

On the Approximation Capability of Recurrent Neural Networks

Barbara Hammer

Department of Mathematics/Computer Science

University of Osnabrück, Albrechtstraße 28, D-49069 Osnabrück,

e-mail: hammer@informatik.uni-osnabrueck.de

Abstract: The capability of recurrent neural networks of approximating functions from lists of real vectors to a real vector space is examined:

Any measurable function can be approximated in probability. Additionally, bounds on the resources sufficient for an approximation can be derived in interesting cases. On the contrary, there exist computable mappings on symbolic data which cannot be approximated in the maximum norm. For restricted input length, some continuous functions on real valued sequences need a number of neurons increasing at least linearly in the input length.

On unary sequences, any mapping with bounded range can be approximated in the maximum norm. Consequently, standard sigmoidal networks can compute any mapping on offline inputs as a computational model.

Keywords: Recurrent neural networks, universal approximation, sigmoidal networks, computational capability

1 Introduction

Recurrent networks are discrete time dynamical systems dealing with sequences of input vectors. They can be used in different ways:

They can serve as a computational function which computes on words, i.e. lists with elements of a finite alphabet. These are accepted or rejected after some time if a specified output unit returns a value corresponding to ‘yes’ or ‘no’, respectively. The fact that a computation does not terminate can be specified in this formalism as well. As a computational model recurrent neural networks are Turing universal [11, 18], for special activation functions they can compute every function [17]. These results demonstrate the power of recurrent networks from a theoretical point of view. However, this is purely theoretically because neural computers of this type do not exist up to now.

A second possibility is to use recurrent networks for the approximation of dynamical systems, e.g. in control theory. Here a nonlinear plant which may be specified by its input-output behavior is modeled by a neural network [15]. For example in [20] it is shown that every dynamical system with continuous transition function can be approximated on a compact input set and time interval because the transition function can be approximated by a standard network arbitrarily well.

A third view is to consider recurrent networks simply as functions that map an input sequence to a real value. The data may be produced by a dynamical system but it can be in principle the output of any function on sequences. In particular, the mapping which is to be approximated is not presented in a recursive form a priori. In practical applications this scenario takes place if structural data, time series or lists are to be considered and the advantage of recurrent networks to deal with inputs of arbitrary length is used [14]. In fact, recurrent networks can be generalized in a natural way to so called folding networks which take not only sequences but more complex structured objects, labeled trees, as inputs. This is a very promising approach which has applications in classical symbolic areas: term classification and theorem proving [5, 16]. In this scenario, recurrent networks and folding networks offer the possibility of using neural and subsymbolic methods in the domain of structured, symbolic, or hybrid data [4].

Here we deal with the third setting and consider the capability of recurrent networks of approximating an unknown function in principle. This capability together with the learnability of recurrent networks established for example in [6] and the existence of learning algorithms [21, 8] justify the use of recurrent networks in practical applications.

Now we proceed as follows: After defining recurrent networks formally we proof their capability of approximating any measurable mapping in probability. If only a finite set of data is dealt with, explicit bounds on the number of neurons can be found. Afterwards, the approximation capability in the maximum norm is considered. There exist functions that cannot be approximated in the maximum norm. If the range is unlimited, this situation can be found even if we restrict to sequences with inputs from a unary alphabet. For limited range this situation may take place for binary sequences, whereas any function on unary sequences can be approximated. In particular, any maybe partial function on a binary alphabet can be computed on offline inputs in exponential time with a standard sigmoidal recurrent network and a limited number of neurons. On restricted input length any continuous function on real valued sequences can be approximated in the maximum norm. But some continuous functions need resources which are increasing at least linearly in the maximum input length. We conclude with a discussion.

2 Definition

A **feedforward neural network** consists of neurons n_1, \dots, n_N that are connected in an acyclic graph. Each connection $n_i \rightarrow n_j$ is assigned a weight $w_{ij} \in \mathbb{R}$ and each neuron n_i is assigned a bias $\theta_i \in \mathbb{R}$. The neurons n_1, \dots, n_m without predecessors are the input neurons. A single neuron n_i computes the function $o_i : \mathbb{R}^m \rightarrow \mathbb{R}$ which is defined recursively as

$$o_i(\mathbf{x}) = \begin{cases} x_i & \text{if } i \leq m, \\ \sigma_i(\sum_{n_j \rightarrow n_i} w_{ji} o_j(\mathbf{x}) + \theta_i) & \text{otherwise} \end{cases}$$

where $\sigma_i : \mathbb{R} \rightarrow \mathbb{R}$ is the activation function of neuron i and $\mathbf{x} = (x_1, \dots, x_m)$ some input vector. The entire network computes the function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ with $f(\mathbf{x}) = (o_{i_1}(\mathbf{x}), \dots, o_{i_n}(\mathbf{x}))$. o_{i_j} are a specified set of so called output units. The neurons which are not input or output neurons are called hidden neurons.

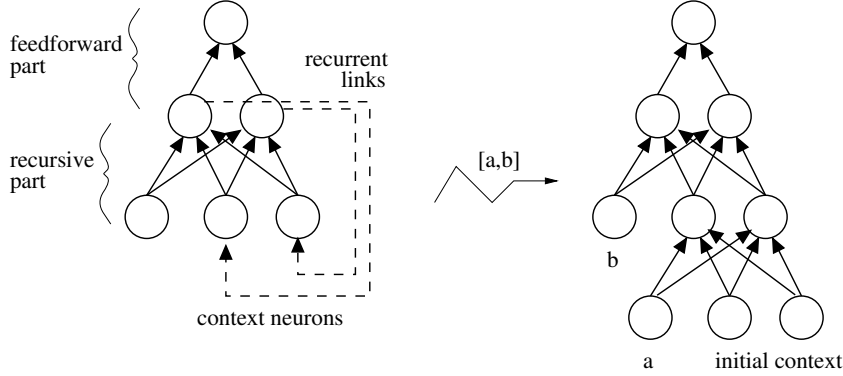


Figure 1: Recurrent network: example for the computation on an input sequence $[a, b]$

In practice, the network graph often has a multilayered structure where the connections are restricted to connections between consecutive layers. Furthermore, the cases $\sigma_i = \text{sgd}(x) = (1 + e^{-x})^{-1}$ for all neurons or the modification $\sigma_i = \text{id}$ for the output units if a function with unlimited range is to be approximated are of special interest in practical applications. In the following, we will assume that all activation functions σ_i equal a certain function σ maybe except for the output units i which may have a linear activation. By a squashing function we refer to any activation function $\sigma : \mathbb{R} \rightarrow [0, 1]$ which is monotonous with $\lim_{x \rightarrow -\infty} \sigma(x) = 0$ and $\lim_{x \rightarrow \infty} \sigma(x) = 1$. A function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ possesses a local linearity if some point $x_0 \in \mathbb{R}$ exists such that σ is continuously differentiable in a neighborhood of x_0 with $\sigma'(x_0) \neq 0$. A property of a function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ holds locally if it is valid in some neighborhood of at least one point.

