operation; see e.g. [7:111].

Feedforward Control. A popular technique in neurocontrol is closely related to model reference adaptive control. Here a controller, in the form for instance of a recurrent network (see below), is trained in such a manner that when cascaded with the plant the composite system emulates a desired model. The training is often done through gradient descent minimization of a cost functional.



A particular case that has been much explored experimentally —see for instance [7:(3,4,56,58,67,79,90)]— is the one where the desired model is a pure multiple delay (in discrete time) or integrator (continuous time), that is, the case of system inversion. A training set is obtained by generating inputs $u(\cdot)$ at random, and observing the corresponding outputs $y(\cdot)$ produced by the plant. The inverse system is then trained by attempting to fit the reversed pairs (y, u). Once that an inverse is calculated, a desired output $y_{\rm d}(\cdot)$ can be obtained for the overall system (possibly delayed or integrated) by feeding $y_{\rm d}$ to the controller. This approach is already subject to serious robustness problems even for linear systems. But in the nonlinear case there are major additional difficulties, starting with the fact that nice inverses can be assumed to exist at best locally, so training on incomplete data cannot be expected to result in a good interpolation or "generalization" capability, which after all is the main objective of learning control.

If only certain particular outputs $y_{\rm d}(\cdot)$ are of interest, then generating training data in the random manner just sketched is very inefficient. It is more reasonable in that case to train the controller so that when given those *particular* targets $y_{\rm d}$ it generates inputs to the plant that produce them. Given such a more limited objective, the procedure described above amounts to exhaustive search, and steepest descent is preferable. In order to apply steepest descent, one sets up an appropriate cost to be minimized. Since the derivative of this cost involves a derivative of the plant, this may lead to serious errors if the plant is not perfectly known, but only an experimentally obtained approximation is available. A bit more formally, the problem is as follows. One wishes to find a value λ so that, for certain $y_{\rm d}$,

$$P(C(y_{\rm d}, \lambda)) = y_{\rm d}$$

where the "plant" transformation P is approximately known and the "controller" C is a function of outputs and parameters. Equivalently, one attempts to minimize $F(\lambda) = \|P(C(y_{\rm d}, \lambda)) - y_{\rm d}\|^2$. The gradient flow in this case is $\dot{\lambda} = -\nabla_\lambda F(\lambda)$. Now, if instead of P one must work with an approximation P_1 of P, the gradient of F will be computed using derivatives of P_1 (understood in a suitable functional sense, for dynamical systems) rather than of P. Thus it is essential that the approximation of P by P_1 had been previously done in a topology adequate for the problem, in this case a C^1 topology, in other words, so that the derivative of P_1 is close to the derivative of P. This issue was pointed out in the neural nets literature, and used in deriving algorithms, in for instance [7:(46,104,105,106)].

2.6 Modeling via Recurrent Nets

An especially popular architecture for systems identification (models Σ_λ^k) has been that of recurrent neural nets. Described later are some theoretical results for these, but in practice the question that has attracted the most attention has been that of fitting parameters to observed input/output behavior. The work done has consisted mostly of a rediscovery of elementary facts about sensitivity analysis. Essentially, given a system as in Equation (1), an input $u(\cdot)$ on an interval $[0,T]$, and a desired final output $y_{\mathrm{d}}(T)$, one wants to find parameters λ such that the output $y(T)$, say for a given initial state, differs as little as possible from $y_{\mathrm{d}}(T)$, on this input $u(\cdot)$. (One may be interested instead in matching the entire output *function* $y(\cdot)$ to a desired function $y_{\mathrm{d}}(\cdot)$, that is, in minimizing an error functional such as $J(\lambda) = \int_0^T \|y(t) - y_{\mathrm{d}}(t)\|^2 dt$. This can be reduced to the previous case by adding a state variable $\dot{z}(t) = \|y(t) - y_{\mathrm{d}}(t)\|^2$ and minimizing $z(T)$, as routinely done in optimal control. Also, one may also be dealing with a finite set of such pairs u, y_{d} rather than just one, but again the problem is essentially the same.) In order to perform steepest descent, it is necessary to compute the gradient of $\|y(T) - y_{\mathrm{d}}(T)\|^2$ with respect to λ. Denoting by $\partial x(T)/\partial\lambda$ the differential of $x(T)$ with respect to λ, evaluated at the parameter values obtained in the previous iteration of the descent procedure, one must compute

$$\frac{\partial \|h(x(T)) - y_{\mathrm{d}}(T)\|^2}{\partial x(T)} \; \frac{\partial x(T)}{\partial \lambda} \tag{3}$$

(the parameters λ are omitted, for notational convenience.) Viewing the parameters as constant states, the second term can be obtained by solving the variational or linearized equation along the trajectory corresponding to the control $u(\cdot)$, for the system obtained when using the current parameters λ; see for instance Theorem 1 in [33]. (One "old" reference is [26].) Such an approach is known in network circles as the "real time recurrent learning" algorithm, and it is often pointed out that it involves a fairly large amount of variables, as the full fundamental solution (an $n \times n$ matrix of functions) must be solved for. It has the advantage of being an "online" method, as the equations can be solved at the same time as the input u is being applied, in conjunction with the forward evolution of the system. An alternative procedure is as follows. Since the full gradient is not needed, but only the product in (3) is, one may instead propagate the first term in (3) via the adjoint equation, again as done routinely in optimal control theory. This involves two passes through the data (the adjoint equation must be solved backwards in time) but less memory requirements. It is a procedure sometimes called "backpropagation through time". In discrete time, the difference between the two procedures is nothing more than the difference between evaluating a product

$$vA_T \dots A_2 A_1 \tag{4}$$

where $A_1, \dots, A_T$ are matrices and v is a vector, from right to left (so a matrix must be kept at each stage, but the procedure can be started before all A_i's are known) or from left to right (so only a vector is kept). The discussion of the relative merits of each approach seems to have consumed a major amount of effort in this area. Nothing especial about neural network models seems to have been used in any papers, except for some remarks on storage of coefficients. No global convergence theorems are proved.

3 Neural Nets

Artificial neural nets give rise to a particular class of parameterized controllers and models. What is meant precisely by the term neural net varies, depending on the author. Most often, it means a linear interconnection of memory-free scalar nonlinear units, supplemented by memory elements (integrators or delay lines) when dynamical behavior is of interest. The coefficients characterizing the connections, called "weights," play a role vaguely analogous to the concentrations of neurotransmitters in biological synapses, while the nonlinear elements in this over-simplified analogy correspond to the neurons themselves. In practice, one decides first on the class of "neurons" to be used, that is, the type of nonlinearity allowed, which is typically of the "sigmoidal" type reviewed below. The weights are "programmable" parameters that are then numerically adjusted in order to model a plant or a controller. In some of the literature, see e.g. [28], each scalar nonlinear unit acts on a scalar quantity equal to the distance between the incoming signal and a reference (vector) value; this is in contrast to operating on a linear combination of the components of the incoming signal, and gives rise to "radial basis function" nets, which are not treated here. See the textbook [18] for a clear and well-written, if mathematically incomplete, introduction to neural nets.

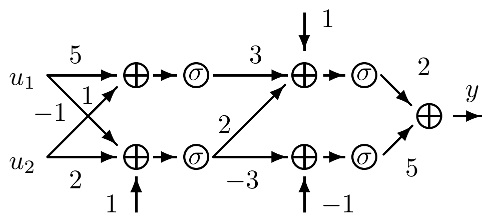
Motivating the use of nets is the belief —still not theoretically justified— that in some sense they are an especially appropriate family of parameterized models. Typical engineering justifications range from parallelism and fault tolerance to the possibility of analog hardware implementation; numerical and statistical advantages are said to include good capabilities for learning (adaptation) and generalization.

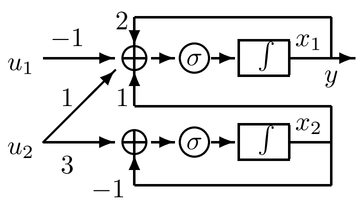
3.1 What are Networks

As explained above, by a ("artificial neural") 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 main functions σ used is $\operatorname{sign}(x) = x/|x|$ (zero for $x = 0$), or its relative, the hardlimiter, threshold, or *Heaviside* function $\mathcal{H}(x)$, which equals 1 if $x > 0$ and 0 for $x \leq 0$ (in either case, one could define the value at zero differently; results do not change much). In order to apply various numerical techniques, one often needs a differentiable σ that somehow approximates $\operatorname{sign}(x)$ or $\mathcal{H}(x)$. For this, 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 may use the *standard sigmoid* $\sigma_{\mathrm{s}}(x) = \frac{1}{1+e^{-x}}$. Also common in practice is a piecewise linear function, $\pi(x) := x$ if $|x| < 1$ and $\pi(x) = \operatorname{sign}(x)$ otherwise; this is sometimes called a "semilinear" or "saturated linearity" function.

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

In the static case, one considers nets formed by interconnections *without loops*, as otherwise the behavior may not be well-defined; these are called "feedforward" nets, in contrast to the terms "feedback," "recurrent," or "dynamic" nets used in the general case. The next Figure provides an example of a static net computing the function $y = 2\sigma[3\sigma(5u_1 - u_2) + 2\sigma(u_1 + 2u_2 + 1) + 1] + 5\sigma[-3\sigma(u_1 + 2u_2 + 1) - 1]$ and of a dynamic net representing the system $\dot{x}_1 = \sigma(2x_1 + x_2 - u_1 + u_2), \dot{x}_2 = \sigma(-x_2 + 3u_2), y = x_1$.





3.2 A Very Brief History

In the 1940s, McCulloch and Pitts introduced in [20:119] the idea of studying the computational abilities of networks composed of simple models of neurons. Hebb ([20:78]) was more interested in unsupervised learning and adaptation, and proposed the training rule known by his name, which is still at the root of much work, and which consists of reinforcing the associations between those neurons that are active at the same time.

