Recurrent neural networks with small weights implement definite memory machines*

Barbara Hammer† and Peter Tiňo‡

January 24, 2003

Abstract

Recent experimental studies indicate that recurrent neural networks initialized with 'small' weights are inherently biased towards definite memory machines (Tiňo, Čerňanský, Beňušková, 2002a; Tiňo, Čerňanský, Beňušková, 2002b). This paper establishes a theoretical counterpart: transition function of recurrent network with small weights and 'squashing' activation function is a contraction. We prove that recurrent networks with contractive transition function can be approximated arbitrarily well on input sequences of unbounded length by a definite memory machine.

*We would like to thank two anonymous reviewers for profound and valuable comments on an earlier version of this manuscript.

†Department of Mathematics/Computer Science, University of Osnabrück, D-49069 Osnabrück, Germany, e-mail: hammer@informatik.uni-osnabrueck.de

‡School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK, e-mail: pxt@cs.bham.ac.uk

Conversely, every definite memory machine can be simulated by a recurrent network with contractive transition function. Hence initialization with small weights induces an architectural bias into learning with recurrent neural networks. This bias might have benefits from the point of view of statistical learning theory: it emphasizes one possible region of the weight space where generalization ability can be formally proved. It is well known that standard recurrent neural networks are not distribution independent learnable in the PAC sense if arbitrary precision and inputs are considered. We prove that recurrent networks with contractive transition function with a fixed contraction parameter fulfill the so-called distribution independent UCED property and hence, unlike general recurrent networks, are distribution independent PAC-learnable.

1 Introduction

Data of interest have a sequential structure in a wide variety of application areas such as language processing, time-series prediction, financial forecasting, or DNA-sequences (Laird and Saul, 1994; Sun, 2001). Recurrent neural networks and hidden Markov models constitute very powerful methods which have been successfully applied to these problems, see for example (Baldi et.al., 2001; Giles, Lawrence, Tsoi, 1997; Krogh, 1997; Nadas, 1984; Robinson, Hochberg, Renals, 1996). Successful applications are accompanied by theoretical investigations which demonstrate the capacities of recurrent networks and probabilistic counterparts such as hidden Markov models[1]: the universal approximation ability of recurrent networks has been proved in (Funahashi and Nakamura, 1993), for example; moreover, they can be related to classical computing mechanisms like Turing machines or even more powerful non-uniform Boolean circuits (Siegelmann and Sontag, 1994; Siegelmann and Sontag, 1995).

Standard training of recurrent networks by gradient descent methods faces severe problems (Bengio, Simard, Frasconi, 1994) and the design of efficient training algorithms for recurrent networks is still a challenging problem of ongoing research; see for example (Hochreiter and Schmidhuber, 1997) for a particularly successful approach and a further discussion on the problem of long-term dependencies. Besides, the generalization ability of recurrent neural networks constitutes a further not yet satisfactorily solved question: unlike standard feedforward networks, common recurrent neural architectures possess VC-dimension which depends on the maximum length of input sequences and is hence in theory infinite for arbitrary inputs (Koiran and Sontag, 1997; Sontag, 1998). The VC-dimension can be thought of as expressing flexibility of a function class to perform classification tasks. We will introduce a variant of the VC dimension – the so-called fat-shattering dimension. Finiteness of the VC-dimension is equivalent to the so-called distribution independent PAC learnability, i.e. the ability of valid generalization from a finite training set the size of which depends only on the given function class (Anthony and Bartlett, 1999; Vidyasagar, 1997). Hence, prior distribution independent bounds on the generalization ability of general recurrent networks are not possible. A first step towards posterior or distribution dependent bounds for general recurrent networks without further restrictions can be found in (Hammer, 1999; Hammer, 2000), however, these bounds are weaker than the bounds obtained via

[1]Although hidden Markov models are usually defined on a finite state space – unlike recurrent neural networks which possess continuous states.

a finite VC-dimension. Of course, bounds on the VC dimension of various restricted recurrent architectures can be derived, e.g. for architectures implementing a finite automaton with a limited number of states (Frasconi et.al., 1995), or for architectures with activation function with finite codomain and finite input alphabet (Koiran and Sontag, 1997). Moreover, the argumentation in (Maass and Orponen, 1998; Maass and Sontag, 1999) shows that the presence of noise in the computation severely limits the capacity of recurrent networks. Depending on the support of the noise, the capacity of recurrent networks reduces to finite automata or even less. This fact provides a further argument for the limitation of the effective VC dimension of recurrent networks in practical implementations. However, these arguments rely on deficiencies of neural network training: the bounds on the generalization error which can be obtained in this way become worse the more computation accuracy and reliability can be achieved. The argumentation can only partially account for the fact that recurrent networks often generalize in practical applications after appropriate training and that they may show particularly good generalization behavior if advanced training methods are used (Hochreiter and Schmidhuber, 1997).

We will focus in this article on the initial phases of recurrent neural network training by formally characterizing the function class of recurrent neural networks initialized with small weights. This allows us to compare the behavior of recurrent networks at the early stages of training with alternative tools for sequence-processing. Furthermore, we will show that small weights constitute a sufficient condition for good generalization ability of recurrent neural networks even if arbitrary precision of the computation and arbitrary real-valued inputs are assumed. This argumentation formalizes one aspect of why recurrent neural network training is often successful: initialization with small weights biases neural network training towards regions of the search space where the generalization ability can be rigorously proved. Naturally, further aspects may account for the generalization ability of recurrent networks if we allow for arbitrary weights, e.g the above mentioned corruption of the network dynamics by a noise, implicit regularization of network training due to the choice of the error function, or the fact that regions in the weight space which give a large VC-dimension cannot be found by standard training because of the problem of long-term dependencies.

Alternatives to recurrent networks or hidden Markov models have been investigated for which efficient training algorithm can be found and prior bounds on the generalization ability can be established. One possibility constitute networks with time-window for sequential data or fixed order Markov models. Both alternatives use only a finite memory length, i.e. perform predictions based on a fixed number of sequence entries (Ron, Singer, Tishby, 1996; Sejnowski and Rosenberg, 1987). Particularly efficient modifications are variable memory length Markov models which adapt the necessary memory depth to contexts in the given input sequence (Bühlmann and Wyner, 1999). Various applications can be found in (Guyon and Pereira, 1995; Ron, Singer, Tishby, 1996; Tiňo and Dorffner, 2001), for example. Note that some of these approaches propose alternative notations for variable length Markov models which are appropriate for specific training algorithms such as prediction suffix trees or iterative function systems. Markov models are much simpler than general hidden Markov models since they operate only on a finite number of observable contexts[2]. Nevertheless they are appropriate for a wide variety of applications as shown in the experiments (Guyon and Pereira, 1995; Ron, Singer, Tishby, 1996; Tiňo and Dorffner, 2001) and the dynamics of large definite memory machines can be learned with neural networks as presented in the articles (Clouse et.al., 1997; Giles, Horne, Lin, 1995).

[2]It is not necessary to do inference about the states for Markov models.

However, hidden Markov models or recurrent networks can obviously simulate fixed order Markov models or definite memory machines. We will theoretically show in this article that recurrent networks are *biased* towards definite memory machines through initialization of the weights with small values. Hence standard neural network training first explores regions of the weight space which correspond to the simpler (but potentially useful) dynamics of definite memory machines before testing more involved dynamics such as finite state machines and other mechanisms which can be implemented by recurrent networks (Tiňo and Šajda, 1995). This bias has the effect that structural differentiation due to the inherent dynamics can be observed even prior to training. This observation has been verified experimentally (Christiansen and Chater, 1999; Kolen, 1994a; Kolen, 1994b; Tiňo, Čerňanský, Beňušková, 2002a; Tiňo, Čerňanský, Beňušková, 2002b). Moreover, the structural bias corresponds to the way in which humans recognize language as pointed out in (Christiansen and Chater, 1999), for example. This article establishes a thorough mathematical formalization of the notion of architectural bias in recurrent networks.

Furthermore, initial exploration of simple definite memory mechanisms in standard neural network training focuses on a region of the parameter search space where prior bounds on the generalization error can be obtained. We formalize this hypothesis within the mathematical framework provided by the statistical learning theory. We prove in the second part of this article that recurrent networks with small weights are distribution independent PAC-learnable and hence yield a valid generalization if enough training data are provided. This contrasts with unrestricted recurrent networks with infinite precision that may yield in theory considerably worse generalization accuracy.

We start by defining the notions of definite memory machines, fixed order Markov models and variations thereof which are particularly suitable for learning. Then we show that standard discrete-time recurrent networks initialized with small weights (or more generally, non-autonomous discrete-time dynamical systems with contractive transition function) driven with arbitrary input sequences can be simulated by definite memory machines operating on a finite input alphabet. Conversely, we show that every definite memory machine can be simulated by a recurrent network with small weights. Finally, we link the results to statistical learning theory and show that small weights constitute one sufficient condition for the distribution independent UCED property.

2 Finite memory models for sequence prediction

Assume X is a set. We denote the set of all finite length sequences over X by X^*. The sequences of length at most L are denoted by $X^{\leq L}$. $[\,]$ denotes the empty sequence, $[x_1, \ldots, x_l]$ denotes the sequence of length l and elements $x_i \in X$. For every $L \in \mathbb{N}$, the *L-truncation* $T_L(s)$ of a sequence $s = [x_1, \ldots, x_l]$ is defined as the first part of length L of the sequence, i.e.

$$T_L(s) = \begin{cases} s & \text{if } l \leq L \\ [x_1, \ldots, x_L] & \text{otherwise} \end{cases}$$

We are interested in predictions on sequences, i.e. functions of the form $f : X^* \to X, f(s) = x$, or probability distributions $p(x|s)$ for $x \in X$ given a sequence s, which allow us, e.g. to predict the next symbol x or its probability, respectively, when the sequence s has been observed. We assume that the sequences are ordered right-to-left, i.e. x_1 is the most recent entry in the sequence $[x_1, \ldots, x_l]$. In the next-symbol prediction setting, $f(s) = x$ indicates that the sequence $s = [x_1, \ldots, x_l]$ is completed to $[x, x_1, \ldots, x_l]$ in the next time step. Obviously, a function $f : X^* \to X$ induces the probability $p(x|s) \in \{0,1\}$ with $p(x|s) = 1 \iff f(s) = x$ and can therefore be seen as a special case of the probabilistic formalism.