Now a recurrent neural network adds recurrent connections to a standard feedforward network such that it can deal with sequences of a priori unlimited length. The recurrent network reads step by step an entry of the input sequence, processes this value with its feedforward connections, stores the computed output using the recursive connections, reads the next entry, \dots ; in a formal notation:

Definition 1 $(\mathbb{R}^m)^*$ denotes the set of finite sequences with elements in \mathbb{R}^m . Any function $g : \mathbb{R}^{m+l} \rightarrow \mathbb{R}^l$ and vector $\mathbf{y} \in \mathbb{R}^l$ induces a function $\tilde{g}_{\mathbf{y}} : (\mathbb{R}^m)^* \rightarrow \mathbb{R}^l$ as follows:

$$\tilde{g}_{\mathbf{y}}([\mathbf{x}_1, \dots, \mathbf{x}_i]) = \begin{cases} \mathbf{y} & \text{if } i = 0, \\ g(\mathbf{x}_i, \tilde{g}_{\mathbf{y}}([\mathbf{x}_1, \dots, \mathbf{x}_{i-1}])) & \text{otherwise.} \end{cases}$$

A function $f : (\mathbb{R}^m)^* \rightarrow \mathbb{R}^n$ is computed by a **recurrent neural network** if there exist feedforward networks $g : \mathbb{R}^{m+l} \rightarrow \mathbb{R}^l$ and $h : \mathbb{R}^l \rightarrow \mathbb{R}^n$ such that $f = h \circ \tilde{g}_{\mathbf{y}}$ for some initial context \mathbf{y} .

g is called the recursive part of the recurrent network, h is called the feedforward part. Of course, the function h could be included in the recursive part. But this notation has the advantage that the number of neurons needed for the recursive computation can be

made explicit. This improves bounds on the number of examples necessary to train a network correctly with high reliability [6].

One example of a recurrent network and the computation performed on a special input is depicted in Figure 1. In contrast to feedforward networks, a recurrent network can deal with input sequences of arbitrary length. One can think of the recursive part as a mechanism of encoding the input sequences recursively into a finite dimensional vector space of dimension l . The dimension l is referred to as the encoding dimension of a network. The input neurons number $m + 1, \dots, m + l$ of g are called context neurons since they store the context of an input sequence. In this way recurrent networks are used in speech recognition or time series prediction, for example [3, 12, 14].

In the following, we consider the standard Borel σ -algebra and topology on any finite dimensional real vector space which are both obtained as the smallest algebra and topology, respectively, containing the open intervals as a subset. Consequently, mappings between real vector spaces are continuous if small input deviations cause only small output deviations. Mappings between real vector spaces are measurable in nearly every situation – it requires some tricky mathematical investigation to find a mapping which is not measurable. On $(\mathbb{R}^m)^* = \bigcup_{i=0}^{\infty} (\mathbb{R}^m)^i$ we consider the algebra or topology, respectively, induced by the Borel algebra or standard topology on $(\mathbb{R}^m)^i$. In particular, a mapping $f : (\mathbb{R}^m)^* \rightarrow \mathbb{R}^n$ is measurable or continuous if and only if any restriction $f|_{(\mathbb{R}^m)^i}$ is measurable or continuous. Again this definition implies that any mapping one can think of is measurable and that a mapping is continuous if small deviations of the labels of a tree do not change the output very much.

3 Approximation in probability

First of all we ask the question as to whether a recurrent network exists for every function such that the probability of input sequences where the recurrent network differs considerably from the function to be approximated is small. It will turn out that this is possible in general. Moreover, explicit bounds on the number of neurons sufficient for an interpolation of a finite number of sequences will be derived.

Let P be a probability measure on the lists $(\mathbb{R}^m)^*$. For measurable functions f_1 and $f_2 : (\mathbb{R}^m)^* \rightarrow \mathbb{R}^n$ we say that f_1 **approximates** f_2 with accuracy $\epsilon > 0$ and confidence $\delta > 0$ **in probability** if

$$P(x \in (\mathbb{R}^m)^* \mid |f_1(x) - f_2(x)| > \epsilon) < \delta.$$

That means the probability of points where f_1 approximates f_2 only poorly is small. If the range of f_1 and f_2 is absolutely bounded by some constant B this inequality implies that the two functions differ at most $\epsilon + 2B\delta$ in the L_p -norm, i.e., $(\int |f_1(x) - f_2(x)|^p dP)^{1/p}$ is small. We say that f_1 **interpolates** f_2 on p examples x_1, \dots, x_p if $f_1(x_i) = f_2(x_i)$ holds for any i .

We are interested in the capability of recurrent networks of approximating a measurable function $f : (\mathbb{R}^m)^* \rightarrow \mathbb{R}^n$ arbitrarily well in probability. Note that $P((\mathbb{R}^m)^{>i_0})$ where $(\mathbb{R}^m)^{>i_0} = \bigcup_{i>i_0} (\mathbb{R}^m)^i$ gets arbitrarily small for large i_0 . Therefore it is sufficient

to approximate functions only up to a certain length of the input sequences. This restriction corresponds to the choice of an index i_0 such that $P((\mathbb{R}^m)^{>i_0}) < \delta$. This fact would offer the possibility of using known results for our purpose: To show the approximation capability we could use the ability of recurrent networks of approximating dynamical systems on a limited time interval [20]. It would remain to show that the values of an arbitrary function f which is not written in a recursive form a priori can be produced by a dynamical system with continuous transition function on compact intervals and up to a fixed time. This can be done using appropriate polynomials and a counter for the time steps, for example.

However, since the entire transition function is to be approximated with a network no bounds on the number of neurons result. Therefore we construct a network approximating f directly. This method has the advantage that for lists with entries from a finite alphabet the number of neurons in the recursive part is limited by 1 whereas in the approximation of a dynamical system no bound can be derived. Furthermore, we obtain bounds on the number of neurons which are sufficient to interpolate a finite set of data exactly.