Rosenblatt, in the late 1950s, pursued the study of "perceptrons" or, as they might be called today, "multilayer feedforward nets" such as the first one shown above, using the Heaviside activation $\mathcal{H}$. He developed various adaptation rules, including a stochastic technique. (He called the latter "backpropagation" but the term is now used instead for a gradient descent procedure. For a *very special* case, that of networks in which all nodes are either input or output nodes, a global convergence result was proved, which came to be known as the "perceptron convergence theorem." It is a widely held misconception that this latter special case was the main one that he considered.) The book [20:148] summarized achievements in the area. His goal was more the understanding of "real" brains than the design of artificial pattern recognition devices, but many saw and promoted his work as a means towards the latter. Unfortunately, the popular press —in Rosenblatt's words, "which (treated Perceptrons) with all the exhuberance and discretion of a pack of happy bloodhounds"— published hyped-up claims of perceptron-based artificial intelligence. When the expectations were not met, Rosenblatt suffered from the backlash, and the work was to a great extent abandoned. The book by Minsky and Papert [7:70] is widely credited with showing the mathematical limitations of perceptrons and contributing to the decline of the area, but Rosenblatt was already well-aware, judging from his book, of the difficulties with his models and analysis techniques.

During the mid 1970s many authors continued work on models of neural nets, notably Grossberg and his school —see for instance [20:71]— with most of this work attempting to produce differential equation models of various conditioning phenomena. Work by Kohonen and others ([20:99]) on feature extraction and clustering techniques was also prominent during this period.

The resurgence in interest during the mid 1980s can be traced to two independent events. One was the work by Hopfield [20:183] on the design of associative memories, and later the solution of optimization problems, using a special type of recurrent networks. The other was the suggestion to return to Rosenblatt's feedforward nets, but using now differentiable, "sigmoidal," activation functions σ. Differentiability makes it possible to employ steepest descent on weight (parameter) space, in order to find nets that compute a desired function or interpolate at desired values. This was emphasized in the very popular series of books and papers by the "Parallel

Distributed Processing" research group —see for instance [20:150]— and came to be known under the term "backpropagation." (The term comes from the fact that in computing the gradient of an error criterion with respect to parameters, via the chain rule, one multiplies a product such as the one in equation (4) from left to right, "propagating backwards" the vector v that corresponds to the error at the output.)

The interest in Hopfield-type nets for associative storage has waned, as their limitations have been well-explored, theoretically and in applications, while the use of the "backpropagation" approach, sometimes modified in various ways, and extended to include general recurrent neural nets, has taken center stage.

It is hard to explain the popularity of the artificial neural nets from a purely mathematical point of view, given what is *currently* known about them. Some of the claimed advantages were mentioned earlier, but many of the same claims could apply in principle to several other classes of function approximators and systems models. Of course, analogies to biological structure, coupled with a substantial marketing effort by their proponents, did not hurt. Ease of use and geometric interpretability of results helps in user-friendliness, and this has been a major factor. But perhaps the best explanation is the obvious one, namely that their recent re-introduction coincided with the advent of cheap massive computational power. Compared with the mostly linear parameter fitting techniques widely in use, it is conceivable that *any* suitably rich nonlinear technique could do better, provided enough computational effort is spent. In fact, without prior knowledge that one must wait for hundreds of thousands of iterations before noticing any sort of convergence, few would have tried this approach even 20 years ago!

3.3 This Paper

The rest of this paper explores various properties of "backpropagation" networks, with an emphasis on results which are specific to them (as opposed to more general nonlinear models). The reader should not interpret these results in any manner as a justification for their use, nor should it be assumed that this is an exhaustive survey, as it concentrates on those topics which the author has found of interest. There will be a brief discussion of *some* basic theoretical results (potentially) relevant to the use of nets for identification and control. Even for the topics that are covered, most details are not included; references to the literature are given for precise definitions and proofs. There are no explanations of experimental results, solely of theory. Moreover, numerical questions and algorithms are ignored, concentrating instead on ultimate capabilities of nets.

In many papers found in the more theoretical neural nets literature, a theorem or algorithm valid for more or less arbitrary parametric families is quoted, and the contribution is to verify in a more or less straightforward manner conditions of applicability. In this category one often finds work in, for instance, the following topics (which will not be treated here):

Numerical techniques for solving neural net approximation problems. Much work has been done on using conjugate gradient and quasi-Newton algorithms.

Statistical studies. In network learning systems, all the usual issues associated to estimation arise: the tradeoff between variance and bias (or, in AI terms, between generality and generalization). Cross-validation and other techniques are often applied; see for instance the book [40] for much on this topic.

Studies of the effect of incremental (on-line) versus "all at once" (batch or off-line) learning. In actual applications that involve gradient descent, often the complete gradient is not computed at each stage, but an approximation is used, which involves only new data. This is closely related to standard issues treated in the adaptive control and identification literature, and in fact many papers have been written on the use of stochastic approximation results for choosing learning parameters.

Rather than giving a general definition and then treating particular cases, the exposition will be organized in the opposite way: first single-hidden layer nets will be considered, as these have attracted by far the largest portion of research efforts, then two-hidden layer nets, mostly in order to emphasize that such nets are more natural for control applications, and finally recurrent nets, in which feedback is allowed.

4 Feedforward Nets

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be any function, and let m, n, p be positive integers. A *single-hidden layer net with m inputs, p outputs, n hidden units, and activation function σ* is specified by a pair of matrices B, C and a pair of vectors b_0, c_0, where B and C are respectively real matrices of sizes $n \times m$ and $p \times n$, and b_0 and c_0 are respectively real vectors of size n and p. Denote such a net by a 5-tuple

$$\Sigma = \Sigma(B, C, b_0, c_0, \sigma)\,,$$

omitting σ if obvious from the context.

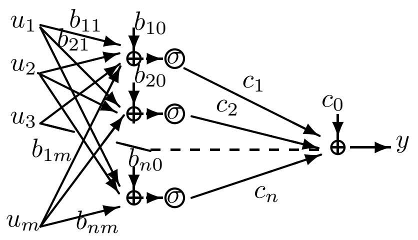
Let $\vec{\sigma}_n : \mathbb{R}^n \to \mathbb{R}^n$ indicate the application of σ to each coordinate of an n-vector: $\vec{\sigma}_n(x_1, \ldots, x_n) = (\sigma(x_1), \ldots, \sigma(x_n))$. (The subscript is omitted as long as its value is clear from the context.) The *behavior* of Σ is defined to be the map

$$\mathrm{beh}_\Sigma \;:\; \mathbb{R}^m \to \mathbb{R}^p \;:\; u \mapsto C\vec{\sigma}(Bu + b_0) + c_0\,. \qquad \text{(1HL)}$$

In other words, the behavior of a network is a composition of the type $f \circ \vec{\sigma} \circ g$, where f and g are affine maps. The name "hidden layer" reflects the fact that the signals in the intermediate space $\mathbb{R}^n$ are not part of inputs or outputs. A function f is said to be *computable by a 1HL net with n hidden units and activation σ* if it is of the type beh_Σ as above. For short, "1HL" will stand from now on for "single hidden layer" when writing about nets or the functions that they compute. Due to lack of space, some notations are unavoidable: $\mathcal{F}^p_{n,\sigma,m}$ will be used for the set of 1HL functions with n hidden units, and $\mathcal{F}^p_{\sigma,m} := \bigcup_{n\geq 0} \mathcal{F}^p_{n,\sigma,m}$. The superscripts are dropped if $p = 1$; note that one can naturally identify $\mathcal{F}^p_{\sigma,m}$ with $(\mathcal{F}_{\sigma,m})^p$.

Often is useful to allow a linear term as well, which can be used for instance to implement local linear feedback control about a desired equilibrium. Such linear terms correspond to direct links from the input to the output node. A function computable by a net *with possible direct input to output connections* (and 1HL) as any function $g : \mathbb{R}^m \to \mathbb{R}^p$ of the form $Fu + f(u)$ where F is linear and $f \in \mathcal{F}^p_{n,\sigma,m}$.

Figure illustrates the interconnection architecture of 1HL nets; the dotted line indicates a possible direct i/o connection. (Here $p = 1$, $b_0 = (b_{01}, \ldots, b_{0n})$, and $C = (c_1, \ldots, c_n)$.)



4.1 Approximation Properties

Much has been written on the topic of function approximation by nets. Hilbert's 13th problem, on realizing functions by superpositions of scalar ones, is sometimes cited in this context. A positive solution of Hilbert's problem was obtained by Kolmogorov, with enhancements by Lorentz and Sprecher. The result implies that any multivariable, real-valued continuous function f can be *exactly* represented, on compacts, in the form $f(x) = \sum_{j=1}^{n} \sigma \left(\sum_{i=1}^{m} \mu_{ij}(u_i)\right)$ where the μ_{ij}'s are universal scalar functions (they depend only on the input dimension, not on f). The number of "hidden units" n can be taken to be $2m + 1$. The function σ, however, depends on the f being represented, and, though continuous, is highly irregular. (Essentially, one is describing a dense curve in the space of continuous functions, parameterized by the scalar function σ.) This result can be used as a basis of *approximation* by nets —with two hidden layers, to be defined later— and using more regular activation functions, but it seems better to start from scratch if one is interested in 1HL theorems.

4.2 Scalar Inputs

Given a function $\sigma : \mathbb{R} \to \mathbb{R}$, write $\mathcal{F}_\sigma$ instead of $\mathcal{F}_{\sigma,1}$; this is the affine span of the set of all dilations and translates of σ. So the elements of $\mathcal{F}_\sigma$ are those functions $\mathbb{R} \to \mathbb{R}$ that can be expressed as finite linear combinations

$$c_0 + \sum_{i=1}^{n} c_i \sigma(B_i u + b_i) \tag{5}$$

(denoting b_i instead of b_{0i} and letting B_i:= ith entry of B).