Assume $X := \Sigma$ is a finite alphabet. A classical and very simple mechanism for next-symbol prediction on sequences over Σ is given by definite memory machines or their probabilistic counterparts, fixed order Markov models, (Ron, Singer, Tishby, 1996).

Definition 2.1 *Assume V is a set. A* definite memory machine (DMM) *computes a function $f : \Sigma^* \to V$, such that some $L \in \mathbb{N}$ exists with*

$$f(s) = f(T_L(s)) \quad \forall s \in \Sigma^* .$$

A fixed order Markov model (FOMM) *defines for each sequence s a probability $p(\cdot|s)$ on V with the following property: Some $L \in \mathbb{N}$ can be found with*

$$p(x|s) = p(x|T_L(s)) \quad \forall x \in V, s \in \Sigma^* .$$

Note that $V = \Sigma$ if the above formalisms are used for predictions on sequences. Only a finite memory of length L is necessary for inferring the next symbol. FOMMs define rich families of sequence distributions and can naturally be used for sequence generation or probability estimation. However, if L increases, estimation of FOMMs on a finite set of examples becomes very hard. Therefore *variable memory length Markov models* (VLMM) have been proposed, where the memory length may depend on the sequence, i.e. they implement probability distributions with

$$p(x|s) = p(x|T_{L(s)}(s)) \quad \forall x \in V, s \in \Sigma^* ,$$

where the length $L(s) \leq L_{\max}$ may depend on the context (Bühlmann and Wyner, 1999; Guyon and Pereira, 1995). The length of the memory is adapted to the context. Since $L(s)$ is universally limited by some value $L_{\max}$, VLMMs constitute a specific efficient implementation of FOMMs. Their in-principle capacity is the same. VLMMs are often represented as prediction suffix trees for which efficient learning algorithms can be designed (Ron, Singer, Tishby, 1996). Alternative models for sequence processing which are more powerful than DMMs and FOMMs are finite state machines and finite memory machines, respectively. The behavior of a finite state machine does only depend on the input and the actual state. Thereby, the state is an element of a finite number of different states. Finite memory machines implement functions the behavior of which can be determined by the last m input symbols and the last n output symbols, for some fixed numbers m and n. Definite memory machines can be alternatively defined as finite memory machines which depend only on the last m input symbols, but no outputs need to be known, i.e. $n = 0$. Formal definitions can be found e.g. in (Kohavi, 1978). Note that definite and finite memory machines cannot produce several simple languages, e.g. they cannot produce the binary number representing the sum of two bitwise presented binary numbers. A finite state machine with only one bit of memory could solve the task. There exists a rich literature which relates recurrent networks (with arbitrary weights) to finite state machines (finite memory machines) and demonstrates the possibility of learning/simulating these models in practice (Carrasco and Forcada, 2001; Frasconi et.al., 1995; Giles, Lawrence, Tsoi, 1997; Omlin and Giles, 1996a; Omlin and Giles, 1996b; Tiňo and Šajda, 1995). Note that definite memory machines constitute particularly simple (though useful) models where only a fixed number of input signals uniquely determines the current output. DMMs are alternatively called DeBruijn automata (Kohavi, 1978). Large DMMs have been successfully learned from examples with recurrent networks as reported e.g. in the articles (Clouse et.al., 1997; Giles, Horne, Lin, 1995).

A very natural way of processing sequences is in a recursive manner. For this purpose, we introduce a general notation of recursive functions induced by standard functions via iteration:

Definition 2.2 *Assume X and U are sets. Every function $f : X \times U \to U$ and element $y \in U$* induces *a recursive function $\tilde{f}_y : X^* \to U$,*

$$\tilde{f}_y(s) = \begin{cases} y & \textit{if } s = [\,] \\ f(x_1, \tilde{f}_y([x_2, \ldots, x_l])) & \textit{if } s = [x_1, \ldots, x_l] \end{cases}$$

y is called the initial context. *The* induced function with finite memory length *L is defined by $\tilde{f}^L_y : X^* \to U$,*

$$\tilde{f}^L_y(s) = \tilde{f}_y(T_L(s))\,.$$

Starting from the initial context y, the sequence $s = [x_1, x_2, \ldots, x_l]$ is processed iteratively, starting from the last entry x_l, applying a transition function f in each step. $\tilde{f}_y$ may use infinite memory in the sense that *all* entries of a sequence may contribute to the output, not just the most recent ones. On the other hand, $\tilde{f}^L_y$ takes into an account only the L most recent entries of the sequence. Functions of the form $\tilde{f}^L_y$ share the idea of DMMs that only a finite memory is available for processing. General recursive functions of the form $\tilde{f}_y$ have more powerful properties. Recurrent neural networks which we will introduce later, constitute one popular mechanism for recursive computation which is more powerful than FLMM. However, we will first shortly mention an alternative to FLMMs which explicitly uses recursive processing.

Fractal prediction machines (FPMs) constitute an alternative approach for sequence prediction through FOMM as proposed in (Tiňo and Dorffner, 2001). Here the L most recent entries of a sequence s are first mapped to a real vector space in a fractal way. Then the fractal codes of L-blocks are quantized into a fixed number of prototypes or codebook vectors. The probability of the next symbol x is defined by the probability vector which is attached to the corresponding nearest codebook vector. Formally, a FPM is given by the following ingredients: The elements $x \in \Sigma$ are identified with binary vectors b_x in $\{0,1\}^D$, $D := \lceil \log |\Sigma| \rceil$. Denote by $R : \Sigma \times [0,1]^D \to [0,1]^D$, the mapping $(x,p) \mapsto \alpha \cdot b_x + (1-\alpha) \cdot p$, where $\alpha \in (0, 1/2]$ is a fixed scalar. Some memory depth $L \in \mathbb{N}$ is fixed. A sequence s is first mapped to $\tilde{R}_y^L(s) \in [0,1]^D$, where $y = \{\frac{1}{2}\}^D$. Sequences are encoded in a fractal way such that all sequences of length at most L are encoded uniquely. In general, if two sequences s_1, s_2 share the most recent entries then their images $\tilde{R}_y^L(s_1)$, $\tilde{R}_y^L(s_2)$ lie close to each other. A finite set of prototypes $p_i \in [0,1]^D$ is given, together with a vector $P^i \in [0,1]^{|\Sigma|}$ for each p_i, with $\sum_{j=1}^{|\Sigma|} P_j^i = 1$ (P_j^i denotes the components of P^i), which represents the probabilities for the next element in the sequence. Hereby, $\|\cdot\|$ denotes the Euclidean metric. Assume $\Sigma = \{x_1, \ldots, x_n\}$. The probability of x_i given s equals the i th entry of the probability vector attached to the codebook vector which is nearest to the fractal encoding of s, i.e.

$$p(x_i|s) = P_i^j \quad \text{s.t. } \|P^j - \tilde{R}_y^L(s)\| \text{ minimal}.$$

This notation has the advantage that an efficient training procedure can immediately be found: If a training set of sequences is given, first all L-blocks are encoded in $[0,1]^D$. Afterwards, a standard vector quantization learning algorithm is applied, e.g. a self organizing map (Kohonen, 1997). Finally, the probability vectors attached to the prototypes are determined such that they correspond to the relative frequencies of next symbols for all L-blocks in the training set codes of which are located in the receptive field of the corresponding codebook. Note that a variable length of the respective memory is automatically introduced through the vector quantization: Regions with a high density of codes attract more prototypes than regions with a low density of codes. Hence the memory length is closer to the maximum length L in the former regions compared to the latter ones.

It is obvious that at most FOMMs can be implemented by FPMs. Conversely, it can be seen easily that each FOMM with corresponding probability p can be approximated up to every desired accuracy with a FPM: We can choose the parameter L in FPM equal to the order of FOMM. Then the encoding in the FPM yields $\tilde{R}^L_y(s_1) = \tilde{R}^L_y(s_2)$ only if the next-symbol-prediction probabilities given by s_1 and s_2 coincide. If enough data points are available, all possible codes in $[0, 1]^D$ of nonzero probability prediction contexts of length L in $[0, 1]^D$ can be observed in the first step of FPM construction. Clustering with a sufficient number of prototypes can simply choose all codes as prototypes, where the nearest prototypes for two codes are identical iff the codes itself are identical. Hence the probabilities attached to a prototype which correspond to the observed frequencies converge to the correct probabilities $p(x_i|s)$ for every s which is mapped to the corresponding prototype. FPM constitute one example for efficient sequence prediction tools. As we will see, recurrent networks initialized with small weights are inherently biased towards these more simple and efficiently trainable mechanisms. Naturally, situations where more complicated dynamics is required and hence recurrent networks with large weights are needed can be easily found.

3 Contractive recurrent networks implement DMMs

We are interested in recursive processing of sequences with recurrent neural networks. The basic dynamics of a recurrent neural network (RNN) used for sequence prediction is given by the above notion of induced recursive functions: A RNN computes a function $h \circ \tilde{f}_y : (\mathbb{R}^n)^* \to \mathbb{R}^m$, where $\tilde{f}_y$ is the function induced by some function $f : \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^l$, which together with $h : \mathbb{R}^l \to \mathbb{R}^m$ are functions of a specific form which defined later. Recurrent networks are more powerful than finite memory models and finite state models for two reasons: They can use an infinite memory and using this memory they can simulate Turing machines, for example, as shown in (Siegelmann and Sontag, 1995). Moreover, they usually deal with real vectors instead of a finite input set such that a priori unlimited information in the inputs might be available for further processing (Siegelmann and Sontag, 1994). Here we are interested in RNNs where the recursive transition function f has a specific property: It forms a contraction. We will see later that this property is automatically fulfilled if a RNN with sigmoid activation function is initialized with 'small' weights, which is a reasonable way to initiate weights, unless one has a strong prior knowledge about the underlying dynamics of the generating source (Elman et.al., 1996). We will show that under these circumstances RNNs can be seen as definite memory machines, i.e. they only use a finite memory and only a finite number of functionally different input symbols exists. This result holds even if arbitrary real-valued inputs are considered and computation is done with perfect accuracy. Hence RNNs initialized in this standard way are biased towards definite memory machines.