Lemma 2 *Assume $\Sigma = \{1, \dots, b\} \subset \mathbb{N}$. Any function $f : \Sigma^* \rightarrow \mathbb{R}^n$ can be approximated arbitrarily well in probability with a recurrent neural network $h \circ \tilde{g}_y$ where $g : \Sigma \times \mathbb{R} \rightarrow \mathbb{R}$ can be computed by one neuron whose activation function σ possesses a local linearity. $h : \mathbb{R} \rightarrow \mathbb{R}^n$ is a multilayer network with one hidden layer with squashing or locally bounded and nonpolynomial activation function and linear outputs. y depends on the activation function σ .*

If a finite set of p sequences with elements in Σ is to be interpolated exactly np hidden neurons in the feedforward part h are sufficient.

Proof: Because Σ^* decomposes into the measurable subsets of sequences of fixed length a finite subset $S \subset \Sigma^*$ exists such that $P(\Sigma^* \setminus S) < \delta$. A value in $\{1, \dots, b\}$ needs at most $b_0 = \lceil \log b \rceil$ digits in a decimal representation. The mapping

$$g : \Sigma \times \mathbb{R} \rightarrow \mathbb{R}, \quad g(x, z) = (x + z) \cdot (0.1)^{b_0}$$

induces a mapping \tilde{g}_0 which simply concatenates the elements of the sequence expanded by 0 to exactly b_0 digits in a decimal representation: $\tilde{g}_0([x_1, \dots, x_i]) = 0.x_i \dots x_1$. \tilde{g}_0 is injective and g can be computed by a recursive network with one computation neuron substituting the identity via the uniform approximation

$$\frac{\sigma(x_0 + \epsilon x) - \sigma(x_0)}{\epsilon \sigma'(x_0)} \approx x$$

for small ϵ , values x in a compact interval, and an activation function σ with a local linearity at x_0 . We can choose ϵ such that the resulting approximation of \tilde{g}_0 remains injective on S . Furthermore, this approximation can be computed by a network with only one computation neuron because the linear terms $\epsilon \sigma'(x_0)$ and $\sigma(x_0)$ can be considered to be part of the links in the recursive steps. The initial context is to be changed to $\sigma(x_0)$. Formally, this argumentation uses the identity

$$\widetilde{(A \circ f)}_{\mathbf{y}'} = A \circ (f \circ \widetilde{(\text{id}, A)})_{\mathbf{y}}$$

for mappings A and f with appropriate arity and $\mathbf{y}' = A(\mathbf{y})$.

On the image $\tilde{g}_0(S)$ of the recursive computation we can define a function h such that $h(\tilde{g}_0(s)) = f(s)$ because \tilde{g}_0 is injective on S . h can be completed to a continuous function and therefore be approximated arbitrarily well by a feedforward network [9, 10](Theorem 1 & 2, Theorem 2.4). As shown in [19](Lemma 9.1) it is sufficient to use np hidden neurons for an exact interpolation of p points. The composition of these mappings h and \tilde{g}_0 approximates f . \square

Hence we obtain a bound on the neurons in the recursive part if symbolic data is considered. The neurons in the feedforward part are restricted if a finite set of data is to be interpolated. If an activation function like the standard sigmoidal activation is used in the hidden nodes of the feedforward part the number of hidden neurons sufficient for an exact interpolation can be improved to $2pn/(n+1)$ because of [2].

Furthermore, the construction can be expanded to approximate a measurable function $f : (\mathbb{R}^m)^* \rightarrow \mathbb{R}^n$, too. It is sufficient to approximate a discretization of f which corresponds to a function on lists with entries in a finite alphabet:

Theorem 3 *Any measurable function $f : (\mathbb{R}^m)^* \rightarrow \mathbb{R}^n$ can be approximated arbitrarily well in probability by a recurrent network $h \circ \tilde{g}_y$ where $g : \mathbb{R}^{m+1} \rightarrow \mathbb{R}$ is a multilayer network with one hidden layer with squashing activation function which possesses a local linearity and an output activation function σ with a local linearity. $h : \mathbb{R} \rightarrow \mathbb{R}^n$ is a multilayer network with one hidden layer with squashing or locally Riemann integrable and nonpolynomial activation function and linear outputs. y depends on σ .*

If a finite set of p sequences with b different elements is to be interpolated, $2bm$ hidden neurons in g and np hidden neurons in h are sufficient.

Proof: The idea of the proof is as follows: For a continuous function f the input can be slightly changed without changing the outputs too much. Now a real valued input is first mapped to a value from a finite alphabet which characterizes a small interval of the input range where the real vector belongs to. The discrete value is processed like in the purely symbolic case described in Lemma 2. Finally it is shown that the scanning of the real numbers into a finite alphabet can be integrated into the recursive part of a recurrent network.

It follows from [10](Theorem 2.4), for example, that any measurable function $f|(\mathbb{R}^m)^i$ can be approximated arbitrarily well with accuracy $\epsilon/2$ and confidence $\delta \cdot (0.5)^{i+1}$ by a continuous function and therefore f with accuracy $\epsilon/2$ and confidence $\delta/2$ by a continuous function. Consequently we can assume that f itself is continuous.

First we discretize f . Choose $T > 0$ such that $P((\mathbb{R}^m)^{>T}) < \delta/4$. Choose $B > 0$ such that $P(([-\infty, -B] \cup [B, \infty])^m)^{\leq T} < \delta/4$. $f|([-B, B]^m)^i$ is equicontinuous for any $i \leq T$ therefore we can find some $\epsilon_0 > 0$ such that for any two sequences $[\mathbf{x}^1, \dots, \mathbf{x}^i]$ and $[\mathbf{y}^1, \dots, \mathbf{y}^i]$ in $([-B, B]^m)^i$ with $i \leq T$ and $|x_k^j - y_k^j| \leq \epsilon_0$ for all elements \mathbf{x}^j and \mathbf{y}^j , $j \leq i$, and coefficients $k \leq m$ the images under f differ at most $\epsilon/2$. Decompose $] - B, B[$ into disjoint intervals

$$I_1 =] - B, b_1[, I_2 =]b_1, b_2[, \dots, I_q =]b_{q-1}, B[$$

of diameter at most ϵ_0 such that the probability of sequences in $([-B, B]^m)^{\leq T}$ which contain at least one element with a coefficient equal to some b_i is smaller than $\delta/4$.

Now we have obtained a discretization: We can map a sequence $[\mathbf{x}_1, \mathbf{x}_2, \dots]$, which is characterized by the intervals I_{i_1}, I_{i_2}, \dots the components of \mathbf{x}_i belong to, to an arbitrary fixed value where some sequence of the same length with elements with coefficients in I_{i_1}, I_{i_2}, \dots is mapped to. The value differs at most $\epsilon/2$ from $f([\mathbf{x}_1, \mathbf{x}_2, \dots])$ except for a set of probability smaller than $3\delta/4$ in $(\mathbb{R}^m)^*$.