The mapping σ is a *universal activation* 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. Note that, at this level of generality, nothing is required 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 B_i, b_i, c_i that result in an approximation of f to within a desired tolerance.

Not every nonlinear function is universal in the above sense, of course; for instance, if σ is a fixed 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. A conclusive result has recently been obtained and is as follows; see [22]: Assume that σ is locally Riemann integrable, that is, it is continuous except at most in a set of measure zero, and it is bounded on each compact. Then, σ is a universal activation *if and only if it is not a polynomial*.

Previous results along these lines were obtained in [19], which established that any σ which is continuous, nonconstant, and bounded is universal (see also [13] and [20:(59,85)] for related older results).

The proof in [22] is based essentially on two steps: First, one reduces, by convolution, to the case in which σ is infinitely differentiable (and nonpolynomial). Locally, Taylor series expansions can be used to write σ approximately as a polynomial of arbitrary degree, and enough linearly independent polynomials can be obtained by suitable dilations and translations of σ. Now the Weierstrass Theorem completes the proof. Rather than providing details, it is instructive to briefly sketch the proof of universality in some special but still quite general cases. These special cases are also of interest since they illustrate situations in which the approximations are constructive and give rise to explicit bounds.

A *sigmoidal* function is any function with the property that both $\lim_{u \to -\infty} \sigma(u)$ and $\lim_{u \to +\infty} \sigma(u)$ exist and are distinct (without loss of generality, assume the limits are 0 and 1 respectively). If σ is also monotone, one says that it is a *squashing* function. For squashing σ, the associated "bump function" function $\overline{\sigma}(u) := \sigma(u) - \sigma(u-1)$ is (Riemann) integrable. To show that σ is universal, it is enough to show that $\overline{\sigma}$ is, so from now on assume without loss of generality that σ is integrable, nonnegative, and not identically zero. Let $\hat{\sigma}$ denote the Fourier transform of σ. As σ is nonzero, there is some ω_0 so that $\hat{\sigma}(\omega_0) \neq 0$, and one may assume $\omega_0 \neq 0$ (by continuity). Then, for each t:

$$e^{it}\hat{\sigma}(\omega_0) \;=\; \int_{-\infty}^{\infty} \sigma(x)e^{-i\omega_0 x}e^{it}\,dx \;=\; \frac{1}{|\omega_0|}\int_{-\infty}^{\infty} \sigma\left(\frac{u+t}{\omega_0}\right)e^{-iu}\,du\,.$$

Taking real and imaginary parts, and approximating the integral by Riemann sums, one concludes that sines and cosines can be approximated by elements of $\mathcal{F}_\sigma$. From here, density of trigonometric polynomials provides the desired result. To approximate a given function, one first needs to obtain an approximation by trigonometric polynomials, and then each sine and cosine is approximated by a 1HL net. The speed and accuracy of approximation in this manner is thus determined essentially by Fourier information about both the function to be fit and σ.

Another proof, which allows accuracy to be estimated by the local oscillation of the function being fit, is as follows. Take any continuous function f on $[\alpha,\beta]$, and assume one wants to approximate it to within error $\varepsilon > 0$. First approximate f uniformly, to error $\varepsilon/2$, by a piecewise constant function, i.e. by an element $g \in \mathcal{F}_\mathcal{H}$; this can be done in such a manner that all discontinuous steps be of magnitude less than $\varepsilon/4$. Assume a linear combination with k steps achieves this. Now, if σ is squashing, each term in this sum corresponding to a $c\mathcal{H}(u+b)$ can be approximated by a term of the form $c\sigma(B(u+b))$ for large enough positive B; this can be done to within error $\varepsilon/4k$ uniformly away from the discontinuity and with values everywhere between the limits of the step function. Adding all terms, there results a uniform approximation of g by an element of $\mathcal{F}_\sigma$, to tolerance $\varepsilon/2$, and hence also an ε-approximation of the original function.

The above proof works for any continuous function (in fact, for any "regulated" function: f must have one-sided limits at each point), but the number of terms and/or the coefficients needed may be very large, if f is fast changing. It can be proved, and this will be used below, that if f happens to be a function of bounded variation (for instance, if f has a continuous first derivative), then, letting $V(f)$ denote the total variation of f on the interval $[\alpha,\beta]$, the following holds. For each $\varepsilon > 0$ there exists a sum as in (5), with $\sigma = \mathcal{H}$, which approximates f uniformly to within error ε and so that the sum of the absolute values of the coefficients satisfies $\sum_{i>0}|c_i| \leq V$. The term c_0 can be taken to equal $f(\alpha)$. In fact, the variation V is exactly the smallest possible number so that for each ε there is an approximation with such a coefficient sum. This result holds for any function of bounded variation, even if not continuous, and characterizes the classical bounded variation property. An equivalent statement, after normalizing, is as follows. The variation of f is the smallest $V > 0$ such that $f - f(\alpha)$ is in the closure (in the uniform metric) of the convex hull of $\{\pm V\mathcal{H}(\pm u + b), b \in \mathbb{R}\}$.

4.3 Multivariable Inputs

Universality of σ implies the corresponding multivariable result. That is, for each m, p, and each compact subset K of $\mathbb{R}^m$, the restrictions to K of elements of $\mathcal{F}^p_{\sigma,m}$ (i.e., the functions $F : \mathbb{R}^m \to \mathbb{R}^p$ computable by 1HL) form a dense subset of $C^0(K, \mathbb{R}^p)$. This can be proved as follows.

Working one coordinate at a time, one may assume that $p = 1$. For any $\sigma : \mathbb{R} \to \mathbb{R}$, universal or not, a *$\sigma$-ridge* function $f : \mathbb{R}^m \to \mathbb{R}$ is one of the form $f(u) = \sigma(Bu + b)$, where $B \in R^{1\times m}$ and $b \in \mathbb{R}$. A finite sum $f(u) = \sum_{i=1}^r \sigma_i(B_i u + b_i)$ of functions of this type is a "multiridge" function. Such multiridge functions (not the standard terminology) are used in statistics and pattern classification, in particular when applying projection pursuit techniques, which incrementally build f by choosing the directions B_i, $i = 1, \ldots, r$ in a systematic way so as to minimize an approximation or classification error criterion; see for instance [15]. Note that $\mathcal{F}_{\sigma,m}$ is precisely the set of those multiridge functions for which there are scalars c_i such that $\sigma_i = c_i\sigma$ for all i.

To reduce to the single-input case treated in the previous section, it is enough to show that the set of all multiridge functions (with regular enough σ_i's) is dense, as one can then approximate each ridge term $\sigma_i(B_i u + b_i)$ by simply approximating each scalar function $\sigma_i(x)$ separately and then substituting $x = B_i u + b_i$. Now observe that, by the Weierstrass Theorem, polynomials are dense, so it suffices to show that each monomial in m variables can be written as sum of ridge polynomials. By induction on the number of variables, it is enough to see this for each monomial in *two* variables. Write such a monomial in the following form, with $0 < r < d$: $u^{d-r}v^r$. The claim is now that this can be written as: $\sum_{i=0}^d c_i(a_i u + v)^d$ for any fixed choice of distinct nonzero $a_0, \ldots, a_d$ (for instance, $a_0 = 1, a_1 = 2, \ldots, a_d = d+1$) and suitable c_i's. Dividing by u^d and letting $z := v/u$ it is equivalent to solve $\sum_{i=0}^d c_i(a_i + z)^d = z^r$ for the c_i's. As the polynomials $(a_i + z)^d$ are linearly independent —computing derivatives of order $0, \ldots, d$ at $z = 0$ results in a Vandermonde matrix,— they must span the set of all polynomials of degree $\leq d$, and in particular z^r is in the span, as desired.

Instead of polynomials, one can also base the reduction proof on Fourier expansions. For this, it is enough to see that trigonometric polynomials satisfy the conditions of the Stone-Weierstrass Theorem. Other proofs use various algorithms for reconstruction from projections, such as Radon transforms (see [11]). In conclusion, universality guarantees that functions of the type (5) are dense in C^0. One can also establish density results for L^q

spaces, $q < \infty$, on compact sets, simply using for those spaces the density of continuous functions. Approximation results on noncompact spaces, in L^q but for finite measures, are also possible but considerably harder; see [19].

It is false, on the other hand, that one can do uniform approximation of more or less arbitrary discontinuous functions, that is, approximation in L^∞, even on compact sets. In input dimension 1, the approximated function must be regulated (see above), which is not too strong an assumption. But in higher input dimensions, there are strong constraints. For instance, the characteristic function of a unit square in the plane cannot be approximated by 1HL nets to within small error. This leads to very interesting questions for control problems, where it is often the case that the only solution to a problem is one that involves such an approximation. Later the paper deals with such issues.

4.4 Number of Units

It is interesting to ask how many units are needed in order to solve a given classification or interpolation task. This can be formalized as follows. Let $\mathcal{U}$ be a set, to be called the *input set*, and let $\mathcal{Y}$ be another set, the *output set*. In the results below, for 1HL nets, $\mathcal{U} = \mathbb{R}^m$, but the definitions are more general. To measure discrepancy in outputs, it may be assumed that $\mathcal{Y}$ is a metric space. For simplicity, assume from now on that $\mathcal{Y} = \mathbb{R}$, or $\mathcal{Y}$ is the subset $\{0, 1\}$, if binary data is of interest. A *labeled sample* is a finite set $S = \{(u_1, y_1), \ldots, (u_s, y_s)\}$, where $u_1, \ldots, u_s \in \mathcal{U}$ and $y_1, \ldots, y_s \in \mathcal{Y}$. (The y_i's are the "labels;" they are *binary* if $y_i \in \{0, 1\}$.) It is assumed that the sample is consistent, that is, $u_i = u_j$ implies $y_i = y_j$. A *classifier* is a function $F : \mathcal{U} \to \mathcal{Y}$. 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 $\mathcal{F}$ of classifiers will be called an *architecture*. Typically, and below, $\mathcal{F} = \{F_{\vec{w}} : \mathcal{U} \to \mathcal{Y}, \vec{w} \in R^r\}$ is a set of functions parameterized by $\vec{w} \in R^r$, where $r = r(\mathcal{F})$. An example is that of nets with m=1 inputs, p =1 outputs, and n hidden units, that is, $\mathcal{F}^1_{n,\sigma,1}$; here the parameter set has dimension r=$3n + 1$.