First, we formally define contractions and focus on the general case of recursive functions induced by contractions. Assume X and U are sets and $f : X \times U \to U$ is a function. Assume the set U is equipped with a metric structure. We denote the distance of two elements u_1 and u_2 in U by $|u_1 - u_2|$.

Definition 3.1 *A function $f : X \times U \to U$ is a* contraction *with respect to U if a real value $C \in [0, 1)$ exists such that the inequality*

$$|f(x, u_1) - f(x, u_2)| \leq C \cdot |u_1 - u_2|$$

holds for all $x \in X$ and u_1, $u_2 \in U$.

If the transition function is a contraction and U is bounded with respect to the metric then we can approximate the recursive function induced by f by the respective induced function with only a finite memory length:

Lemma 3.2 *Assume $f : X \times U \to U$ is a contraction with parameter $C \in [0, 1)$ with respect to U. Assume $|u_1 - u_2| \leq C_0$ for all u_1, $u_2 \in U$ and fix $\epsilon > 0$. Then, for memory length $L = \log_C(\epsilon/C_0)$, we have*

$$|\tilde{f}_y(s) - \tilde{f}_y^L(s)| \leq \epsilon$$

for every initial context $y \in U$ and every sequence $s \in X^*$

Proof. Choose $s = [x_1, \ldots, x_l] \in X^*$. If $l \leq L$, the inequality follows immediately. Assume $l > L$. Then

$$\begin{aligned}
|\tilde{f}_y(s) - \tilde{f}_y^L(s)| &= |\tilde{f}_y([x_1, \ldots, x_l]) - \tilde{f}_y([x_1, \ldots, x_L])| \\
&= |f(x_1, \tilde{f}_y([x_2, \ldots, x_l])) - f(x_1, \tilde{f}_y([x_2, \ldots, x_L]))| \\
&\leq C \cdot |\tilde{f}_y([x_2, \ldots, x_l]) - \tilde{f}_y([x_2, \ldots, x_L])| \\
&\leq \ldots \\
&\leq C^L \cdot |\tilde{f}_y([x_{L+1}, \ldots, x_l]) - \tilde{f}_y([\,])| \\
&\leq C^L \cdot C_0 = \epsilon/C_0 \cdot C_0 = \epsilon,
\end{aligned}$$

where $L = \log_C(\epsilon/C_0)$. □

Hence we can approximate the dynamics by a dynamics with a finite memory length if the transition function is a contraction. The memory length depends on the parameter C of the contraction. Usually, the space of internal states U is a compact subset of a real vector space, e.g. the set $(0, 1)^d$, d denoting the respective dimensionality. We have already seen that we need only a finite length if we approximate recursive functions with contractive transition function. We would like to go a step further and show that we do not need infinite accuracy for storing the intermediate real vectors in U. Rather, a finite set U will do. For this purpose, we first need an intermediate result.

Definition 3.3 *Assume $\mathcal{F}$ is a function class with domain X and codomain Y, such that Y is equipped with a metric $|\cdot|$. For f, $g \in \mathcal{F}$ we denote the maximum distance by $|f-g|_\infty = \sup_{x\in X} |f(x) - g(x)|$. Assume $\epsilon > 0$. An* external covering *of $\mathcal{F}$ with accuracy ϵ consists of a set of functions $C(\mathcal{F})$ where $g : X \to Y$ may be arbitrary for $g \in C(\mathcal{F})$, such that for all $f \in \mathcal{F}$ a function $g \in C(\mathcal{F})$ can be found with $|f-g|_\infty \leq \epsilon$.*

Note that for every function class an external covering, the class itself, can be found. A *finite ϵ-covering* of a set U consists of a finite number of points $\{u_1, \dots, u_n\}$ such that for every $u \in U$ some u_i with $|u - u_i| \leq \epsilon$ exists. Note that we can find a finite covering for every bounded set in a metric space, i.e. $|u_1 - u_2| \leq C_0$ for all $u_1, u_2 \in U$ and some $C_0 < \infty$.

Denote by $\tilde{\mathcal{F}}$ the set of all functions of the form $\tilde{f}_y$ for $f \in \mathcal{F}$ and $y \in U$. $\tilde{\mathcal{F}}_L$ denotes the set of all functions of the form $\tilde{f}^L_y$ for $f \in \mathcal{F}$ and $y \in U$. External coverings of $\mathcal{F}$ extend to external coverings of $\tilde{\mathcal{F}}$ and $\tilde{\mathcal{F}}_L$, respectively:

Lemma 3.4 *Assume $\mathcal{F}$ is a set of functions mapping $X \times U$ to U, such that every $f \in \mathcal{F}$ forms a contraction with respect to U with parameter C. Assume $\epsilon > 0$. Assume $C(\mathcal{F})$ is an external covering of $\mathcal{F}$ with parameter ϵ. Assume U is bounded and the constants $\{u_1, \dots, u_n\}$ cover U with parameter ϵ. Then $\{\tilde{g}_{u_j} | g \in C(\mathcal{F}), j = 1, \dots, n\}$ is an external covering of $\tilde{\mathcal{F}}$ with parameter $\epsilon/(1-C)$ and $\{\tilde{g}^L_{u_j} \mid g \in C(\mathcal{F}), j = 1, \dots, n\}$ is an external covering of $\tilde{\mathcal{F}}_L$ with parameter $\epsilon/(1-C)$.*

Proof. Assume $f \in \mathcal{F}$ and $y \in U$. Choose a function g from the covering $C(\mathcal{F})$ such that $|f-g|_\infty \leq \epsilon$. Choose a value u_j from $\{u_1, \dots, u_n\}$ such that $|y - u_j| \leq \epsilon$.

It follows by induction over the length l of a sequence $s \in X^*$ that $|\tilde{f}_y(s) - \tilde{g}_{u_j}(s)| \leq \epsilon \cdot (1 - C^{l+1})/(1 - C)$ as follows:

For $s = [\,]$ we find

$$|\tilde{f}_y(s) - \tilde{g}_{u_j}(s)| = |y - u_j| = \epsilon \leq \epsilon \cdot \frac{1 - C^1}{1 - C}\,.$$

For $s = [x_1, \ldots, x_l]$ we find

$$\begin{aligned}
& |\tilde{f}_y([x_1, \ldots, x_l]) - \tilde{g}_{u_j}([x_1, \ldots, x_l])| \\
= \;& |f(x_1, \tilde{f}_y([x_2, \ldots, x_l])) - g(x_1, \tilde{g}_{u_j}([x_2, \ldots, x_l]))| \\
\leq \;& |f(x_1, \tilde{f}_y([x_2, \ldots, x_l])) - f(x_1, \tilde{g}_{u_j}([x_2, \ldots, x_l]))| \\
& + |f(x_1, \tilde{g}_{u_j}([x_2, \ldots, x_l])) - g(x_1, \tilde{g}_{u_j}([x_2, \ldots, x_l]))| \\
\leq \;& C \cdot |\tilde{f}_y([x_2, \ldots, x_l]) - \tilde{g}_{u_j}([x_2, \ldots, x_l])| + \epsilon \\
\leq \;& C \cdot \epsilon \cdot (1 - C^l)/(1 - C) + \epsilon = \epsilon \cdot (1 - C^{l+1})/(1 - C)
\end{aligned}$$

by induction. □

Assume U is bounded and $\{u_1, \ldots, u_n\}$ is an ϵ-covering of U. Then we can obviously approximate every function $f : X \times U \to U$ by a function the codomain of which is contained in $\{u_1, \ldots, u_n\}$ only. Hence we can cover every set of functions mapping to U by functions with images in the discrete set $\{u_1, \ldots, u_n\}$. Since the initial contexts in the above Lemma can be chosen as elements of the set $\{u_1, \ldots, u_n\}$ and the approximations in the cover only yield values in that set, we obtain as an immediate corollary that a finite set U is sufficient for internal processing:

Corollary 3.5 *Assume $\mathcal{F}$ and U are as above. Assume $\epsilon > 0$; assume $\{u_1, \ldots, u_n\}$ is an ϵ-covering for U. Denote by $Q : U \to \{u_1, \ldots, u_n\}$ the quantization mapping, which maps a value $u \in U$ to the nearest value u_i (some fixed nearest u_i if this is not unique). Denote by f^Q the composition $f^Q(x, u) = Q(f(x, u))$. Then*

$\{(\tilde{f}^Q)_{u_j} \mid f \in \mathcal{F}, j = 1, \ldots, n\}$ *forms an* $\epsilon/(1-C)$*-covering of* $\tilde{\mathcal{F}}$ *and* $\{(\tilde{f}^Q)^L_{u_j} \mid f \in \mathcal{F}, j = 1, \ldots, n\}$ *forms an* $\epsilon/(1-C)$*-covering of* $\tilde{\mathcal{F}}_L$*. Note that these functions use values of* $\{u_1, \ldots, u_n\}$ *only.*

Proof. Note that $\{f^Q \mid f \in \mathcal{F}\}$ constitutes an external ϵ-covering of $\mathcal{F}$ because the outputs are changed by at most ϵ. Moreover, $\{u_1, \ldots, u_n\}$ constitutes an ϵ-cover of U by assumption. Hence we can apply Lemma 3.4. As a consequence, the recursive classes $\{(\tilde{f}^Q)_{u_j} \mid f \in \mathcal{F}, j = 1, \ldots, n\}$ and $\{(\tilde{f}^Q)^L_{u_j} \mid f \in \mathcal{F}, j = 1, \ldots, n\}$ form $\epsilon/(1-C)$-covers of $\tilde{\mathcal{F}}$ and $\tilde{\mathcal{F}}_L$, respectively. □