The mapping $c : \mathbb{R}^m \rightarrow \Sigma = \{1, \dots, q^m\}$,

$$\mathbf{x} = (x_1, \dots, x_m) \mapsto 1 + \sum_{i=1}^m q^{i-1} \sum_{j=1}^q (j-1) 1_{I_j}(x_i)$$

encodes the information to which interval the coefficients of \mathbf{x} belong. Here 1_{I_j} is the characteristic function of I_j . The single components x_i each belong to some interval with index $j_i \in \{1, \dots, q\}$. The representation (j_1, \dots, j_m) of the sequence $[x_1, \dots, x_m]$ is mapped to a number in Σ interpreting the vector as the digits of a natural number denoted with respect to base q . Therefore a mapping $h \circ \tilde{g}_y$ approximating the discretization $f_d : \Sigma^* \rightarrow \mathbb{R}^m$ where $[x_1, \dots, x_i] \mapsto f([\mathbf{x}_1, \dots, \mathbf{x}_i])$ for some vectors \mathbf{x}_j with $x_j = c(\mathbf{x}_j)$ gives rise to an approximation $h \circ \tilde{G}_y$ of f where

$$G : \mathbb{R}^{m+1} \rightarrow \mathbb{R}, \quad G(\mathbf{x}, z) = g(c(\mathbf{x}), z).$$

$h \circ \tilde{g}_y$ exists because of Lemma 2. Because of the construction g and h can be chosen as continuous mappings.

We approximate the indicator function 1_{I_j} in c by

$$\sigma\left(\frac{x - b_{i-1}}{\epsilon_1}\right) + \sigma\left(\frac{b_i - x}{\epsilon_1}\right) - 1$$

where $b_0 = -B$ and $b_q = B$ for $\epsilon_1 \rightarrow 0$ and a squashing activation function σ . We approximate the identity in g by an activation function σ with a local linearity. This approximation changes $h \circ \tilde{G}_y$ by at most $\epsilon/2$ except for inputs with some label with a coefficient approximately equal to some b_i . If ϵ_1 is small this leads to an increase of the confidence at most $\delta/4$. In a network implementation this method results in a hidden layer in the recursive part as described in the theorem.

If only a finite set of data is to be interpolated it is sufficient to choose the intervals I_j as intervals which scan only the possible input elements uniquely resulting in a number of $2bm$ hidden neurons in the recursive part. Furthermore, the recursive part can be chosen such that it is injective on the finite number of input sequences resulting in a bound for the hidden layer of the feedforward part. \square

Hence it follows that any measurable function which is not written in a recursive form a priori can be approximated by a recurrent neural network. Upper bounds for the number of neurons are useful for any concrete learning problem in applications. Furthermore, the same technique as in Theorem 3 can be used to expand the proof in [7]

to a similar result concerning the approximation of mappings on trees with real valued labels. As a consequence of this theorem recurrent networks are in principle well suited for approximation purpose as long as we do not want to approximate a function on every input sequence of arbitrary length. If we use inputs from a finite alphabet for modeling terms, for example, we need only one neuron in the recursive part. I.e., the situation is particularly well behaved for symbolic data. However, even for the real valued case explicit bounds on the number of neurons which are sufficient for an exact interpolation of a finite set of data have been derived, too.

Since we have approximated any measurable mapping with a recurrent network which leads in particular to a recursive representation we can apply the argumentation of [20] to the situation. Compared to Theorem 3 the following result reduces the number of hidden layers but does not lead to bounds on the number of neurons.

Corollary 4 *Any measurable function $f : (\mathbb{R}^m)^* \rightarrow \mathbb{R}^n$ can be approximated by a recurrent network $h \circ \tilde{g}_{\mathbf{y}}$ in probability where h can be chosen as a linear mapping and g can be chosen as a feedforward network without hidden layer and a squashing or locally Riemann integrable and nonpolynomial activation function.*

Proof: As already shown f can be approximated by a recurrent network $f_1 \circ \tilde{f}_{2\mathbf{y}}$ with continuous functions f_1 and f_2 computed by feedforward networks with activation function sgd , for example. For an approximation in probability, it is sufficient to approximate $f_1 \circ \tilde{f}_{2\mathbf{y}}$ only on sequences up to a fixed length with elements in a compact set. We can assume that f_1 is linear adding n neurons which compute the output values of f_1 to the outputs of f_2 in the encoding layer if necessary. f_2 can be approximated arbitrarily well by a feedforward network g with one hidden layer with squashing or locally Riemann integrable and nonpolynomial activation function on any compact set in the maximum norm [9, 10](Theorem 1, Theorem 2.4). In particular, f_2 can be approximated by g such that the induced function $\tilde{g}_{\mathbf{y}}$ differs at most ϵ on the relevant sequences. We can consider the linear outputs of this approximation to be part of the weights in the first layer of g or the function f_1 , respectively, which leads to a change of the initial context \mathbf{y} . Setting $h = f_1$, a network of the desired structure results. \square

In particular, this result minimizes the number of hidden layers and is valid for the perceptron activation function in the recursive part as well. But the encoding dimension cannot be limited in general.

4 Approximation in the maximum norm

Here we ask whether any reasonable mapping can be approximated by a recurrent network such that the approximation differs at most ϵ from the function to be approximated for every input – in contrast to the last paragraph where coincidence was only required for a subset of high probability. We show that in the symbolic case this is possible when dealing with functions on unary sequences and a restricted domain. Weakening this conditions leads to examples of mappings which cannot be approximated in such a way.

The positive result does not seem useful for practical applications since usually one does not consider only unary sequences. However, the result shows that recurrent networks have more power than finite automata. We expand this result to a proof for the fact that a sigmoidal recurrent network can compute every mapping in exponential time in an appropriate formalism. Afterwards we show that real valued data does not allow an approximation in the maximum norm even for restricted inputs unless the recursive part of an approximating network is allowed to write all entries of the input sequences into a vector of very large dimension, i.e., is allowed to implement simply some trivial encoding on the sequences.

In the feedforward case it is well known that any continuous mapping can be approximated on a compact set arbitrarily well in the **maximum norm** [10]. We ask the same question in the recurrent case, i.e., does for every function f and positive ϵ some network g exist with $|f(x) - g(x)| < \epsilon$ for every x ? In particular, inputs of arbitrary length are to be approximated which is of interest when assessing the long time behavior of a time series, for example. Actually, this reaches the topic of computability. Recurrent networks where the activation function is for any point Lipschitz continuous in some neighborhood can be simulated by a family of non-uniform Boolean circuits with resources limited by the time the network uses for the computation [17]. Therefore, any function that is not computable by a non-uniform circuit family with resources growing at most polynomially cannot be computed by a recurrent network. It remains to show that such a function exists.