The sample S is *loadable into* $\mathcal{F}$ iff $\inf_{f \in \mathcal{F}} E(F, S) = 0$. Note that for a binary sample S and a binary classifier F, $E(F, S)$ just counts the number of missclassifications, so in the binary case loadability corresponds to being able to obtain exactly the values y_i by suitable choice of $f \in \mathcal{F}$. For continuous-valued y_i's, loadability means that values arbitrarily close to the desired y_i's can be obtained.

One may define the *capacity* $c(\mathcal{F})$ of $\mathcal{F}$ via the requirement that:

$$c(\mathcal{F}) \geq \kappa \text{ iff every } S \text{ of cardinality } \kappa \text{ is loadable.}$$

(Other natural definitions of capacity measures are possible, in particular the VC dimension mentioned below.) That is, $c(\mathcal{F}) = \infty$ means that all finite S are loadable, and $c(\mathcal{F}) = \kappa < \infty$ means that each S of cardinality $\leq \kappa$ is loadable but some S of cardinality $\kappa + 1$ is not.

Various relations between capacity and number of neurons are known for nets with one hidden layer and Heaviside or sigmoidal activations. It is an easy exercise to show that the results are independent of the input dimension m, for any fixed activation type σ and fixed number of units n. (Sketch of proof: the case $m = 1$ is included in the general case, by simply taking collinear points u_i. Conversely, given any set of points u_i in $\mathbb{R}^m$, there is always some vector v whose inner products with the distinct u_i's are all different, and this reduces everything to the one dimensional case.) In the case $m = 1$, parameter counts are interesting, so that case will be considered next. Observe that, for 1HL nets with one input, and n hidden units (and p=1), there are $3n + 1$ parameters (appearing nonlinearly) —or $3n + 2$ if allowing direct connections, that is, when there is an extra linear term $c_{n+1}u$ in (5)— though for $\mathcal{H}$, effectively only $2n + 1$ matter. (In the case of the standard sigmoid, a Jacobian computation shows that these parameters are independent.) For classification purposes, it is routine to consider just the sign of the output of a neural net, and to classify an input according to this sign. Thus one introduces the class $\mathcal{H}(\mathcal{F}^1_{n,\sigma,1})$ consisting of all $\{0, 1\}$-valued functions of the form $\mathcal{H}(f(u))$ with $f \in \mathcal{F}^1_{n,\sigma,1}$.

Of interest are scaling properties as $n \to \infty$. Let $\text{CLSF}(\sigma) := \underline{\lim}_{\,n\to\infty} c(\mathcal{F})/r(\mathcal{F})$ for $\mathcal{F} = \mathcal{H}(\mathcal{F}^1_{n,\sigma,1})$ and $\text{INTP}(\sigma) := \underline{\lim}_{\,n\to\infty} c(\mathcal{F})/r(\mathcal{F})$ for $\mathcal{F} = \mathcal{F}^1_{n,\sigma,1}$. Define similarly $\text{CLSF}^{\text{d}}, \text{INTP}^{\text{d}}$ when using 1HL nets with direct connections.

Consider the property (*): $\exists\, c$ s.t. σ is differentiable at c and $\sigma'(c) \neq 0$. Then, for classification, the following results are given in [34]:

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

assuming that σ is sigmoidal and (*) holds. The last bound is best possible, in the sense that for the piecewise linear π one has $\text{CLSF}(\pi) = 2/3$ while it is the case that $\text{CLSF}(\sigma) = \infty$ for some "nice" (even, real-analytic) sigmoidal functions σ satisfying (*). Regarding continuous-valued interpolation, these are the results:

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

assuming in the last case that (*) holds and σ is a continuous sigmoidal. Again here, $\text{INTP}(\pi) = 2/3$, and one can also show that $2/3 \leq \text{INTP}(\sigma_{\text{s}}) \leq 1$ 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; see also [24]). Furthermore, the inequality $\text{INTP}(\sigma) \geq 1/3$ holds for any universal nonlinearity. Obtaining the precise value for σ_{s} is a very interesting open problem.

Note that the above discussion focused on general bounds on what can be achieved. In practice, however, one is given a particular labeled sample and the problem is to decide whether this sample can be loaded into the desired architecture. For differentiable activations, numerical techniques are used to estimate the answer. But one may ask about the abstract (in the sense of computer science) *computational complexity* of the loading problem: do there exist weights that satisfy the desired objective? For fixed input dimension and Heaviside activations, this becomes essentially a linear programming problem, but even for such activations, the problem is NP-hard when scaling with respect to the number of input or output dimensions; see [9] and [21] for much on this issue.

4.5 Learnability and VC Dimension

One of the main current approaches to defining and understanding the question of learning, which after all underlies much of the reason for the use of neural nets in control, is based on the *probably approximately correct* ("PAC") model proposed in computational learning theory by Valiant in the early 1980s. Very closely related ideas appeared in statistics even earlier, in the work of Vapnik and Chervonenkis —see the excellent book [38]— and the interactions between statistics and computer science are the subject of much current research. The next few paragraphs introduce the basic ideas, using terminology from learning theory; for more details see for instance the textbook [3].

In the PAC paradigm, a "learner" has access to data given by a labeled sample $S = (u_1, y_1), \ldots, (u_s, y_s)$. The inputs u_i are being generated at random, independently and identically distributed according to some probability measure P. There is some fixed but unknown function f so that, for each i, $y_i = f(u_i)$, and f belongs to some known class of functions $\mathcal{F}$. This class of functions is used to characterize the assumptions ("bias") being made about what is common among the observed input/output pairs. The learner knows the class $\mathcal{F}$ but not the particular f being used. For instance, in systems identification, the u_i's might correspond to inputs applied to a plant and the y_i's would be the corresponding outputs, while $\mathcal{F}$ might consist of all stable SISO systems of a certain degree. The learner's objective is to use the information gathered from the observed labeled sample in order to guess the correct f in $\mathcal{F}$. In a control environment, a learning controller might be trained in this manner to recognize certain features of the state space which are associated to a particular control action.

For simplicity, because the theory is far simpler in that case, and because of the application to pattern classification, only binary-valued functions

f (and hence binary samples $y_i \in \{0,1\}$) will be considered here. Thus instead of a class of functions $\mathcal{F}$ one could work equivalently with a class of *concepts*, that is, those subsets of the input space which are of the form $\{u \mid f(u) = 1\}$ for the various $f \in \mathcal{F}$. There are many generalizations of this basic setup, including dealing with noisy data, continuous-valued outputs, or allowing the learner to guess a function not in the original class $\mathcal{F}$, but the basic ideas are best illustrated in this simplest case.

One defines the *error* of a hypothesis $\hat{f}$ made by the learner as the probability that it will incorrectly classify a new randomly chosen example (u, y), that is, the probability that $\hat{f}(u) \neq y$ (the prediction error). It is assumed that u is picked with the same probability distribution P that was used to generate the training inputs u_i. Since only a limited number of samples are presented during training, they will in general not be sufficient to distinguish between all possible concepts (possible functions f), and this is a source of error. Another possible source of error arises from the fact that the inputs u, having been randomly chosen, might not be representative enough of future inputs. Neither of these errors need be as serious as it may seem at first sight, however. First of all, if the sample is large enough, and $\hat{f}(u_i) = y_i$ for all i, it is quite unlikely that $f \neq \hat{f}$ unless the class is too rich. It may happen, but with small probability. For the second type of error, if the probability distribution P being used to generate the inputs u_i is concentrated in a part of the input space where f and $\hat{f}$ coincide, then the new testing input u will likely come from this part of the space as well, and hence the prediction error will again be small. The term PAC learning refers to the fact that (very) "probably" the estimate will be "approximately correct."

Before giving precise definitions of learnability in the sense just discussed, it is instructive to consider very informally a case which does *not* lead to learnability and one that does. Assume that the inputs u are real numbers uniformly distributed in the interval $[0,1]$ and that the set of functions $\mathcal{F}$ consists of the functions $f_k(u) := \mathcal{H}(\sin(ku))$, over all positive integers k. That is, $f_k(u)$ is 1 if $\sin(ku) > 0$ and zero otherwise. Now, given a random sequence $(u_1, f(u_1)), \ldots, (u_s, f(u_s))$, where $f = f_k$, there is in general no possible good prediction of $f(u)$ for a new u. Indeed, with probability one, the complete set $u_1, \ldots, u_s, u$ will consist of rationally independent real numbers, and therefore there exists some $j \neq k$ so that $f_j(u_i) = f(u_i)$ for $i = 1, \ldots, s$ but $f_j(u) \neq f(u)$. (Recall that the set of values of $(\sin(lu_1), \ldots, \sin(lu_s), \sin(lu))$, as l ranges over the positive integers, is dense in $[-1,1]^{s+1}$.) Since the learner is not able to decide, on the basis of the observed data, if $f = f_k$ or $f = f_j$, the prediction $\hat{f}(u)$ cannot be made with any degree of reliability. Contrast this example with the following one, at the other extreme: the functions $\mathcal{F}$ are now of the type $f_a(x) := \mathcal{H}(x - a)$, with $a \in [0,1]$. The concepts to be identified are the sets of the form $\{x > a\}$. Identifying the concept means identifying the cut point a. Given a large enough ($s \gg 1$) sample, there will be enough pairs (u_i, y_i) with u_i near a so that a good estimate of a can be obtained, for instance by estimating a as the midpoint of the interval $[a_1, a_2]$, where a_1 is the largest observed u_i so that $f(u_i) = 0$ and a_2 is the smallest one so that $f(u_i) = 1$. The only errors would be due to a bad sample (it so happened that all the u_i's were far from a, so the interval is very large) or to the test input u falling in the interval $[a_1, a_2]$, where no information about a is available. But these errors occur with low probability.