Hence, we can substitute every recursive function where the transition constitutes a contraction by a function which uses only a *finite* number of different values for U and a *finite* memory length. Depending on the form of X, the internal values for U can be substituted by values consisting of sequences in X. More precisely we get the following result:

Corollary 3.6 *For every* $\epsilon > 0$*, function* $f : X \times U \to U$ *with bounded domain* U *such that* f *is a contraction with parameter* C*, initial context* $y \in U$*, we can find a memory length* L*, a finite set* $\{x_1, \ldots, x_m\}$ *in* X*, and a quantization* $Q : X \to \{x_1, \ldots, x_m\}$ *such that the following holds: there exists a function* $g : \{x_1, \ldots, x_m\}^{\leq L} \to U$ *such that*

$$|\tilde{f}_y(s) - g(T_L(\tilde{Q}(s)))| \leq \epsilon$$

where $\tilde{Q}(s)$ *denotes the element-wise application of* Q *to the sequence* s*. If* X *is finite,* Q *can be chosen as the identity.*

Proof. As a consequence of Lemma 3.2 and Corollary 3.5 we can approximate $\tilde{f}_y$ by a function $\tilde{h}^L_{u_{i_0}}$ which uses only a finite number of values $\{u_1, \ldots, u_n\}$ in U and a finite memory length L. Define equivalence classes on X via the definition $x \sim x'$ for

x, $x' \in X$ iff $h(x, u_j) = h(x', u_j)$ for all $u_j \in \{u_1, \ldots, u_n\}$. This yields only a finite number of equivalence classes. Choose a fixed value x_j from each equivalence class. Define $Q : x \mapsto x_j$, such that x lies in the equivalence class of x_j. Then the choice $g := \tilde{h}^L_{u_{i_0}}$ yields the desired approximation. The same choice is possible if X itself is finite and Q is the identity. □

This result tells us that we can substitute recursive maps with compact codomain and contractive transition functions by definite memory machines if the input alphabet is finite. Otherwise, the input alphabet can be quantized accordingly such that an equivalent definite memory machine with a finite number of different input symbols and the same behavior can be found. In case of RNNs, further processing is added to the recursive computation, i.e. we are interested in functions of the form $h \circ \tilde{f}_y$, where h is some function which maps the processed sequence to the desired output, but itself does not contribute to the recursive computation. If h is continuous, obviously similar approximation results can be obtained, since we can simply combine the above approximation g with h. Note that h is then uniformly continuous on the compact domain U. Therefore, approximation of $\tilde{f}_y$ by the function $g \circ T_L \circ \tilde{Q}$ up to ϵ yields approximation of $h \circ \tilde{f}_y$ with $h \circ g \circ T_L \circ \tilde{Q}$ up to a value which depends on the modulus of continuity of g.

We are here interested in recurrent neural networks and their connection to definite memory machines. We assume that $X = \mathbb{R}^{n_1}$ and $U = \mathbb{R}^{n_2}$ are real vector spaces equipped with the maximum norm which we denote by $|\cdot|_\infty$.

Definition 3.7 *A* recurrent network *(RNN) computes a function of the form* $h \circ \tilde{f}_y : (\mathbb{R}^{n_1})^* \to \mathbb{R}^{n_3}$*, where* $h : \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$ *and* $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_2}$ *are of the form*

$$h(u) = M_1 \cdot \boldsymbol{\sigma}(M_2 \cdot u + d), \quad f(x, u) = \boldsymbol{\sigma}(W_1 \cdot x + W_2 \cdot u + b)$$

where $M_1 \in \mathbb{R}^{n_3 \times n_4}$, $M_2 \in \mathbb{R}^{n_4 \times n_2}$, $W_1 \in \mathbb{R}^{n_2} \times \mathbb{R}^{n_1}$, *and* $W_2 \in \mathbb{R}^{n_2} \times \mathbb{R}^{n_2}$ *are matrices,* $d \in \mathbb{R}^{n_4}$, $b \in \mathbb{R}^{n_2}$, *and* $\boldsymbol{\sigma}$ *denotes the component-wise application of a transition function* $\sigma : \mathbb{R} \to \mathbb{R}$.

In the above definition, h constitutes a so-called feedforward network with one hidden layer which maps the recursively processed sequences to the desired outputs, f defines the recurrent part of the network. Popular choices for σ are the hyperbolic tangent $\tanh$ or the logistic function $\mathrm{sgd}(x) = (1 + \exp(-x))^{-1}$. We can apply the above results if the transition function f constitutes a contraction and the internal values are contained in a bounded set. Under these circumstances, RNNs simply implement a definite memory machine and can be substituted by a fractal prediction machine, as an example. We first refer to the case where σ is the identity.

Definition 3.8 *A function* $f : X \to Y$ *is* Lipschitz continuous *with parameter* K *with respect to metrics* $|\cdot|$ *on* X *and* Y *if* $|f(x) - f(x')| \leq K \cdot |x - x'|$ *for all* $x, x' \in X$.

Lemma 3.9 *The function* $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_2}$, $(x, u) \mapsto W_1 \cdot x + W_2 \cdot u + b$ *as above is Lipschitz continuous with respect to the second input parameter* u *and the maximum norm* $|\cdot|_\infty$ *with parameter* K *where* $K = n_2 \cdot \max |w_{jk}|$ *and* w_{jk} *are the components of matrix* W_2. *The mapping is a contraction for* $\max |w_{jk}| < 1/n_2$.

Proof. We find

$$\begin{aligned} & |(W_1 \cdot x + W_2 \cdot u + b) - (W_1 \cdot x + W_2 \cdot u' + b)|_\infty \\ = \; & |W_2 \cdot (u - u')|_\infty = \max_j |\textstyle\sum_k w_{jk}(u_k - u'_k)| \\ \leq \; & n_2 \cdot \max_{jk}(|w_{jk}| \cdot |u_k - u'_k|) \leq n_2 \cdot \max_{jk} |w_{jk}| \cdot |u - u'|_\infty \end{aligned}$$

Obviously, a contraction is obtained for $\max |w_{jk}| < 1/n_2$. □

Hence if we can in addition make sure that the image of the transition function is bounded, e.g. due to the fact that $b = 0$ and the elements of input sequences are contained in a compact set, we can approximate the above recursive computation by a definite memory machine. The necessary length of the sequences L depends on the degree of the contraction, i.e. the magnitude of the weights and the desired accuracy of the approximation.

Note the following simple observation which allows us to obtain results for non-linear activation functions σ: If f_1 and f_2 are Lipschitz continuous with constants K_1 and K_2, respectively, the composition $f_1 \circ f_2$ is Lipschitz continuous with constant $K_1 \cdot K_2$. Hence arbitrary activation functions σ which are Lipschitz continuous with parameter K lead to contractive transition functions f if the weights W_2 fulfill $\max |w_{jk}| < 1/(n_2 \cdot K)$. In particular, differentiable activation functions σ such that σ' can be uniformly limited by a constant K are Lipschitz continuous with parameter K. Hence they yield to contractions. Since many standard activation functions like the hyperbolic tangent or the logistic activation function fulfill this property and map, moreover, to a limited domain such as $(0, 1)$ or $(-1, 1)$ only, we have finally obtained the result that recurrent networks with small weights can be approximated arbitrarily well with definite memory machines.

Note that, before training, the weights are usually initialized with small random vectors. If they are initialized in a small enough domain, e.g. their absolute value is not larger than, e.g. $4/n_2$ if the logistic function σ is used, they have contractive transition functions, i.e. act like definite memory machines. This argumentation implies that through the initialization recurrent networks have an architectural bias towards definite memory machines. Feedforward neural networks with time window input constitute a popular alternative method for sequence processing (Sejnowski and Rosenberg, 1987; Waibel et.al., 1989). Since a finite time window corresponds to a finite memory of definite memory machines, recurrent networks are biased towards these successful alternative training methods where the size of the time window is not fixed a-priori.

We add a remark on recurrent neural networks used for the approximation of probability distributions as proposed for example in (Bengio and Frasconi, 1996).

Definition 3.10 *A* probabilistic recurrent network *computes a function of the form $h \circ \tilde{f}_y : (\mathbb{R}^{n_1})^* \to \mathbb{R}^{n_3}$ where $h : \mathbb{R}^{n_2} \to \{x \in \mathbb{R}^{n_3} \mid \sum_{i=1}^{n_3} x_i = 1, x_i \geq 0\}$, and $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_2}$ is of the form*

$$f(x, u) = \boldsymbol{\sigma}(W_1 \cdot x + W_2 \cdot u + b)$$

where $W_1 \in \mathbb{R}^{n_2} \times \mathbb{R}^{n_1}$ and $W_2 \in \mathbb{R}^{n_2} \times \mathbb{R}^{n_2}$ are matrices, $b \in \mathbb{R}^{n_2}$, and $\boldsymbol{\sigma}$ denotes the component-wise application of a transition function $\sigma : \mathbb{R} \to \mathbb{R}$. $h \circ \tilde{f}_y$ defines a conditional probability distribution on a set $\{x_1, \ldots, x_{n_3}\}$ of cardinality n_3 given a sequence $s \in (\mathbb{R}^{n_1})^*$ via the choice $p(x_i|s) = h_i(\tilde{f}_y(s))$, where h_i denotes the i th output component of h.