The result of [17] indicates that there may be problems if a function which is computationally hard is to be approximated. Here we do not use this simulation but construct a computable function which cannot be approximated by a recurrent network in the maximum norm directly. The first construction holds even for inputs from a unary alphabet and for any continuous activation function.

Example: Assume the activation function is continuous. We show the existence of a function $f : \{1\}^* \rightarrow \mathbb{R}$ which cannot be approximated by any reasonable network. On every sequence x_i in the set of unary sequences $\{x_1 = [1], x_2 = [1, 1], \dots\}$ a function value $y_i \in \mathbb{R}$ is chosen such that it lies outside the range of any network with at most i neurons and weights absolutely bounded by i . This is possible because a network architecture defines a continuous mapping on the weights for any fixed input sequence x_i . The values (x_i, y_i) can be completed to a continuous mapping on \mathbb{R}^* which is computable on the restriction to \mathbb{N}^* . This function cannot be approximated by any recurrent network because it increases too fast. A network which approximates f up to sequences of length i necessarily contains at least $i + 1$ neurons or some weight with absolute value larger than i . \square

Even if the function that is to be approximated has a limited range an approximation with a network may be impossible. But here we need inputs from a binary alphabet.

Example: Assume that the activation function σ possesses a local linearity. Assume the number of sequences of length T where any mapping to $\{0, 1\}$ can be approximated by a recurrent network with some fixed architecture with N neurons and activation function

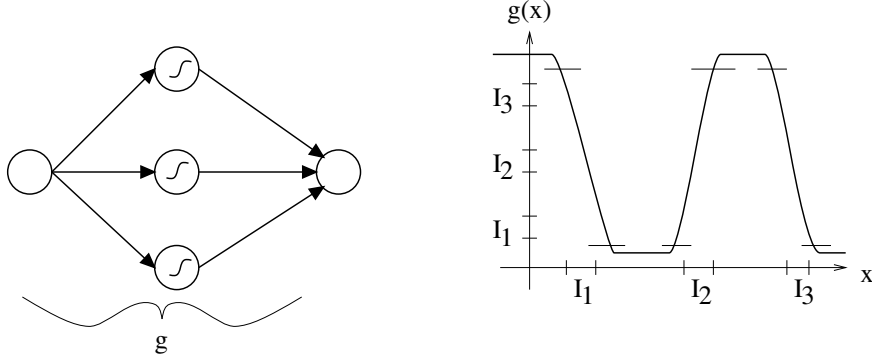


Figure 2: Function g with the property $g(I_j) \supset \cup_{i=1}^3 I_i$ for $j = 1, 2, 3$.

σ and an appropriate choice of the weights is limited by a function $p(T, N)$ which is polynomial in T . Then a function $\{0, 1\}^* \rightarrow \{0, 1\}$ which cannot be approximated by a recurrent network with activation function σ can be constructed as follows:

Assume the function is constructed on sequences up to length T_0 . Assume N_0 is the minimum number of neurons such that some network with N_0 neurons exists which approximates these points. Note that any network with N neurons and some connection structure can be simulated by a standard multilayer network with N^2 neurons where the identity is approximated using the local linearity of σ . Choose $T > T_0$ such that $2^T > p(T, N_0^2)$. Then there exists at least one mapping on the sequences of length T and values in $\{0, 1\}$ that cannot be approximated by a network with N_0 neurons. \square

Note that this argumentation is constructive if the loading problem for feedforward networks is solvable. The number of input sequences which can be mapped to arbitrary values approximating $\{0, 1\}$ by an appropriate choice of the weights in a network structure is limited by the pseudodimension of the network structure. This pseudodimension measures the capacity of the architecture and plays a key role when considering the learnability of networks. For all standard architectures and for all architectures which are useful for efficient learnability, in particular, the pseudodimension is polynomial as required in the example.

As a consequence, there exist computable mappings that cannot be approximated in the maximum norm. But on the other hand there exist mappings that are not computable but can be implemented by a neural network [17]. In fact, any mapping on unary input sequences with limited domain can be approximated:

Theorem 5 *Assume σ is a continuous squashing function. Then any function $f : \{1\}^* \rightarrow [0, 1]$ can be approximated in the maximum norm by a recurrent network without hidden layer and activation function σ in the recursive part and one linear neuron in the feedforward part. The number of neurons which is sufficient for an approximation can be limited with respect to the accuracy ϵ .*

Proof: For unary sequences only the length of the input is relevant. At each recursive time step there is given one value which is to be approximated by the network's output. First, the output range is discretized according to the required approximation accuracy. Choose $n \in \mathbb{N}$, n even with $1/n \leq \epsilon$. Define $x_i = 1/(2n) + (i-1)/n$ for $i = 1, \dots, n$, $I_i = [x_i - 1/(4n), x_i + 1/(4n)]$. Because of the choice of n it is sufficient to show that instead of a specified output value y only some value in the interval where y belongs to or is nearest to can be produced in a recursive computation step. Therefore it is sufficient to show that a network can be constructed the output sequence of which has entries in I_{i_1}, I_{i_2}, \dots for any given sequence of indices i_1, i_2, \dots . In the following we construct a function such that the image of each interval I_i contains all intervals I_1, I_2, \dots . Hence for any value in I_i an inverse image in each interval I_1, I_2, \dots exists; the recursive choice of an appropriate inverse image leads to an initial context.

In detail: Since σ is a squashing function we can find some $K > 0$ such that

$$\sigma(Kx) \begin{cases} > 1 - 1/(8n^2) & \text{if } x > 1/(4n), \\ < 1/(8n^2) & \text{if } x < -1/(4n). \end{cases}$$

The function

$$g(x) = \sum_{i=1}^n \sigma((-1)^i K \cdot (x - x_i)) - (n/2 - 1)$$

has the property $g(I_j) \supset \bigcup_{i=1}^n I_i$ because of the continuity of g and

$$\begin{aligned} g(x_j - 1/(4n)) &\leq \sum_{i < j, i \text{ even}} 1 + \sum_{i < j, i \text{ odd}} 1/(8n^2) + 1/(8n^2) \\ &\quad + \sum_{i > j, i \text{ even}} 1/(8n^2) + \sum_{i > j, i \text{ odd}} 1 - (n/2 - 1) \\ &\leq (n+1)/(8n^2) \leq 1/(4n) \end{aligned}$$

if j is even, $g(x_j - 1/(4n)) \geq 1 - 1/(4n)$ if j is odd, $g(x_j + 1/(4n)) \geq 1 - 1/(4n)$ if j is even, and $g(x_j + 1/(4n)) \leq 1/(4n)$ if j is odd (see Fig. 2).