The definitions are as follows (they are standard, but the terminology is changed a bit in order to make it closer to systems and control usage). An input space $\mathcal{U}$ as well as a collection $\mathcal{F}$ of maps $\mathcal{U} \to \{0, 1\}$ are given. The set $\mathcal{U}$ is assumed to be countable, or an Euclidean space, and the maps in $\mathcal{F}$ are assumed to be Borel measurable. In addition, mild regularity assumptions are made which insure that all sets appearing below are measurable (details are omitted; see the references). Let W be the set of all sequences

$$w = (u_1, f(u_1)), \ldots, (u_s, f(u_s)) \tag{6}$$

over all $s \geq 1$, $(u_1, \ldots, u_s) \in \mathcal{U}^s$, and $f \in \mathcal{F}$. An *identifier* is a map $\varphi : W \to \mathcal{F}$. The value of φ on the sequence appearing in (6) will be denoted as φ_w. The *error* of φ with respect to a probability measure P on $\mathcal{U}$, an $f \in \mathcal{F}$, and a sequence $(u_1, \ldots, u_s) \in \mathcal{U}^s$, is $\mathrm{Err}\,(P, f, u_1, \ldots, u_s) := \mathrm{Prob}\,[\varphi_w(u) \neq f(u)]$ (where the probability is being understood with respect to P).

The class $\mathcal{F}$ is (uniformly) *learnable* if there is some identifier φ with the following property: For each $\varepsilon, \delta > 0$ there is some s so that, for every probability P and every $f \in \mathcal{F}$, $\mathrm{Prob}\,[\mathrm{Err}\,(P, f, u_1, \ldots, u_s) > \varepsilon] < \delta$ (where the probability is being understood with respect to P^s on $\mathcal{U}^s$).

In the learnable case, the function $s(\varepsilon, \delta)$ which provides, for any given ε and δ, the smallest possible s as above, is called the *sample complexity* of the class $\mathcal{F}$. It can be proved that $s(\varepsilon, \delta)$ is automatically bounded by a polynomial in $1/\varepsilon$ and $1/\delta$. In fact, it can be bounded by $-(c/\varepsilon) \log(\delta\varepsilon)$, where c is a constant that depends only on the class $\mathcal{F}$. It can also be proved that, if there is any identifier at all in the above sense, then one can always use the following naive identification procedure: pick any element f which is consistent with the observed data. This leads computationally to the loading question discussed earlier. In the statistics literature —see [38]— this "naive technique" is a particular case of what is called *empirical risk minimization*.

The above discussion corresponds to being able to learn uniformly with respect to unknown input distributions. Much research is currently taking place on the question of learning with respect to particular subfamilies of distributions on $\mathcal{U}$. It is perfectly possible for an $\mathcal{F}$ not to be learnable in the above sense but to be learnable if the inputs are assumed to be Gaussian, for instance. Notions of Kolmogorov metric entropy are used to characterize such learning. (In the case of learning with respect to particular data distributions, the naive identification procedure is no longer guaranteed to work, and more sophisticated methods must be used.) Another variation consists of studying the computational effort required in order to use the identifier φ; the definition given above is purely information-theoretic, but the original work of Valiant emphasized that aspect, essentially the loading problem, as well.

Checking learnability is in principle a difficult issue, and the introduction of the following combinatorial concept is extremely useful in that regard. A subset $S = \{u_1, \ldots, u_s\}$ of $\mathcal{U}$ is said to be *shattered* by the class of binary functions $\mathcal{F}$ if all possible binary labeled samples $\{(u_1, y_1), \ldots, (u_s, y_s)\}$ (all $y_i \in \{0, 1\}$) are loadable into $\mathcal{F}$. The *Vapnik-Chervonenkis dimension* $\textsc{vc}\,(\mathcal{F})$ is the supremum (possibly infinite) of the set of integers κ for which there is *some* set S of cardinality κ that can be shattered by $\mathcal{F}$. (Thus, $\textsc{vc}\,(\mathcal{F})$ is at least as large as the capacity $c(\mathcal{F})$ defined earlier.)

The main result, due to [10], but closely related to previous results in statistics (see [38]) is: *The class $\mathcal{F}$ is learnable if and only if* $\textsc{vc}\,(\mathcal{F}) < \infty$. So the VC dimension completely characterizes learnability with respect to unknown distributions; in fact, the constant "c" in the sample complexity bound given earlier is a simple function of the number $\textsc{vc}\,(\mathcal{F})$, independent of $\mathcal{F}$ itself.

It follows from the results in [8] that, for the class $\mathcal{H}(\mathcal{F}_{n,\sigma,m})$ of classifiers implementable by 1HL nets with with n hidden units and m inputs (n and m fixed) and activation $\sigma = \mathcal{H}$, $\textsc{vc}\,(\mathcal{F}) < dmn(1 + \log(n))$, for a small constant d. Thus 1HL nets with threshold activations are learnable in the above sense. Note that finiteness is trivial to verify in one very special case, namely $m = 1$: in this case, a function f computable by a 1HL $\mathcal{H}$-net with n hidden units is piecewise constant with at most n discontinuities. This means that, given any set with $n+2$ elements or more, the alternate labeling $0, 1, 0, 1, \ldots$ cannot be implemented; in other words, $\textsc{vc}\,(\mathcal{F})$ is at most $n + 1$.

It is easy to construct examples of sigmoids, even extremely well-behaved ones (analytic and squashing, for instance) for which the VC dimension of the class $\mathcal{H}(\mathcal{F}_{n,\sigma,m})$ is infinite; see [34]. However, for the standard sigmoid, a recent result in [24] proves that $\textsc{vc}\,(\mathcal{F}) < \infty$, so neural nets with activation σ_{s} are also learnable. Thus sigmoidal nets appear to have some special properties, vis a vis other possible more general parametric classes of functions, at least from a learnability viewpoint. Other results on finiteness of learning, but from a more statistical viewpoint (nonlinear regression, estimation of joint densities), are given in [17], where metric entropy estimates are obtained for networks with bounded weights.

The results on learnability just explained extend to other classes of feedforward nets, including 2HL nets (defined below) and nets involving products of inputs ("high order nets"), but such more general results will not be reviewed here. Note also that the general question of guessing values of a function at unseen inputs is essentially also that of extrapolation, and it is classical in numerical analysis. In that context, the "prior" class of functions $\mathcal{F}$ often reflects a smoothness constraint.

4.6 Quality of Approximation

Certain recent results due to Andrew Barron and Lee Jones have been used to support the claim that 1HL neural network approximations may require less parameters than conventional techniques. What is meant by this is that, given a function f to be approximated (for instance, a pattern classifier or a controller), approximations of f to within a desired error tolerance ε can be obtained using "small" networks, while using, for instance, orthogonal polynomials, splines, or Fourier series, would require an astronomical number of terms, especially for multivariate inputs.

At least as the results currently stand, this claim represents a misunderstanding of the very nice contributions of Barron and Jones. First of all, their results would, in general terms, apply equally well to show that one can obtain efficient approximations with various types of classical basis functions, *as long as the basis elements can be chosen in a nonlinear fashion*, just as with neural networks. For instance, splines with varying (rather than fixed) nodes, or trigonometric series with adaptively selected frequencies, will have the same properties. What is important is the possibility of *selecting* terms adaptively, in contrast to the use of a large basis containing many terms and fitting these through the use of least squares. Thus, the important fact about the recent results is that they emphasize that *non*linear parameterizations may require less parameters than linear ones to achieve a guaranteed degree of approximation. (Abstractly, this is not so surprising: for an analogy, consider the fact that a *one*-parameter analytic curve, based on ergodic motions, can be used to approximate arbitrarily well every element in an Euclidean space $\mathbb{R}^d$, but no $(d-1)$-dimensional subspace can do so.) The most interesting —and as yet unresolved— issues have to do with the tradeoffs between rate of approximation and number of parameters, and the related balance between generalization capabilities and computational complexity.

The rest of this section reviews some of the basic results in question. The main tool is the following Lemma, which is often attributed to Maurey (see [29]): *Let G be a bounded subset of a Hilbert space H, with $\|g\| \leq \gamma$ for all $g \in G$, and let G_n be the subset of H consisting of all convex combinations of at most n elements of G. Then, for each f in the closed convex hull of G, and for each $n \geq 1$,*

$$\inf_{g\in G_n} \|f-g\| \leq \frac{\gamma}{\sqrt{n}} \,.$$

There are several different proofs, probabilistic and geometric, of this result and related ones in which the constant in the bound may be allowed to be slightly larger. Jones pointed out that one may use an incremental approximation, monotonically decreasing the approximation error while recursively adding one element of G at a time.