Note that elements in $\{x \in \mathbb{R}^{n_3} \mid \sum_{i=1}^{n_3} x_i = 1, x_i \geq 0\}$ correspond to probability distributions over n_3 discrete elements. Hence a probabilistic recurrent network induces a distribution for the next symbol given a sequence s if the output components of the network are interpreted as a probability distribution over the alphabet. Usually, h consists of a linear function combined possibly by component-wise nonlinear transformation and followed by normalization. In (Bengio and Frasconi, 1996), the outputs of f are normalized, too, such that the intermediate values can be interpreted as a probability distribution on a finite set of hidden states and training can be performed for example with a generalized EM algorithm (Neal and Hinton, 1998). Note that the above approximation results can be transferred immediately to a probabilistic network if the transition function is a contraction and the set of intermediate values is bounded. Here we obtain the result that the function which maps a sequence to the next symbol probabilities can be approximated by a function implemented by a definite memory machine. Such probabilistic recurrent networks can be approximated arbitrarily well by FOMMs.

Note that approximation of probability distributions p and q on the finite set of possible events $\{x_1, \ldots, x_n\}$ up to degree ϵ here means that $|p(x_i) - q(x_i)| \leq \epsilon$ for all i. Based on this estimation, and assuming $q(x_i) \geq \delta > 0$, we can obtain a bound on the Kullback-Leibler divergence $\sum_i p(x_i) \log(p(x_i)/q(x_i))$, which is smaller than $\log((\delta + \epsilon)/\delta)$. This term becomes arbitrarily small if ϵ approaches 0.

One can obtain explicit bounds on the weights W_2 such that the contraction condition is fulfilled as above if f consists of a linear function and a component-wise nonlinearity like the logistic function. Assumed a normalization of the outputs is added in the recursive steps of f, too, as proposed in (Bengio and Frasconi, 1996) then alternative bounds on the magnitudes of the weights can be derived using the fact that the mapping $x \mapsto x/\|x\|$ is Lipschitz continuous with parameter $2/\epsilon$ for $\|x\| > \epsilon$ where $\|\cdot\|$ denotes the Euclidean metric.

4 Every DMM can be implemented by a contractive recurrent network

We have seen that, loosely speaking, recurrent networks with contractive transition functions implement at most DMMs (or FOMMs). Here we establish the converse direction, every DMM or FOMM, respectively, can be approximated arbitrarily well by a recurrent network with contractive transition function. Note that several possibilities of injecting finite automata or finite state machines (and thus also definite memory machines) into recurrent networks have been proposed in the literature, e.g. (Carrasco and Forcada, 2001; Frasconi et.al., 1995; Omlin and Giles, 1996a; Omlin and Giles, 1996b). Since these methods deal with general finite automata, the transition function of the constructed RNNs is not a contraction and does not fulfill the condition of small weights.

We assume that $\Sigma = \{x_1, \ldots, x_n\}$ is a finite alphabet. We are interested in processing of sequences over Σ. We assume that input sequences in Σ^* are presented to a recurrent network in a unary way, i.e. x_i corresponds to the unit vector $e_i \in \mathbb{R}^n$ with entry 1 at position i and 0 for all other positions. Denote by $Q : \Sigma \to \mathbb{R}^n$ the coding $x_i \mapsto e_i$. Denote by $\tilde{Q}(s)$ the element-wise application of Q to the entries of a sequence s. We assume that the nonlinearity σ used in the network is of sigmoid type, i.e. it has a specific form which is fulfilled for popular activation functions like the hyperbolic tangent. More precisely, we assume the properties that σ is a monotonically increasing and continuous function which has finite limits $\lim_{x\to\infty} \sigma(x) \neq \lim_{x\to-\infty} \sigma(x)$.

Lemma 4.1 *Assume σ is a monotonously increasing, continuous function with finite limits $\lim_{x\to-\infty} \sigma(x) =: l < r := \lim_{x\to\infty} \sigma(x)$. Assume $g : \Sigma^* \to \Sigma$ is computed by a DMM, i.e. there exists some $L \in \mathbb{N}$ such that $g(s) = g(T_L(s))$ for all $s \in \Sigma^*$. Assume $C \in (0,1)$. Then there is $m \in \mathbb{N}$ and $y \in \mathbb{R}^m$, so that we can find functions $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ and $h : \mathbb{R}^m \to \mathbb{R}^n$ of a recurrent network $h \circ \tilde{f}_y$, such that $h \circ \tilde{f}_y(\tilde{Q}(s)) = Q(g(s))$, for all $s \in \Sigma^*$ and f is a contraction with parameter C with respect to the second argument.*

Proof. Assume $n = |\Sigma|$. We choose $m = L \cdot n$ and let y be the origin. First, we define the transition function f of the recursive part of the form $f(x, u) = \boldsymbol{\sigma}(W_1 \cdot x + W_2 \cdot u + b)$. We start constructing the recursive part for the case $\sigma(0) = 0$: Because of the continuity of σ, we can find some positive δ such that f constitutes a contraction with respect to the second argument and inputs in $[l, r]^m$ with parameter C

if the absolute value of all coefficients in W_2 is at most δ. We can think of the outputs of $\tilde{f}_y$ as L blocks of n coefficients. We will define f such that, given the input sequence s, coefficient j of block i is larger than 0 iff the element i of the input sequence s is x_j and it is 0, otherwise. For this purpose, denote by index : $\{1,\ldots,L\} \times \{1,\ldots,n\} \to \{1,\ldots,m = L\cdot n\}$ a fixed bijective mapping. We enumerate the coefficients of W_2 by tuples $(\text{index}(i,j), \text{index}(k,l))$ where i, k are in $\{1,\ldots,L\}$, j, l are in $\{1,\ldots,n\}$. We enumerate the entries of W_1 by tuples $(\text{index}(i,j), k)$ where $i \in \{1,\ldots,L\}$, j, k are in $\{1,\ldots,n\}$. We choose $b = 0$, and all entries of W_1 and W_2 as 0 except for $(W_1)_{(\text{index}(1,j),j)} = 1$ for $j \in \{1,\ldots,n\}$, and $(W_2)_{(\text{index}(i,j),\text{index}(i+1,j))} = \delta$ for $i < L$, $j \in \{1,\ldots,n\}$. This choice has the effect that the actual input is stored in the first block and the inputs of the last $L-1$ steps, which can be found in the first to $L-1$st block in the previous step, are transferred to the second to Lth block. Hence the last L values of an input sequence are stored in the activations of the network. Precisely all different prefixes of length L of sequences yield unique outputs of $\tilde{f}_y$.

Assume that $\sigma(0) \neq 0$. Then we can construct a recursive part of a network which uniquely encodes prefixes of length L as follows: The function σ_t with $\sigma_t(x) = \sigma(x) - \sigma(0)$ is a monotonously increasing and continuous function with finite limits and the property $\sigma_t(0) = 0$. Hence we can use σ_t to construct a recursive part of a network $\tilde{f}_y$ with the above properties, where the transition function is of the form $\boldsymbol{\sigma}_t(W_1 \cdot x + W_2 \cdot u + b)$. We find for all sequences s the equality $\tilde{f}_y(s) = \tilde{g}_z(s) - \boldsymbol{\sigma}(0)$ where $g(x,u) = \boldsymbol{\sigma}(W_1 \cdot x + W_2 \cdot u + b - W_2 \cdot \boldsymbol{\sigma}(0))$, $z = W_2 \cdot \boldsymbol{\sigma}(0)$, and $\boldsymbol{\sigma}(0)$ is the vector with components $\sigma(0)$. Obviously, $\tilde{f}_y(s)$ encodes the prefixes of length L uniquely iff $\tilde{f}_y(s) + \boldsymbol{\sigma}(0)$ encodes the prefixes uniquely, hence $\tilde{g}_z$ constitutes a recursive part of a network with the desired properties and activation function σ.

Hence we obtain a unique encoding of the last L entries of the sequence through the recursive transformation in both cases. It follows immediately from well-known approximation or interpolation results, respectively, for feedforward networks that some h can be found which maps the outputs of $\tilde{f}_y$ to the desired values (Hornik, 1993; Hornik, Stinchcombe, White, 1989; Sontag, 1992). h can be chosen as a feedforward network with one hidden layer. $\square$

Note that we can obtain the further extension of the above result that every DMM can be approximated by a RNN of the above form with arbitrarily small weights in the recursive and feedforward part. We have already seen, that the weights in W_2 can be chosen arbitrarily small. Choosing the entry in W_1 as δ instead of 1 does not change the argumentation. Moreover, the universal approximation capability of feedforward networks also holds for analytic σ (e.g. the hyperbolic tangent) if the bias and the weights are chosen from an arbitrarily small open interval (Hornik, 1993).[3] Hence we can limit the weights in the feedforward part, too.

The above result can be immediately transferred to approximation results for the probabilistic counterparts of DMMs. Note that even if the output of the recursive part is in addition normalized as in (Bengio and Frasconi, 1996), the fact that all sequences of length at most L are mapped to unique values through the recursive computation is not altered. Hence we can find an appropriate h which outputs the probabilities of the next symbol in a sequence. h can be computed by a feedforward network followed by normalization. Therefore, FOMM can obviously be approximated (even precisely interpolated) by probabilistic recurrent networks up to any desired degree, too.

[3]Note that the number of hidden neurons in h might increase if the weights are restricted. For unlimited weights, we can bound the number of hidden neurons in h by the finite number of possible different outputs of $\tilde{f}_y$, which depends (exponentially) on m and L only.

5 Learnability

We have shown that RNNs with small weights and DMMs implement the same function classes if restricted to a finite input set. The respective memory length sufficient for approximating the RNN depends on the size of the weights. Since initialization of RNNs often puts a bias towards DMMs or their probabilistic counterpart and FLMMs possess efficient training algorithms like fractal prediction machines, the latter constitute a valuable alternative to standard RNNs for which training is often very slow (Ron, Singer, Tishby, 1996; Tiňo and Dorffner, 2001).