g is trivially expanded to inputs from \mathbb{R}^2 by just ignoring the first component of the input. The function $\tilde{g}_y : \mathbb{R}^* \rightarrow \mathbb{R}$ can be implemented by a recurrent network with n neurons with activation function σ in the recursive part and a linear output in the feedforward part. The linearity in g as defined above is considered to be part of the weights in the networks g and h , respectively. Furthermore, \tilde{g}_y can approximate any function $f : \{1\}^* \rightarrow [0, 1]$ in the maximum norm with accuracy ϵ by an appropriate choice of y . It is sufficient to choose a value y in $\bigcap_{i \in \mathbb{N}} (g^i)^{-1}(I_{k_i})$ if $f(\underbrace{[1, \dots, 1]}_{i \text{ times}}) \in [x_{k_i} -$

$1/(2n), x_{k_i} + 1/(2n)]$. Note that such a value exists because I_{k_i} is compact, g is continuous, and for any finite number i_0 the intersection $\bigcap_{i \leq i_0} (g^i)^{-1}(I_{k_i})$ is not empty because of the property $g(I_j) \supset \bigcup_i I_i$. Since in a network implementation the recursive part consists only of one layer with squashing activation this value y has to be changed to the initial context $(y + n/2 - 1, 0, \dots, 0)$. \square

The construction can be expanded to show that a sigmoidal network can compute any mapping on offline inputs in exponential time. An equivalent result is already known for

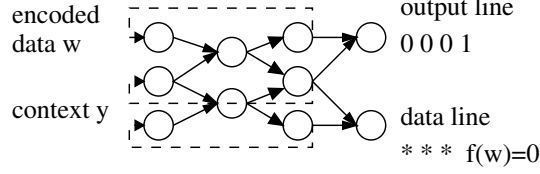


Figure 3: Recurrent neural network as a computational model

the semilinear activation function [17] but for the standard sigmoidal function only the Turing capability with an exponential increase of time was established in [11].

To define the notation of a computation with recurrent networks formally, assume that $c : \{0, 1\}^+ \rightarrow \mathbb{R}$ is an injective function which serves as an encoding function. A maybe partial function $f : \{0, 1\}^+ \rightarrow \{0, 1\}$ is **computed** by a recurrent network **on offline inputs** which are encoded via c if the recurrent network computes $h \circ \tilde{g}_{(c(x), \mathbf{y})} : (\mathbb{R}^0)^* \rightarrow \mathbb{R}^2$, where h , g , and \mathbf{y} are fixed for any f , such that the following holds for some $\epsilon \in]0, 0.5[$:

- For any word $x \in \{0, 1\}^+$ where $f(x)$ is defined a number $t \in \mathbb{N}$, the computation time, exists with $h \circ \tilde{g}_{(c(x), \mathbf{y})}(\underbrace{[\top, \dots, \top]}_{t \text{ times}}) \in [f(x) - \epsilon, f(x) + \epsilon] \times [1 - \epsilon, 1]$ and for any prefix of this input string with shorter length the network outputs some value in $\mathbb{R} \times [0, \epsilon]$.
- If $f(x)$ is not defined $h \circ \tilde{g}_{(c(x), \mathbf{y})}(\underbrace{[\top, \dots, \top]}_{t \text{ times}}) \in \mathbb{R} \times [0, \epsilon]$ for any $t \in \mathbb{N}$.

Here \top stands for the dummy element in \mathbb{R}^0 . Consequently, two output neurons exist, one for the data and one that indicates whether data is present or the network is still computing. The input is encoded in the initial value of one neuron directly using c and all input neurons of the recurrent network are dropped. See Fig 3 for an example. In the recursive part, the encoded input is stored in one neuron, afterwards the recursive computation proceeds as usual. In any recursive step the output line produces a value 0 or 1, in the latter case the output of the computation, $f(w)$ can be found at the data line.

Corollary 6 *Assume the encoding function $c : \{0, 1\}^+ \rightarrow \mathbb{R}$ is given by $c([x_1, \dots, x_j]) = (3^{-\sum_{i=1}^j x_i 2^{i-1}} + 1)/2$. Then there exists a recurrent network with standard sigmoidal activation and 19 neurons such that for any maybe partial function $f : \{0, 1\}^+ \rightarrow \{0, 1\}$ an initial context \mathbf{y} exists such that for any input sequence x the recurrent network with initial context $(c(x), \mathbf{y})$ computes $f(x)$ on offline inputs.*

Proof: The idea of the construction is as follows: The outputs of the function f are reproduced by one part of a recursive network: Some kind of memory outputs the value $f(n)$, if existing, in the n th recursive step. Additionally, a part implementing a counter

is included. This part counts until the input value is reached and then tells the memory that its output at this time step is the output of the entire network.

More precisely: It follows from Theorem 5 that a recurrent network $h_1 \circ (\tilde{g}_1)_{\mathbf{y}}$ exists with a linear unit h_1 and a sigmoidal network g_1 and the property

$$h_1 \circ (\tilde{g}_1)_{\mathbf{y}}(\underbrace{[\top, \dots, \top]}_{n \text{ places}}) \begin{cases} < 0.1 & \text{if } f(x) = 0, \\ \in]0.45, 0.55[& \text{if } f(x) \text{ is not defined,} \\ > 0.9 & \text{if } f(x) = 1 \end{cases}$$

for $x = [x_1, \dots, x_j]$ and $n = \sum_i x_i 2^i$. Only the initial context \mathbf{y} depends on f , the other weights can be chosen as fixed values. Additionally, there exists a network $h_2 \circ (\tilde{g}_2)_{c(x)}$ with a sigmoidal network g_2 and a linear unit h_2 such that

$$h_2 \circ (\tilde{g}_2)_{c(x)}(\underbrace{[\top, \dots, \top]}_{n \text{ places}}) \begin{cases} < 0.15 & \text{if } n < \sum_i x_i 2^i, \\ \in]0.2, 0.4[& \text{if } n = \sum_i x_i 2^i, \\ > 0.7 & \text{if } n > \sum_i x_i 2^i \end{cases}$$

as we will show immediately. The simultaneous computation of $h_1 \circ (\tilde{g}_1)_{\mathbf{y}}$, the memory, and $h_2 \circ (\tilde{g}_2)_{c(x)}$, the counter, with outputs o_1 and o_2 , respectively, is combined with the computation

$$(o_1 > 0.9) \wedge o_2 \in]0.2, 0.4[$$

for the output line and

$$((o_1 > 0.9) \vee (o_1 < 0.1)) \wedge o_2 \in]0.2, 0.4[$$

for the line indicating whether output is present. This latter computation can be approximated arbitrarily well in the feedforward part of a sigmoidal network because the identity, the perceptron activation, and Boolean connections can be approximated. Therefore, the entire construction leads to a sigmoidal network that computes f .