In applications, the set H is a space of square integrable functions and, for some fixed $\Gamma > 0$, $G = G(\Gamma)$ is the set of ridge functions $c\sigma(Bu + b)$, with $|c| \leq \Gamma$, $B \in \mathbb{R}^{1\times m}$, and $b \in \mathbb{R}$. In that case, $G(\Gamma)_n$ is the set of 1HL σ-nets with n units and weight sums $\sum |c_i| \leq \Gamma$. The approximation result says that those functions f which can be approximated arbitrarily close by elements of the $G(\Gamma)_n$'s (fixed Γ), can in fact be also approximated with a mean square error $O(1/n)$ using 1HL nets with no more than n units. To apply all of this, one must understand what functions f are of this form. One such example was discussed earlier, namely the case of scalar functions of bounded variation on an interval $[\alpha, \beta]$. In that case, it was remarked that f-$f(\alpha)$ is in the closed convex hull of $\{\pm V\mathcal{H}(\pm u - b)\}$ (in the uniform norm, and hence also in L^2 for any finite measure on $[\alpha, \beta]$). Generalizing, Barron in [6] suggested *defining* a function $f : Q \to \mathbb{R}$, where Q is a bounded subset of $\mathbb{R}^m$, to have "bounded variation with respect to halfspaces" if this property holds: there exists a real number V so that, for some $q \in Q$, f-$f(q)$ is in the closed convex hull, in the uniform norm, of the functions $\pm V\mathcal{H}(Bu + b)$. The smallest such V he called the variation $V_{f,Q}$ of the function f on the set Q. For functions with $V_{f,Q} < \infty$, there is then a rate of approximation theorem using 1HL $\mathcal{H}$-nets, and from there also one for sigmoidal nets (just approximate each Heaviside function by a sigmoid). The conclusion is that one can approximate by 1HL nets with n units with an error as small as $V_{f,Q}/\sqrt{n}$ in $L^2(Q, \mu)$, for any probability measure μ on Q.

One source of examples of functions of bounded variation in this generalized sense is as follows. Let Q for simplicity be taken to be the unit ball. Assume that f admits a Fourier representation $f(u) = \int_{\mathbb{R}^m} e^{i\omega . u}\tilde{f}(\omega)d\omega$ for each $u \in Q$, and that $C_f = \int_{\mathbb{R}^m} \|\omega\||\tilde{f}(\omega)|d\omega < \infty$. Then, $V_{f,Q} \leq 2C_f$. (Norm of ω is standard Euclidean norm.) For the spaces $\{f \,|\, C_f \leq k\}$, k a fixed constant, Barron also proved (see references in [6]) that approximations by linear subspaces of dimension n would result in a worst-case error of at least $O(n^{-1/d})$, which is asymptotically worse than $O(1/\sqrt{n})$ when $d > 2$.

Note that the approximations, as they use Maurey-type arguments, hold a priori only in L^2 (or other Hilbert spaces). Indeed, these arguments are false for approximation in L^∞, if the set G is arbitrary (see [14] for this and related remarks). However, Barron in the above reference was also able to prove a similar result for the particular case in which G corresponds to characteristic functions of half-spaces, using a deeper result due to Dudley. This beautiful recent contribution is closer to being a true "neural networks" theorem, as it uses essentially a property involving VC dimension which is not true in general.

4.7 Uniqueness

As discussed, in most applications dealing with learning and pattern recognition, neural nets are employed as models whose parameters must be fit to training data characterized by a labeled sample. Gradient descent and other algorithms are used in order to minimize $E(F, S)$ over all F corresponding to a fixed network architecture. Among the many numerical complications that arise when following such a procedure are the possibilities of (1) non-global local minima, and (2) multiple global minimizers. The first issue was dealt with by many different authors —see for instance [36] and the references there— and will not be reviewed here. Regarding the second point, observe that there are obvious transformations that leave the behavior of a network invariant, such as interchanges of all incoming and outgoing weights between two neurons (mathematically, the relabeling of neurons) and, for odd activation functions, flipping the signs of all incoming and outgoing weights at any given node. Two networks differing in such a manner give the same error on the training data. When there is a net that fits perfectly the data, all nets differing from it by one of the above transformations also attain the global minimum (zero) of the error functional.

It is then natural to ask if neuron exchanges and sign flips are the *only* function-preserving transformations that can generically occur. If these are the only possible ones, then essentially all the internal structure is uniquely determined by the external behavior of the network. Moreover, the set of invariant transformations is then finite, and there is no possible dimensionality reduction in the parameter space. Such a situation is in sharp contrast to classical linear systems, where canonical forms have to be introduced in order to achieve parameter identifiability. (Seen more positively, the parameterizations provided by neural networks are then irredundant.)

For simplicity, assume from now on that $p = 1$; generalizations to the multiple-output case are not hard but they complicate notations. Also, assume that σ is an odd function. Thus, from now on, C in the definition of 1HL net is a row n-vector and c_0 is a constant. Two networks Σ and $\hat{\Sigma}$ are (input/output) *equivalent*, denoted $\Sigma \sim \hat{\Sigma}$, if $\text{beh}_\Sigma = \text{beh}_{\hat{\Sigma}}$ (equality of functions). The question to be studied, then, is: to what extent does $\Sigma \sim \hat{\Sigma}$ imply $\Sigma = \hat{\Sigma}$?

The function σ satisfies the *independence property* ("IP" from now on) if, for every positive integer l, nonzero real numbers $b_1, \ldots, b_l$, and real numbers $b_{01}, \ldots, b_{0l}$ for which the pairs (b_i, b_{0i}), $i = 1, \ldots, l$ satisfy $(b_i, b_{0i}) \neq \pm(b_j, b_{0j})$ $\forall i \neq j$, it must hold that the functions $1, \sigma(b_1 u + b_{01}), \ldots, \sigma(b_l u + b_{0l})$ are linearly independent. The function σ satisfies the *weak* independence property ("WIP") if the above linear independence property is only required to hold for all pairs with $b_{0i} = 0$, $i = 1, \ldots, l$.

Let $\Sigma(B, C, b_0, c_0, \sigma)$ be given, and denote by B_i the transpose of the ith row of the matrix B and by c_i and b_{0i} the ith entries of C and b_0 respectively. With these notations, $\mathrm{beh}_\Sigma(u) = c_0 + \sum_{i=1}^n c_i \sigma(B_i u + b_{0i})$. As in [37], Σ is called *irreducible* if the following properties hold: $c_i \neq 0$ for each $i = 1, \ldots, n$; $B_i \neq 0$ for each $i = 1, \ldots, n$; and $(B_i, b_{0i}) \neq \pm(B_j, b_{0j})$ for all $i \neq j$. Given $\Sigma(B, C, b_0, c_0, \sigma)$, a *sign-flip* operation consists of simultaneously reversing the signs of c_i, B_i, and b_{0i}, for some i. A *node-permutation* consists of interchanging (c_i, B_i, b_{0i}) with (c_j, B_j, b_{0j}), for some i, j. Two nets Σ and $\hat{\Sigma}$ are *sign-permutation (sp) equivalent* if $n = \hat{n}$ and (B, C, b_0, c_0) can be transformed into $(\hat{B}, \hat{C}, \hat{b_0}, \hat{c}_0)$ by means of a finite number of sign-flips and node-permutations. Of course, sp-equivalent nets have the same behavior (since σ has been assumed to be odd). With this terminology, the following holds: *Let σ be odd and satisfy property* IP. *Assume that Σ and $\hat{\Sigma}$ are both irreducible, and $\Sigma \sim \hat{\Sigma}$. Then, Σ and $\hat{\Sigma}$ are sp-equivalent.* A net Σ has *no offsets* if $b_0 = c_0 = 0$ (the terminology "biases" or "thresholds" is sometimes used instead of offsets, but these terms are used for other very different purposes as well). Then, also: *If σ is odd and satisfies* WIP*, the same statement is true for nets with no offsets.*

Characterizing WIP is especially easy, and very classical: *If σ is a polynomial,* WIP *does not hold. Conversely, if σ is odd and infinitely differentiable, and if there are an infinite number of nonzero derivatives $\sigma^{(k)}(0)$, then σ satisfies property* IP. Nets with no offsets appear naturally in signal processing and control applications, as there it is often the case that one requires that $\mathrm{beh}(0) = 0$, that is, the zero input signal causes no effect, corresponding to equilibrium initial states for a controller or filter. Results in this case are closely related to work in the 1970s by Rugh and coworkers and by Boyd and Chua in the early 1980s.

It appears to be harder to obtain elegant characterizations of the stronger property IP. For obvious examples of functions not satisfying IP, take $\sigma(x) = e^x$, any periodic function, or any polynomial. However, the most interesting activation for neural network applications is $\sigma(x) = \tanh(x)$, or equivalently after a linear transformation, the standard sigmoid $\frac{1}{1+e^{-x}}$. In this case, Sussmann showed in [37] that the IP property, and hence the desired uniqueness statement, hold. His proof was based on explicit computations for the particular function $\tanh(x)$. An alternative proof is possible, using analytic continuations and residues, and allows a more general result to be established (see [2] for details): *Assume that σ is a real-analytic function, and it extends to an analytic function $\sigma : \mathbb{C} \to \mathbb{C}$ defined on a subset $D \subseteq \mathbb{C}$ of the form:*

$$D = \{z \mid |\mathrm{Im}\, z| \leq \lambda\} \setminus \{z_0, \bar{z}_0\}$$

for some $\lambda > 0$, where $\operatorname{Im} z_0 = \lambda$ and z_0 and $\bar{z}_0$ are singularities, that is, there is a sequence $z_n \to z_0$ so that $|\sigma(z_n)| \to \infty$, and similarly for $\bar{z}_0$. Then, σ satisfies property IP.

4.8 Two-Hidden Layers and Nonlinear Feedback

The previous sections dealt with 1HL nets. Next, the case of two hidden layers is treated. For the topics treated here, there is no need to define nets themselves, but just their behavior. If $\sigma : \mathbb{R} \to \mathbb{R}$ is any function, and m, p are positive integers, a function computed by a two-hidden layer ("2HL") net with m inputs and p outputs and activation function σ is by definition one of the type $f \circ \vec{\sigma}_n \circ g \circ \vec{\sigma}_l \circ h$, where f, g, and h are affine maps ($n + l$ is then called the number of "hidden units"). As earlier, a function computable by a 2HL net with direct input/output connections is one of the form $Fu + f(u)$, where F is linear and f is computable by a 2HL net.

One hidden layer networks have universal approximation properties, and rates of convergence can be estimated. However, these rates may not be as good as those achievable with 2HL nets. For instance, functions that are piecewise constant on squares approximate certain classes of functions in $\mathbb{R}^2$ very efficiently, and while it is not difficult to express them using 2HL nets, often no such 1HL net expression is possible.