Another point which makes DMMs and recurrent networks with small weights attractive concerns their generalization ability. Here we first introduce several definitions: Statistical learning theory provides one possible way to formalize the learnability or generalization ability of a function class. Assume $\mathcal{F}$ is a function class with domain X and codomain Y. We assume in the following that every function or set which occurs is measurable. Assume $|\cdot|$ defines a metric on Y. A learning algorithm for $\mathcal{F}$ outputs a function $g \in \mathcal{F}$ given a finite set of examples $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$ for an unknown function $f \in \mathcal{F}$. Generalization ability of the algorithm refers to the fact that the functions f and g approximately coincide on all possible inputs if they coincide on the given finite set of examples. Denote by $\mathcal{P}$ the set of probability measures on X and by P its elements. P^m is the product measure induced by P on X^m. The distance between functions f and g with respect to P is denoted by

$$d_P(f,g) = \int_X |f(x) - g(x)| d_P(x)\,.$$

The empirical distance between f and g given $\vec{x} = (x_1, \ldots, x_m) \in X^m$ refers to the quantity

$$\hat{d}_m(f,g,\vec{x}) = \sum_i |f(x_i) - g(x_i)|/m$$

which is obtained if the distance of f and g is evaluated at m given data points. The aim in the general training scenario is to minimize the distance between the function to be learned, say f, and the function obtained by training, say g. Usually, this quantity is not available because the function to be learned is unknown. Hence standard training often minimizes the empirical error between f and g on a given set $\vec{x}$ of training examples. A justification of this principle can be established if the empirical distance is representative of the real distance. Since the function obtained by training usually depends on the whole training set (and hence the error on one training example does not constitute an independent observation), a uniform convergence in (high) probability of the empirical distance $\hat{d}_m(f, g, \vec{x})$ for arbitrary functions f and g and sample $\vec{x}$ is established. Generalization then means that $\hat{d}(f, g, \vec{x})$ and $d_P(f, g)$ nearly coincide for large enough m uniformly for f and g.

Definition 5.1 *$\mathcal{F}$ fulfills the* distribution independent uniform convergence of empirical distances property (UCED-property) *if for all $\epsilon > 0$*

$$\lim_{m\to\infty} \sup_{P\in\mathcal{P}} P^m(\vec{x} \mid \exists f, g \in \mathcal{F}\, |d_P(f, g) - \hat{d}_m(f, g, \vec{x})| > \epsilon) = 0\,.$$

Since one can think of f as the function to be learned and of g as the output of the learning algorithm, this property characterizes the fact that we can find prior bounds (independent of the underlying probability) on the necessary size of the training set, such that every algorithm with small training error yields good generalization with high probability. For short, the UCED-property is one possible way of formalizing the generalization ability. Note that the framework tackled by statistical learning theory usually deals with a more general scenario, the so-called agnostic setting (Haussler, 1992). There, the function class $\mathcal{F}$ used for learning need not contain the unknown function which is to be learned, and the error is measured by a general loss function.

Valid generalization then refers to the property of uniform convergence of empirical means (UCEM) of a class associated to $\mathcal{F}$ via the loss function. However, under several conditions on $\mathcal{F}$ and the loss function, learnability of this associated class can be related to learnability of $\mathcal{F}$ (Anthony and Bartlett, 1999; Vidyasagar, 1997). For simplicity, we will only investigate the UCED property of recurrent networks with small weights. The following is a well known fact:

Lemma 5.2 *Finite function classes fulfill the UCED-property.*

Assume Σ is a finite alphabet and $\mathcal{F}$ is the class of functions from Σ^* to Σ which can be computed by a DMM with fixed finite memory length L. Then $\mathcal{F}$ fulfills obviously the UCED-property because the function class is finite. Hence DMMs with fixed length L can generalize, when provided with enough training data.

Assume $\mathcal{F}$ is the function class which is given by the functions computed by all recurrent neural networks as defined in Definition 3.7 where the dimensionalities n_i and σ are fixed, but the entries of the matrices can be chosen arbitrarily and arbitrary computation accuracy is assumed. Then $\mathcal{F}$ does not possess the UCED-property as shown in (Bartlett, Long, Williamson, 1994; Hammer, 1997; Koiran and Sontag, 1997), for example. Hence general recurrent networks with no further restrictions do not yield valid generalization in the above sense unlike fixed length DMM. One can prove weaker results for recurrent networks, which yield bounds on the size of a training set such that valid generalization holds with high probability as derived in (Hammer, 2000; Hammer, 1999), for example. However, these bounds are no longer independent of the underlying (unknown) distribution of the inputs. Training of general RNNs may need in theory an exhaustive number of patterns for valid generalization and certain underlying input distributions. One particularly 'bad' situation is explicitly constructed in (Hammer, 1999) where the number of examples necessary for valid generalization increases more than polynomially in the required accuracy. Naturally, restriction of the search space e.g. to finite automata with a fixed number of states offers a method to establish prior bounds on the generalization error of RNNs. Moreover, in practical applications, because of the computation noise and finite accuracy, the effective VC dimension of RNNs is finite. Nevertheless, more work has to be done to formally explain, why neural network training often shows good generalization ability in common training scenarios. Here we offer a theory for initial phases of RNN training by linking RNNs with small weights to the definite memory machines.

Note that RNNs with small weights and a finite input set approximately coincide with DMMs with fixed length, where the length depends on the size of the weights. Hence we can conclude that RNNs with a priori limited small weights and a finite input alphabet possess the UCED property contrarious to general RNNs with arbitrary weights and finite input alphabet. That means, the architectural bias through the initialization emphasizes a region of the parameter search space where the UCED property can be formally established. We will show in the remaining part of this section that an analogous result can be derived for recurrent networks with small weights and arbitrary real-valued inputs. This shows that function classes given by RNNs with a priori limited small weights possess the UCED property in contrast to general RNNs with arbitrary weights and infinite precision.

We consider function classes $\mathcal{F}$ with domain X and codomain equal to $[0, 1]$ equipped with the maximum norm. Moreover, we assume that the constant function 0 is contained in $\mathcal{F}$, too. Then alternative characterization for the UCED property can be found in the literature which relate the generalization ability to the capacity of the function class. Appropriate formalizations of the term 'capacity' are as follows:

Definition 5.3 *Assume $\mathcal{F}$ is a function class. Let $\epsilon > 0$. The* external covering number

$L(\epsilon, \mathcal{F}, |\cdot|_\infty)$ denotes the size of the smallest external ϵ-covering of $\mathcal{F}$ with respect to the metric $|\cdot|_\infty$. $L(\epsilon, \mathcal{F}, |\cdot|_\infty)$ is infinite if no finite external covering of $\mathcal{F}$ exists.

*The ϵ-*fat shattering dimension *$\mathcal{VC}_\epsilon(\mathcal{F})$ of $\mathcal{F}$ is the largest size (possibly infinite) of a set of points $\{x_1, \ldots, x_n\}$ in X which can be shattered with parameter ϵ. Shattering with parameter ϵ means that real values $r_1, \ldots, r_n$ exist such that for each function $d : \{x_1, \ldots, x_n\} \to \{-1, 1\}$ some function $f \in \mathcal{F}$ exists with $|f(x_i) - r_i| \geq \epsilon$ and $(f(x_i) - r_i) \cdot d(x_i) > 0$.*

Both, the covering number and the fat-shattering dimension measure the richness of $\mathcal{F}$: the number of essentially different functions up to ϵ, or the number of points where a rich behavior can be observed within the function class, respectively. Assume $\vec{x} = (x_1, \ldots, x_m) \in X^m$ is a vector. Denote the restriction of $\mathcal{F}$ to $\vec{x}$ by $\mathcal{F}|_{\vec{x}} = \{g : \{x_1, \ldots, x_m\} \to Y \mid \exists f \in \mathcal{F}\ g(x_i) = f(x_i)\}$. Proofs for the following alternative characterizations of the UCED property can be found in (Anthony and Bartlett, 1999; Bartlett, Long, Williamson, 1994; Vidyasagar, 1997):

Lemma 5.4 *The following characterizations are equivalent for a function class $\mathcal{F}$ with codomain $[0, 1]$ which contains the constant function 0:*

- *$\mathcal{F}$ fulfills the UCED-property.*
- $\displaystyle\lim_{m\to\infty} \sup_{P \in \mathcal{P}} \frac{E_{P^m}(\log_2 L(\epsilon, \mathcal{F}|_{\vec{x}}, |\cdot|_\infty))}{m} = 0 \quad \forall \epsilon > 0\,.$
- *$\mathcal{VC}_\epsilon(\mathcal{F})$ is finite for every $\epsilon > 0$.*

E_{P^m} denotes expectation with respect to $\vec{x} = \{x_1, \ldots, x_m\}$. Furthermore, the estimation

$$L(\epsilon, \mathcal{F}|_{\vec{x}}, |\cdot|_\infty) \leq 2 \left(\frac{4\,m}{\epsilon^2}\right)^{d \log_2(12\,m/(d\epsilon))}$$

holds for every $\vec{x} = \{x_1, \ldots, x_m\}$ where $d = \mathcal{VC}_{\epsilon/4}(\mathcal{F})$.

Using this alternative characterization, we can prove that recurrent networks with small weights and arbitrary inputs fulfill the UCED property, too. Denote by $\mathcal{G} \circ \mathcal{F}$ the class of compositions $\{g \circ f \mid g \in \mathcal{G}, f \in \mathcal{F}\}$ for function classes $\mathcal{F}$ and $\mathcal{G}$ with common domain of $\mathcal{G}$ and codomain of $\mathcal{F}$, respectively.