It remains to show that a mapping $h_2 \circ \tilde{g}_2$ with the demanded properties exists. The recursive mapping induced by $x \mapsto 3x$ computes on the initial context $3^{-\sum_i x_i 2^i - 1}$ the value $3^{-\sum_i x_i 2^i - 1 + n}$ for an input of length n . This mapping is combined with the function \tanh which approximates the identity for small values. In fact, $g(x) = \tanh(3x)$ computes

$$\tilde{g}_{3^{-\sum_i x_i 2^i - 1}}(\underbrace{[\top, \dots, \top]}_n) \begin{cases} > 0.7 & \text{if } n \geq \sum_i x_i 2^i + 1, \\ \in](1 - 3^{2(-\sum_i x_i 2^i - 1 + n)})3^{-\sum_i x_i 2^i - 1 + n}, \\ & (1 + 3^{2(-\sum_i x_i 2^i - 1 + n)})3^{-\sum_i x_i 2^i - 1 + n}[& \text{otherwise.} \end{cases}$$

Since $|1 - \tanh(x)/x| < x^2/3$ for $x \neq 0$ this can be seen by induction:

If $\tilde{g}_{3^{-k}}(\top^n) \in](1 - 3^{2(-k+n)})3^{-k+n}, (1 + 3^{2(-k+n)})3^{-k+n}[$ for $k > n + 1$ then

$$\begin{aligned} \tilde{g}_{3^{-k}}(\top^{n+1}) &= \tanh(3 \cdot \tilde{g}_{3^{-k}}(\top^n)) \\ &\in]3\tilde{g}_{3^{-k}}(\top^n)(1 - 3\tilde{g}_{3^{-k}}(\top^n)^2), 3\tilde{g}_{3^{-k}}(\top^n)(1 + 3\tilde{g}_{3^{-k}}(\top^n)^2)[\\ &\subset]3^{-k+n+1}(1 - 3^{2(-k+n)})(1 - 3(3^{-k+n})^2(1 - 3^{2(-k+n)})^2), \\ &\quad 3^{-k+n+1}(1 + 3^{2(-k+n)})(1 + 3(3^{-k+n+1})^2(1 + 3^{2(-k+n)})^2)[\\ &\subset]3^{-k+n+1}(1 - 3^{2(-k+n+1)}), 3^{-k+n+1}(1 + 3^{2(-k+n+1)})[\end{aligned}$$

where \top^n denotes the sequence of length n with elements \top . For $k \leq n + 1$ we obtain

$$\tilde{g}_{3-k}(\top^{n+1}) \geq \tanh(3(3^{-1}(1 - 3^{-2}))) = \tanh(8/9) > 0.7.$$

The function $h_2 \circ \tilde{g}_2$ with the desired properties can be obtained because $\tanh(x) = 2 \cdot \text{sgd}(2x) - 1$. \square

However, this result relies on the fact that the computation is performed with perfect reliability. For a computation with finite precision or a computation which is affected by some noise the computational power reduces to at most finite automata as shown in [13].

So far we have only considered mappings with discrete inputs which are to be approximated in the maximum norm. All negative results transfer to the case of continuous labels, of course. But here an additional question occurs: Can any continuous mapping be approximated in the maximum norm with a folding network on restricted inputs? Note that approximation in the maximum norm on restricted inputs is a special case of approximation in probability if we only consider symbolic, i.e. discrete domains. For continuous labels the following result shows that an approximation is possible. However, the encoding dimension necessarily increases in dependence on the maximum input length for realistic networks and some functions that are to be approximated.

Theorem 7 *Choose $T \in \mathbb{N}$ and a compact set $B \subset \mathbb{R}^m$. For any continuous mapping $f : B^{\leq T} \rightarrow \mathbb{R}^n$ and $\epsilon > 0$ a recurrent network $h \circ \tilde{g}_{\mathbf{y}}$ exists such that $|h \circ \tilde{g}_{\mathbf{y}}(x) - f(x)| < \epsilon$ for all $x \in B^{\leq T}$. g can be chosen as a feedforward network without hidden layer and an activation function which possesses a local linearity. h can be chosen as a single hidden layer feedforward network with linear outputs and locally Riemann integrable and nonpolynomial or squashing activation function in the hidden layer. If g is continuous and the interior of B is not empty, the encoding dimension increases at least linearly with T for some ϵ and real valued and continuous f regardless of the number of hidden layers and hidden neurons in g .*

Proof: Without a restriction on the encoding dimension it is easy to construct an encoding g such that $\tilde{g}_{\mathbf{y}}$ simply writes the single elements of an input sequence of length T into one real vector of dimension $(T + 1)m$. An encoding is induced by $h(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, y_1, \dots, y_{Tm})$ and an initial context $(\mathbf{b}, 0, \dots, 0)$ with some $\mathbf{b} \notin B$. This computation can be approximated with an activation function with a local linearity in the maximum norm. h can be chosen such that it approximates the continuous mapping on these codes in $\mathbb{R}^{(T+1)m}$ to \mathbb{R}^n [9, 10].

Surprisingly, this brute force method is in some way the best possible encoding. This fact relies on topological properties of continuous mappings from a high dimensional real vector space into lower dimensions: Such a mapping necessarily maps different points to the same value. Here we use a correlated result, the Theorem of Borsuk-Ullam [1]. This theorem tells us that for any continuous mapping from a sphere in some \mathbb{R}^{n_1} into \mathbb{R}^{n_2} with $n_1 > n_2$ necessarily antipodal points on the sphere exist which are mapped to the same value.