Another disadvantage of 1HL vis a vis 2HL nets arises from the topologies in which the 1HL approximation theorems hold. This has serious implications in control applications, and can be illustrated with the following idea. Suppose that there is some *discontinuous* feedback law $u = k(x)$ which globally asymptotically stabilizes the planar system $\dot{x} = f(x, u)$ with respect to the origin. Assume that k is perfectly known but one wishes to restrict possible controllers to those computable by 1HL nets. It would appear that this would be easy to achieve, as one may merely approximate the given k by a 1HL function $\hat{k}$ and then use this $\hat{k}$ as the controller. The problem is that, for general discontinuous functions, the results only insure approximation in L^p norm (p finite), but it is impossible in general to approximate k *uniformly*. (Uniform approximation of functions that are continuous, or that are of "bounded variation with respect to half spaces," is possible, however, so when there is a continuous feedback k that stabilizes, this type of obstruction dissappears.) A weak type of approximation may not be enough for control purposes. For instance, it may be the case that for each approximant $\hat{k}$ there is some simple closed curve Γ encircling the origin where the approximation is bad (a set of measure zero!) and that this causes the vector field $f(x, \hat{k}(x))$ to point outward everywhere on Γ; in that case the closed loop behavior cannot be globally asymptotically stable, as trajectories cannot cross Γ.

It is possible to construct examples of systems which are otherwise stabilizable but such that every possible feedback implementable by a 1HL net (with basically any type of activation, continuous or not) must give rise to a nontrivial periodic orbit. On the other hand, it can be shown that every system that is stabilizable, by whatever k, can also be stabilized using 2HL nets with discontinuous activations (under mild technical conditions, and using sampled control). See [35] for details.

To summarize, if stabilization requires discontinuities in feedback laws, it may be the case that no possible 1HL net stabilizes. Thus the issue of stabilization by nets is closely related to the standard problem of continuous and smooth stabilization of nonlinear systems, one that has attracted much research attention in recent years. Roughly, there is a hierarchy of state-feedback stabilization problems: those that admit continuous solutions, those that don't but can still be solved using 1HL nets with discontinuous activations, and more general ones (solvable with 2HL). It can be expected that an analogous situation will be true for other control problems. The reason that most neurocontrol papers have used 1HL nets is that they almost always dealt with feedback linearizable systems, which admit continuous stabilizers.

Some Details.

Before discussing stabilization, one can understand the necessity of 2HL nets by means of a more abstract type of question. Consider 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 nets with continuous σ cannot be used. However, and this is the interesting part, [35] 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 activation 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.

Consider now 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 smooth stabilizer k is essentially equivalent to the possibility of stabilizing using 1HL nets (with smooth 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.)

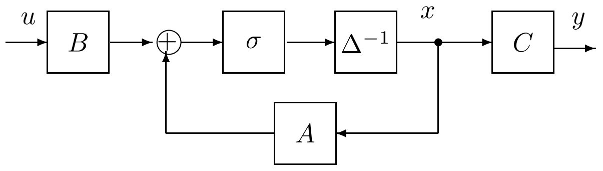
More precisely, assume that f is twice continuously differentiable, that k is also in C^2, that the origin is an exponentially stable point for $\dot{x}{=}f(x,k(x))$, and that K is a compact subset of the domain of stability. Pick any σ which has the property that twice continuously differentiable functions can be approximated uniformly, together with their derivatives, using 1HL nets (most interesting twice-differentiable scalar nonlinearities will do; see [19]). Then, one can conclude that there is also a different k, this one a 1HL net with activation 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 [33] 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 (this last part uses the fact that σ approximates in $C^2(K)$). Now continuity of solutions on the right-hand side gives the result globally on K.)

In general, smooth (or even continuous) stabilizers fail to exist, as discussed for instance in [33], Section 4.8 and references there. Thus 1HLN feedback laws, with continuous σ, do not provide a rich enough class of controllers. This motivates the search for discontinuous feedback. 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, [35] 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. On the other hand, an example is given there of a system satisfying the asymptotic controllability condition but for which every possible 1HL stabilizer gives rise to a nontrivial periodic orbit.

5 Recurrent Nets

A *recurrent net (or σ-system) with m inputs, p outputs, dimension n, and activation function σ* is specified by a triple of matrices A, B, C where A, B, and C are respectively real matrices of sizes $n \times n$, $n \times m$ and $p \times n$. Use the notation $\Sigma = \Sigma(A, B, C, \sigma)$, omitting σ if obvious from the context. One interprets the above data (A, B, C) as defining a controlled and observed dynamical system evolving in $\mathbb{R}^n$ (in the standard sense of control theory;

see e.g. [33]) by means of a differential equation $\dot{x} = \vec{\sigma}\,(Ax + Bu)\,,\; y = Cx$ in continuous-time (dot indicates time derivative), or a difference equation $x^+ = \vec{\sigma}\,(Ax + Bu)\,,\; y = Cx$ in discrete-time ("+" indicates a unit time shift). See the block diagram in the next Figure, where $\Delta = x^+$ or $= \dot{x}$ in discrete or continuous time respectively.



Other systems models are possible; for instance, "Hopfield nets" have dynamics of the form $\dot{x} = -Dx + \vec{\sigma}\,(Ax + Bu)$ (with D a diagonal matrix and often A symmetric); results analogous to those to be described can be obtained for these more general models as well.

Depending on the interpretation (discrete or continuous time), one defines an appropriate behavior beh_Σ, mapping suitable spaces of input functions into spaces of output functions, again in the standard sense of control theory, for any fixed initial state. For instance, in continuous time, one proceeds as follows: For any measurable essentially bounded $u(\cdot) : [0, T] \to \mathbb{R}^m$, denote by $\phi(t, \xi, u)$ the solution at time t with initial state $x(0) = \xi$; this is defined at least on a small enough interval $[0, \varepsilon)$, $\varepsilon > 0$. (The maps σ of interest in neural network theory are usually globally Lipschitz, in which case $\varepsilon = T$.) For each input, let $\mathrm{beh}_\Sigma(u)$ be the output function corresponding to the initial state $x(0) = 0$, that is, $\mathrm{beh}_\Sigma(u)(t) := C(\phi(t, 0, u))$, defined at least on some interval $[0, \varepsilon)$. Two recurrent nets Σ and $\hat{\Sigma}$ (necessarily with the same numbers of input and output channels, i.e. with $p = \hat{p}$ and $m = \hat{m}$) are *equivalent* (in discrete or continuous time, depending on the context) if it holds that $\mathrm{beh}_\Sigma = \mathrm{beh}_{\hat{\Sigma}}$; as before, denote $\Sigma \sim \hat{\Sigma}$. (To be more precise, in continuous-time, one requires that for each u the domains of definitions of $\mathrm{beh}_\Sigma(u)$ and $\mathrm{beh}_{\hat{\Sigma}}(u)$ coincide, and their values be equal for all t in the common domain.)

Electrical circuit implementations of recurrent nets, 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, as well as in signal processing ([25]) and control ([30]), recurrent nets are used as identification models or as prototype dynamic controllers (for partially observed systems or systems given in input/output form); they are often fit to experimental data by means of the gradient-descent optimization (the so-called "dynamic back-propagation" procedure) of some cost criterion.

5.1 Approximation

Recurrent nets provide universal identification models, in a suitable sense. Consider a continuous- or discrete-time, time-invariant, control system Σ:

$$\begin{aligned} \dot{x} \; [\text{ or } x^+ \;] \quad &= \quad f(x,u) \\ y \quad &= \quad h(x) \end{aligned} \tag{7}$$

under standard smoothness assumptions. (For instance, $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, and $y(t) \in \mathbb{R}^p$ for all t, and f and h are continuously differentiable.) For any measurable essentially bounded control $u(\cdot) : [0,T] \to \mathbb{R}^m$, denote by $\phi(t, x_0, u)$ the solution at time t of (7) 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 $\varepsilon = T$; so assume here that the controls being considered are so that solutions exist globally, at least for initial states on some compact set of interest. It is not hard to prove that, on compacts and for finite time intervals, the behavior of Σ can be approximated by the behavior of a recurrent σ-net, if σ is universal.

Assume given two systems Σ and $\widetilde{\Sigma}$, as in (7), where tildes denote data associated to the second system, and with same number of inputs and outputs (but possibly $\widetilde{n} \neq n$). Suppose also given compact subsets $K_1 \subseteq \mathbb{R}^n$ and $K_2 \subseteq \mathbb{R}^m$, as well as an $\varepsilon > 0$ and a $T > 0$. Suppose 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 $t \in [0,T]$. 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.

Assume that σ is a universal activation, 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*.* The proof if not hard, and it involves first simply using universality in order to approximate the right-hand side of the original equation, and then introducing dynamics for the "hidden units" consistently with the equations. This second part requires a little care; for details, see [31].

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.

5.2 Computation

The paper [31], and the references given there, dealt with computational capabilities of recurrent networks, seen from the point of view of classical formal language theory. This work studied discrete-time recurrent networks, focusing on the activation $\sigma = \pi$. (Though more general nonlinearities as well as continuous-time systems are of interest, note that using $\sigma = \text{sign}$ would give no more computational power than finite automata.) The main results —after precise definitions— 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. 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.

To state precisely the simulation results, consider then 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 $u(\cdot) = \alpha_1, \ldots, \alpha_k, 0, 0, \ldots$, 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 y is a sequence of the form

$$y(\cdot) = \underbrace{0, 0, \ldots, 0}_{s}, \beta_1, \ldots, \beta_l, 0, 0, \ldots \ , \tag{8}$$

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 an input as above, and the corresponding output, which is either identically zero or has the form in Equation (8). If $y \equiv 0$, then $\phi(w)$ is undefined; otherwise, if Equation (8) holds, then $\phi(w)$ is defined as the sequence $\beta_1, \ldots, \beta_l$, and one says that the response to the input sequence w was computed *in time* $s + l$. The (partial) function ϕ is *realized* by (Σ, ξ).