Lemma 5.5 *Assume $C_1 \in (0,1)$ and $C_2 > 0$ are fixed. Assume U is a bounded set. Assume $\mathcal{F}$ is a function class with domain $X \times U$ and codomain U such that every function in $\mathcal{F}$ is a contraction with parameter C_1 with respect to the second argument. Assume $\mathcal{G}$ is a function class with domain U and codomain $[0,1]$ such that every function in $\mathcal{G}$ is Lipschitz continuous with parameter C_2. Then the function class $\mathcal{G} \circ \tilde{\mathcal{F}}$ fulfills the UCED property if the function class $\mathcal{G} \circ \tilde{\mathcal{F}}_L$ fulfills the UCED property for every $L \in \mathbb{N}$.*

Proof. Assume $\epsilon > 0$. Assume $\vec{s} = (s_1, \ldots, s_m)$ is a vector of m sequences over X. Because of Lemma 3.2 and because every $g \in \mathcal{G}$ is Lipschitz continuous with parameter C_2, we can find some L such that every $g \circ \tilde{f}_y$ in $G \circ \tilde{\mathcal{F}}$ deviates from $g \circ \tilde{f}_y^L$ in $\mathcal{G} \circ \tilde{\mathcal{F}}_L$ by at most $\epsilon/2$ for all input sequences s. Hence

$$L(\epsilon, \mathcal{G} \circ \tilde{\mathcal{F}}|_{\vec{s}}, |\cdot|_\infty) \leq L(\epsilon/2, \mathcal{G} \circ \tilde{\mathcal{F}}_L|_{\vec{s}}, |\cdot|_\infty) = L(\epsilon/2, \mathcal{G} \circ \tilde{\mathcal{F}}_L|_{T_L(\vec{s})}, |\cdot|_\infty)$$

where $T_L(\vec{s})$ denotes the application of the truncation T_L to every s_i in $\vec{s}$. Hence we can bound the term $L(\epsilon, \mathcal{G} \circ \tilde{\mathcal{F}}|_{\vec{s}}, |\cdot|_\infty)$ for every $\vec{s}$ by $2\,(16m/\epsilon^2)^{d\,\log_2(24\,m/(d\epsilon))}$ where $d = \mathcal{VC}_{\epsilon/8}(\mathcal{G} \circ \tilde{F}_L)$ is finite because $\mathcal{G} \circ \tilde{F}_L$ fulfills the UCED property. Hence the quotient $E_{P^m}(\log_2\, L(\epsilon, \mathcal{G} \circ \tilde{\mathcal{F}}|_{\vec{s}}, |\cdot|_\infty)/m)$ becomes arbitrarily small for large m, every $\epsilon > 0$, and every $P \in \mathcal{P}$. $\square$

As a consequence, standard recurrent networks with small weights in the recursive part such that the transition function constitutes a contraction and with limited weights in the feedforward part such that Lipschitz continuity is guaranteed fulfill the UCED property: the function classes $\mathcal{G} \circ \tilde{\mathcal{F}}_L$ from the above proof correspond in this case to simple feedforward networks with more than one hidden layer which have a finite fat-shattering dimension and therefore fulfill the UCED property for standard activation functions like the hyperbolic tangent (Baum and Haussler, 1989; Karpinski and Macintyre, 1995).

An alternative proof for the UCED property given real valued inputs can be obtained relating $\mathcal{G} \circ \tilde{\mathcal{F}}$ to the class $\mathcal{G} \circ \mathcal{F}$, which is non recursive, as follows:

Lemma 5.6 *Assume $C_1 \in (0,1)$ and $C_2 > 0$ are fixed. Assume U and X are bounded sets. Assume $\mathcal{F}$ is a function class with domain $X \times U$ and codomain U such that every function in $\mathcal{F}$ is a contraction with parameter C_1 with respect to the second argument. Assume that in addition, every function in $\mathcal{F}$ is Lipschitz continuous with parameter C_1'. Assume $\mathcal{G}$ is a function class with domain U and codomain $[0,1]$ such that every function in $\mathcal{G}$ is Lipschitz continuous with parameter C_2. Then $\mathcal{G} \circ \tilde{\mathcal{F}}$ fulfills the UCED property if $\mathcal{G} \circ \mathcal{F}$ does.*

Proof. Note that

$$L(\epsilon, \mathcal{G} \circ \tilde{\mathcal{F}}|_{\vec{s}}, |\cdot|_\infty) \leq L(\epsilon, \mathcal{G} \circ \tilde{\mathcal{F}}, |\cdot|_\infty)$$

for all $\vec{s}$. Because of Lemma 3.4 and the Lipschitz continuity of all functions in $\mathcal{G}$ with parameter C_2, we find

$$L(\epsilon, \mathcal{G} \circ \tilde{\mathcal{F}}, |\cdot|_\infty) \leq L(\delta, \mathcal{G} \circ \mathcal{F}, |\cdot|_\infty)$$

for some $\delta > 0$ which depends on ϵ, C_1, and C_2. Because X and U are bounded, we can find a finite covering $\vec{xu} := \{(x_1, u_1), \ldots, (x_c, u_c)\}$ with parameter $\delta/(3\, C_1' C_2)$ of the set $X \times U$. Denote by $N(\delta, \mathcal{H}, |\cdot|_\infty)$ the smallest size of a δ-covering of a function class $\mathcal{H}$ with respect to the metric $|\cdot|_\infty$ such that all functions in the cover are contained in $\mathcal{H}$ itself. Because of the triangle inequality, the estimation

$$N(2\delta, \mathcal{H}, |\cdot|_\infty) \leq L(\delta, \mathcal{H}, |\cdot|_\infty) \leq N(\delta, \mathcal{H}, |\cdot|_\infty)$$

follows immediately for every function class $\mathcal{H}$. Now we find

$$L(\delta, \mathcal{G} \circ \mathcal{F}, |\cdot|_\infty) \leq N(\delta/3, \mathcal{G} \circ \mathcal{F}|_{\vec{xu}}, |\cdot|_\infty)$$

because of the following: choose for $(x, u) \in X \times U$ a closest (x_i, u_i) in $\vec{xu}$ and for $g \circ f$ in $\mathcal{G} \circ \mathcal{F}$ a function $g_j \circ f_j$ corresponding to a function in $N(\delta/3, \mathcal{G} \circ \mathcal{F}|_{\vec{xu}}, |\cdot|_\infty)$ such that the distance to $g \circ f$ is minimum on $\vec{xu}$. Then

$$\begin{aligned}
& |g(f((x,u))) - g_j(f_j((x,u)))| \\
\leq\ & |g(f((x,u))) - g(f((x_i,u_i)))| \\
& + |g(f((x_i,u_i))) - g_j(f_j((x_i,u_i)))| \\
& + |g_j(f_j((x_i,u_i))) - g_j(f_j((x,u)))| \\
\leq\ & C_1' C_2 \delta/(3C_1' C_2) + \delta/3 + C_1' C_2 \delta/(3C_1' C_2) = \delta\,.
\end{aligned}$$

Since the UCED property holds for $\mathcal{G} \circ \mathcal{F}$, we can bound the quantity

$$N(\delta/3, \mathcal{G} \circ \mathcal{F}|_{\vec{xu}}, |\cdot|_\infty) \leq L(\delta/6, \mathcal{G} \circ \mathcal{F}|_{\vec{xu}}, |\cdot|_\infty) \leq 2 \left(\frac{4 \cdot 36 \cdot c}{\delta^2} \right)^{d \log_2(12 \cdot 6 \cdot c/(d\delta))}$$

where c only depends on δ, C_1', and C_2, and $d = \mathcal{VC}_{\delta/24}(\mathcal{G} \circ \mathcal{F})$ is finite because of the UCED property of $\mathcal{G} \circ \mathcal{F}$. Hence the quantity $L(\epsilon, \mathcal{G} \circ \tilde{\mathcal{F}}|_{\vec{s}}, |\cdot|_\infty)$ can be limited by a finite number for fixed ϵ. Therefore, the UCED property of $\mathcal{G} \circ \tilde{F}$ follows. $\square$

Hence the additional property that the set X is bounded allows us to connect the learnability of recurrent architectures with contractive transition function to the learnability of the corresponding non-recursive transition function.

We conclude this section by performing two experiments which give some hints on the effect of small recurrent weights on the generalization ability. We use RNNs for sequence prediction for two sequences: the Mackey-Glass time series with dynamic $dx/d\tau = b \cdot x(\tau) + a \cdot x(\tau - d)/(1 + x(\tau - d))^{10}$ with $a = 0.2$, $b = -0.1$, and $d = 17$ (Mackey and Glass, 1977). The task for the RNN is to predict the related discrete-time series S_1: $x(3 \cdot t) - 0.4$ for $t = 0, 1, 2 \ldots$ with values in $(0, 1)$ and quasiperiodic behavior. In addition, we consider the Boolean time series $x(t) = x(t-1) \vee \neg x(t-2)$, $t = 2, 3, \ldots$ with $x(0) = x(1) = 0$. We introduce observation noise by flipping each entry with probability 0.1. The second task for the RNN is to predict the related sequence S_2: $0.8 \cdot x(t) + 0.1 \in (0, 1)$. For both tasks we generated 2000 training instances and 1334 test instances. We are interested in the generalization ability of networks which fit these sequences with different sizes of the weights on recurrent connections. A small network with 2 hidden neurons and the logistic activation function is used for prediction. To separate effects of RNN training from the effect of small weights, we use no training algorithm but consider only randomly generated RNNs. For different sizes of the recurrent weights we compare the test set error of the fraction of 100000 randomly generated networks which have the mean absolute training error smaller than 0.15. Hence 'training' consists in our case only of accepting or rejecting networks based their training set performance. To separate the positive effect of weight restriction for the recurrent dynamic from the benefit of small weights for feedforward networks (Bartlett, 1997) we initialize the output weights and the weights connected to the input randomly in the interval $(-4, 4)$ in all cases. The recurrent connections are randomly initialized in the interval $(-B, B)$ and B is varied from 0.1 to 10.1. Note that the recurrent mapping need no longer be a contraction for $B \geq 2$. The relationship between the fraction of randomly generated networks with training error smaller than 0.15 and the size B of recurrent connections is presented in Fig. 1.