Now we construct a real valued continuous function which requires a linearly increasing encoding dimension because of this fact. Let an encoding dimension $l(T)$ be given which

does not increase linearly with T . Assume g is continuous. Choose $\mathbf{c} \in B$ and $\epsilon > 0$ such that the ball of radius ϵ with center \mathbf{c} is contained in B . Choose T with $m(T-1) > l(T)$. Assume $\mathbf{a}^{11}, \dots, \mathbf{a}^{m(T-1)}$ are different points in B . We consider the following mapping f with images in $[-1, 1]$: The image of a sequence with length T of the form $[\mathbf{x}^1, \dots, \mathbf{x}^{T-1}, \mathbf{a}^{ij}]$ is

$$\frac{x_i^j - c_i}{\max\{|x_k^l - c_k| \mid k, l\}}$$

if the point $(\mathbf{x}^1, \dots, \mathbf{x}^{T-1})$ is not contained in the ball of radius $\epsilon/2$ and center $(\mathbf{c}, \dots, \mathbf{c})$ in B^{T-1} . Otherwise, f is an arbitrary continuation of this function. In particular, sequences $[\mathbf{x}^1, \dots, \mathbf{x}^T, \mathbf{a}^{ij}]$ and $[\mathbf{y}^1, \dots, \mathbf{y}^T, \mathbf{a}^{ij}]$ are mapped to values differing by 2 for some \mathbf{a}^{ij} if $(\mathbf{x}^1, \dots, \mathbf{x}^T)$ and $(\mathbf{y}^1, \dots, \mathbf{y}^T)$ correspond to points on opposite sides of the ball with center $(\mathbf{c}, \dots, \mathbf{c})$.

The approximation $h \circ \tilde{g}_{\mathbf{y}}$ on these sequences of length T decomposes into a mapping $\bar{g} : B^{T-1} \rightarrow \mathbb{R}^{l(T)}$ and $h \circ g : B \times \mathbb{R}^{l(T)} \rightarrow \mathbb{R}$ where necessarily antipodal points in the sphere of radius ϵ and center $(\mathbf{c}, \dots, \mathbf{c})$ in B^{T-1} exist which are mapped by \bar{g} to the same value because of the Theorem of Borsuk-Ullam. Consequently, at least one value of $h \circ \tilde{g}_{\mathbf{y}}$ differs from the desired output at least 1. \square

As a consequence, continuous mappings cannot be approximated in the maximum norm with limited resources even for restricted inputs. The necessity to increase the encoding dimension is another proof that shows that arbitrary mappings cannot be approximated in the maximum norm on sequences with unlimited length with a recurrent network.

5 Conclusion

The capability of recurrent networks of approximating functions arbitrarily well in probability has been established. Consequently, they are in principle well suited for learning tasks where the input consists of lists or sequences, i.e. the input dimension can vary from example to example. Explicit bounds on the number of neurons which are sufficient for an exact interpolation of a finite set of data are of special interest for any concrete learning algorithm. The situation is particularly well behaved if symbolic data is dealt with because the recursive part reduces to only one computation neuron in this case.

Approximation is not possible in the maximum norm in general due to the connection of recurrent networks with inputs of arbitrary length to computational models similar to Turing machines. Furthermore, topological arguments prohibit an approximation of continuous mappings on restricted inputs in the maximum norm with restricted encoding dimension in general. This indicates that problems may occur if the long time behavior of a sequence is to be modeled or mappings which are very sensitive to a small change of the input labels are to be approximated.

However, on unary sequences any mapping with bounded range can be approximated which leads to the interesting theoretical consequence that standard sigmoidal recurrent networks can compute any mapping on offline inputs.

Considering further work, the question whether in the positive approximation results the weights can be restricted by some constant is of interest for practical applications. Our argumentation uses the possibility of approximating the identity or perceptron activation with a standard activation function which results in a priori unlimited weights. Furthermore, it would be nice to limit the number of neurons necessary for the approximation of an entire function. Such a limitation could take the smoothness of the function that is to be approximated into account, for example.

References

- [1] P. Alexandroff and H. Hopf, *Topologie* 1 (Springer-Verlag, Berlin, 1974).
- [2] A. Eliseeff and H. Paugam-Moisy, Size of multilayer networks for exact learning: analytic approach, in: M.C. Mozer, M.I. Jordan, and T. Petsche, eds., *Advances in Neural Information Processing Systems* 9 (The MIT Press, Cambridge, MA, 1997) 162–168.
- [3] J.L. Elman, Distributed representations, simple recurrent networks, and grammatical structure, *Machine Learning* 7 (1991) 195–225.
- [4] C.L. Giles and M. Gori, eds., *Adaptive Processing of Sequences and Data Structures* (Springer-Verlag, Berlin, 1998).
- [5] C. Goller and A. Küchler, Learning task-dependent distributed representations by backpropagation through structure, in: *Proceedings of the IEEE International Conference on Neural Networks* (1996) 347–352.
- [6] B. Hammer. On the learnability of recursive data, *Mathematics of Control, Signals, and Systems* 12 (1999) 62–79.
- [7] B. Hammer and V. Sperschneider. Neural networks can approximate mappings on structured objects, in: P.P. Wang, ed., *International Conference of Information Sciences* 2 (1997) 211–214.
- [8] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Computation* 9 (1997) 1735–1780.
- [9] K. Hornik, Some new results on neural network approximation, *Neural Networks* 6 (1993) 1069–1072.
- [10] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* 2 (1989) 359–366.
- [11] J. Kilian and H.T. Siegelmann, The dynamic universality of sigmoidal neural networks, *Information and Computation* 128 (1996) 48–56.

- [12] R. Lippmann, Review of neural networks for speech recognition, *Neural Computation* 1 (1989) 1–38.
- [13] W. Maass and P. Orponen, On the effect of analog noise in discrete-time analog computation, *Neural Computation* 10 (1998) 1071–1095.
- [14] M. Mozer. Neural net architectures for temporal sequence processing, in: A. Weigend and N. Gershenfeld, eds., *Predicting the future and understanding the past* (Addison-Wesley, Reading, MA, 1993).
- [15] K.S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks* 1 (1990) 4-27.
- [16] S. Schulz, A. Küchler, and C. Goller, Some experiments on the applicability of folding architectures to guide theorem proving, in: D.D. Dankel, ed., *Proceedings of the 10th International FLAIRS Conference* (Florida AI Research Society, 1997) 377-381.
- [17] H.T. Siegelmann and E.D. Sontag, Analog computation, neural networks, and circuits, *Theoretical Computer Science* 131 (1994) 331–360.
- [18] H.T. Siegelmann and E.D. Sontag, On the computational power of neural networks, *Journal of Computer and System Sciences* 50 (1995) 132–150.
- [19] E.D. Sontag, Feedforward nets for interpolation and classification, *Journal of Computer and System Sciences* 45 (1992) 20–48.
- [20] E.D. Sontag, Neural nets as systems models and controllers, in *7th Yale Workshop on Adaptive and Learning Systems* (1992) 73–79.
- [21] R. Williams and D. Zipser, A learning algorithm for continually running fully recurrent neural networks, *Neural Computation* 1 (1989) 270–280.