In order to be fully compatible with standard recursive function theory, the possibility is allowed 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 $|\alpha_1, \ldots, \alpha_k| = k$, then (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 $\{-1, 1\}$.

Disregarding computation time, some of the main results can be summarized as follows: *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.*

Given $T : \mathbb{N} \to \mathbb{N}$, the language L is *computed in time T* if the corresponding characteristic function is, for some admissible pair that computes in time T. It can be proved that languages recognizable in polynomial time by rational admissible pairs are exacly those in the class P of polynomial-time recursive languages. Using real weights, a new class, "analog P," arises. This class includes many languages not in P, but a theorem shows that NP is most likely not included in analog P.

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.

It is of course much more interesting to impose resource constraints, in particular in terms of computation time. Restrict to language recognition, for simplicity of exposition, but similar results can be given for computation of more general functions. The main result is that the class of languages recognized in polynomial time using recurrent nets with real weights, that is, "analog P," is exactly the same as a class also studied in computer science, namely the class of languages recognized in polynomial time by Turing machines which consult oracles, where the oracles are sparse sets. This gives a precise characterization of the power of recurrent nets in terms of a known complexity class. 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.

5.3 Identifiability

Finally, there are analogs for recurrent nets of the uniqueness questions discussed earlier. Assume from now on that σ is infinitely differentiable, and that it satisfies the following assumptions:

$$\sigma(0) = 0\,,\;\; \sigma^{'}(0) \neq 0\,,\;\; \sigma^{''}(0) = 0\,,\;\; \sigma^{(q)}(0) \neq 0 \text{ for some } q > 2\,. \qquad (*)$$

Let $\mathcal{S}(n,m,p)$ denote the set of all recurrent nets $\Sigma(A,B,C,\sigma)$ with fixed n,m,p. Two nets Σ and $\hat{\Sigma}$ in $\mathcal{S}(n,m,p)$ are *sign-permutation equivalent* if there exists a nonsingular matrix T such that $T^{-1}AT = \hat{A}, T^{-1}B = \hat{B}, CT = \hat{C}$, and T has the special form: $T = PD$, where P is a permutation matrix and $D = \mathrm{diag}(\lambda_1,\ldots,\lambda_n)$, with each $\lambda_i = \pm 1$. The nets Σ and $\hat{\Sigma}$ are just *permutation equivalent* if the above holds with $D = I$, that is, T is a permutation matrix.

Let $\mathbf{B}^{n,m}$ be the class of $n \times m$ real matrices B for which: $b_{i,j} \neq 0$ for all i,j, and for each $i \neq j$, there exists some k such that $|b_{i,k}| \neq |b_{j,k}|$. For any choice of positive integers n,m,p, denote by $S^c_{n,m,p}$ the set of all triples of matrices (A,B,C), $A \in R^{n\times n}$, $B \in R^{n\times m}$, $C \in R^{p\times n}$ which are "canonical" (observable and controllable, as in [33], section 5.5). This is a generic set of triples, in the sense that the entries of the ones that do not satisfy the property are zeroes of certain nontrivial multivariable polynomials. Finally, let:

$$\tilde{\mathcal{S}}(n,m,p) \;=\; \left\{\Sigma(A,B,C,\sigma) \;\middle|\; B \in \mathbf{B}^{n,m} \text{ and } (A,B,C) \in S^c_{n,m,p}\right\}\;.$$

Then, in [1], the following result was proved: *Assume that σ is odd and satisfies property (*). Then $\Sigma \sim \hat{\Sigma}$ if and only if Σ and $\hat{\Sigma}$ are sign-permutation equivalent.* An analogous result can be proved when σ is not odd, resulting in simply permutation equivalence. Also, discrete-time results are available.

6 Acknowledgements

The author acknowledges extremely useful suggestions made by H. Trentelman and M. Vidyasagar, based on a preliminary version of this paper, as well as many constructive comments by C. Darken, S. Hanson, and G. Lafferriere.

References

[1] Albertini, F., and E.D. Sontag, "For neural networks, function determines form," *Neural Networks*, to appear. See also *Proc. IEEE Conf. Decision and Control, Tucson, Dec. 1992*, IEEE Publications, 1992, pp. 26-31.

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

[3] Anthony, M., and N.L. Biggs, *Computational Learning Theory: An Introduction*, Cambridge U. Press, 1992.

[4] Aubin, J.-P., *Mathematical Methods of Artificial Intelligence*, to appear.

[5] Baker, W.L., and J.A. Farrell, "An introduction to connectionist learning control systems," in [39].

[6] Barron, A.R., "Neural net approximation," in *Proc. Seventh Yale Workshop on Adaptive and Learning Systems*, Yale University, 1992, pp. 69-72.

[7] Barto, A.G., "Connectionist learning for control: An overview," in [27].

[8] Baum, E.B., and D. Haussler, "What size net gives valid generalization?," *Neural Computation* **1**(1989): 151-160.

[9] Blum, A., and R.L. Rivest, "Training a 3-node neural network is NP-complete," in *Advances in Neural Information Processing Systems 2* (D.S. Touretzky, ed), Morgan Kaufmann, San Mateo, CA, 1990, pp. 9-18.

[10] Blumer, A., A. Ehrenfeucht, D. Haussler, and M. Warmuth, "Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension," in *Proc. 18th. Annual ACM Symposium on Theory of Computing*, pp. 273-282, ACM, Salem, 1986.

[11] Carroll, S.M., and B.W. Dickinson, B.W., "Construction of neural nets using the Radon transform," in *Proc. 1989 Int. Joint Conf. Neural Networks*, pp. I: 607–611.

[12] Chen, F.C., and H.K. Khalil, "Adaptive control of nonlinear systems using neural networks," *Proc. IEEE Conf. Decision and Control, Hawaii, Dec. 1990*, IEEE Publications, 1990.

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

[14] Darken, C., M. Donahue, L. Gurvits, and E. Sontag, "Rate of approximation results motivated by robust neural network learning," submitted.

[15] Flick, T.E., L.K. Jones, R.G. Priest, and C. Herman, "Pattern classification using projection pursuit," *Pattern Recognition* **23**(1990): 1367-1376.

[16] Franklin, J.A., "Historical perspective and state of the art in connectionist learning control," *Proc. IEEE Conf. Decision and Control, Tampa, Dec. 1989*, IEEE Publications, 1989.

[17] Haussler, D., "Decision theoretic generalizations of the PAC model for neural net and other learning applications," *Information and Computation* **100**(1992): 78-150.

[18] Hertz, J., A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, 1991.

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

[20] Hunt, K.J., D. Sbarbaro, R. Zbikowski, and P.J. Gawthrop, "Neural networks for control systems: A survey," *Automatica* **28**(1992): 1083-1122.

[21] Judd, J.S., *Neural Network Design and the Complexity of Learning*, MIT Press, Cambridge, MA, 1990.

[22] Leshno, M., V.Ya. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a non-polynomial activation function can approximate any function," *Neural Networks*, 1993, to appear.

[23] Livstone, M.M., J.A. Farrell, and W.L. Baker, "A computationally efficient algorithm for training recurrent connectionist networks," in *Proc. Amer. Automatic Control Conference*, Chicago, June 1992.

[24] Macintyre, M., and E.D. Sontag, "Finiteness results for sigmoidal 'neural' networks," in *Proc. 25th Annual Symp. Theory Computing*, San Diego, May 1993, to appear.

[25] Matthews, M., "A state-space approach to adaptive nonlinear filtering using recurrent neural networks," *Proc. 1990 IASTED Symp. on Artificial Intelligence Applications and Neural Networks*, Zürich, pp. 197-200, July 1990.

[26] McBride, L.E., and K.S. Narendra, "Optimization of time-varying systems," *IEEE Trans. Autom. Control*, **10**(1965): 289-294.

[27] Miller, T., R.S. Sutton, and P.J. Werbos (eds.), *Neural networks For Control*, MIT Press, Cambridge, 1990.

[28] Niranjan, M. and F. Fallside, "Neural networks and radial basis functions in classifying static speech patterns," *Computer Speech and Language* **4** (1990): 275–289.

[29] Pisier, G., "Remarques sur un resultat non publiè de B. Maurey," in *Seminaire d'analyse fonctionelle 1980-1981*, Ecole Polytechnique, Palaiseau, 1981.

[30] Polycarpou, M.M., and P.A. Ioannou, "Neural networks and on-line approximators for adaptive control," in *Proc. Seventh Yale Workshop on Adaptive and Learning Systems*, pp. 93-798, Yale University, 1992.

[31] Siegelmann, H.T., and E.D. Sontag, "Some results on computing with 'neural nets'," *Proc. IEEE Conf. Decision and Control, Tucson, Dec. 1992*, IEEE Publications, 1992, pp. 1476-1481.

[32] Slotine,J.-J., and R.M. Sanner, "Neural networks for adaptive control and recursive identification: a theoretical framework," in *Perspectives In Control* (H.L. Trentelman and J.C. Willems, eds.), Birkhauser, Boston, 1993.

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

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

[35] Sontag, E.D., "Feedback stabilization using two-hidden-layer nets," *IEEE Trans. Neural Networks* **3** (1992): 981-990.

[36] Sontag, E.D., and H.J. Sussmann, "Backpropagation separates where perceptrons do," *Neural Networks*, **4**(1991): 243-249.

[37] Sussmann, H.J., "Uniqueness of the weights for minimal feedforward nets with a given input-output map," *Neural Networks* **5**(1992): 589-593.

[38] Vapnik, V.N., *Estimation of Dependencies Based on Empirical Data*, Springer, Berlin, 1982.

[39] White, D.A., and D.A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, Van Nostrand Reinhold, NY, 1992.

[40] Weiss, S.M., and C.A. Kulikowski, *Computer Systems That Learn*, Morgan Kaufmann, San Mateo, CA, 1991.