Fig. 2 shows the mean absolute training and test set error for the two tasks. For comparison, the constant mapping to the expected value for S_1 has an error 0.19, and the default classification according to the majority in S_2 gives the error 0.24. In our experiments, the mean error on the training set remains almost constant whereas the mean error on the test set increases for increasing size of the recurrent weights. Note that this increase is smooth, hence no dramatic decrease of the generalization ability can be observed if non contractive recursive mapping might occur, i.e. the weights come from an interval with $B > 2$. For S_1, the test error becomes as large as 0.24 which almost corresponds to random guessing. The test error approximates 0.2 for large weights for S_2 which is still better than a majority vote, hence generalization can here be observed even for large recurrent weights. The generalization error, i.e. the absolute distance of the training and test set errors, is depicted in Fig. 3. The mean generalization error reaches values of 0.1 and 0.06, respectively, for large weights and is much smaller for small weights. As shown in Fig. 4, the percentage of networks with low training error and test error comparable to the training error decreases with increasing radius B of the size of recurrent connections. For small recurrent weights, nearly 100% or 65%, respectively, of the networks with small training error have a test error of at most 0.17, whereas the percentage decreases to 40% or 30%, respectively, for increasing size of the weights of recurrent connections. These experiments indicate that in this setting the generalization ability of RNNs without further restrictions is better for smaller *recurrent* weights. However, particularly 'bad' situations which could occur in theory for non-contractive transition function cannot be observed for randomly generated networks: the increase of the test error is smooth with respect to the size of the weights. Note that no training has been taken into account in this setting. It is very likely that training adds additional regularization to the RNNs. Hence randomly generated networks might not be representative for typical training outputs

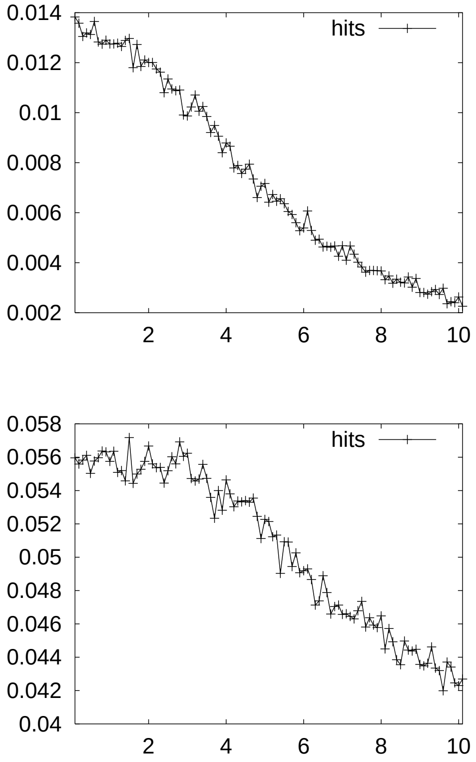


Figure 1: Fraction (max 1) of randomly generated networks with training error smaller than 0.15 for S_1 (top) and S_2 (bottom), respectively, depending on the size B of recurrent connections. Among 100000 randomly generated networks, we obtain about 200 up to 1400 hits for S_1, and 4200 up to 4500 hits for S_2.

and the generalization error of trained networks with possibly large recurrent weights might be much better than the reported results. Further investigation is necessary to answer the question whether initialization with small weights has a positive effect on the generalization ability in realistic training settings; but such experiments are beyond the scope of this article.

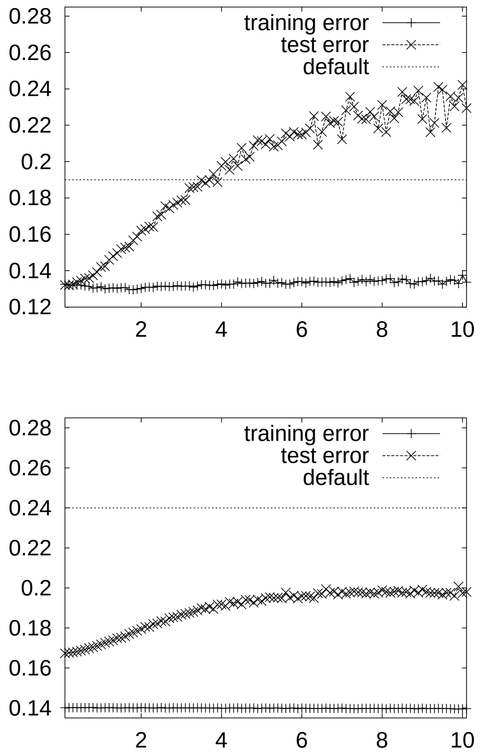


Figure 2: Mean training and test error of RNNs with randomly initialized weights on the two time series S_1 (top) and S_2 (bottom). The x-axes shows the radius B of the interval in which recurrent weights have been chosen. The default horizontal line shows the error of constant prediction of the expected value for S_1 (left) and the error of constant classification to the majority class for S_2 (right). The default models represent naive memoryless predictors.

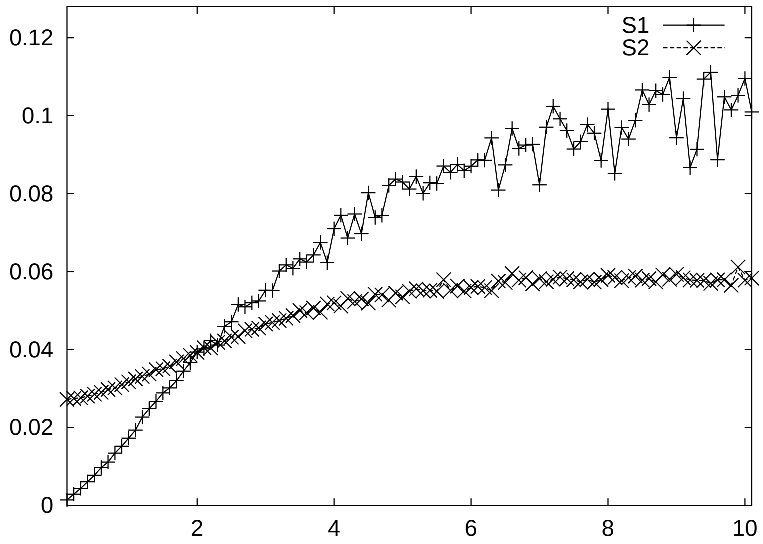
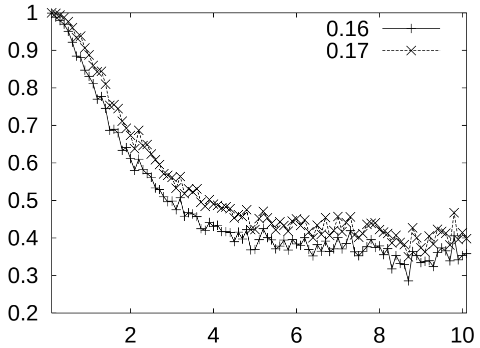


Figure 3: Mean generalization error of RNNs for S_1 and S_2, respectively, depending on the size of the recurrent connections.

6 Discussion

We have rigorously shown that initialization of recurrent networks with small weights biases the networks towards definite memory models. This theoretical investigation supports our previous experimental findings (Tiňo, Čerňanský, Beňušková, 2002a; Tiňo, Čerňanský, Beňušková, 2002b). In particular, by establishing simulation of definite memory machines by contractive recurrent networks and vice versa, we proved an equivalence between problems that can be tackled with recurrent neural networks with small weights and definite memory machines. Analogous results for probabilistic counterparts of these models follow from the same line of reasoning and show the equivalence of fixed order Markov models and probabilistic recurrent networks with small weights.



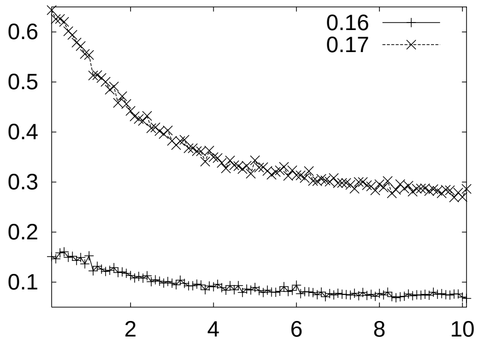


Figure 4: Percentage of networks with test error smaller than 0.16 and 0.17, respectively, among all randomly generated networks with training error at most 0.15 and various sizes of the recurrent connections for S_1 (top) and S_2 (bottom).

We conjecture that this architectural bias is beneficial for training: it biases the architectures towards a region in the parameter space where simple and intuitive behavior can be found, thus guaranteeing initial exploration of simple models where prior theoretical bounds on the generalization error can be derived. A first step into this direction has been investigated in this article, too, within the framework of statistical learning theory. It can be shown that unlike general recurrent networks with arbitrary precision, recurrent networks with small weights allow bounds on the generalization ability which depend only on the number of parameters of the network and the training set size, but neither on the specific examples of the training set, nor on the input distribution. These bounds hold even if infinite accuracy is available and inputs may be real-valued. The argumentation is valid for every fixed weight restriction of recurrent architectures which guarantees that the transition function is a contraction with a given fixed contraction parameter. Note that these learning results can be easily extended to arbitrary contractive transition functions with no a-priory known constant through the luckiness-framework of machine learning (Shawe-Taylor et.al., 1998). The size of the weights or the parameter of the contractive transition function, respectively, offers a hierarchy of nested function classes with increasing complexity. The contraction parameter controls the structural risk in learning contractive recurrent architectures.

Note that although the VC-dimension of RNNs might become arbitrarily large in theory if arbitrary inputs and weights are dealt with, it is not likely to occur in practice: it is well known that lower bounds on the VC dimension need high precision of the computation and the bounds are effectively limited if the computation is disrupted by noise. The articles (Maass and Orponen, 1998; Maass and Sontag, 1999) provide bounds on the VC dimension in dependence on the given noise. Moreover, the problem of long-term dependencies likely restricts the search space for RNN training to comparably simple regions and yields a restriction of the effective VC-dimension which can be observed when training RNNs. In addition, the choice of the error function (e.g. quadratic error) puts an additional bias towards training and might constitute a further limitation of the VC-dimension achieved in practice. Hence the restriction to small weights in initial phases of training which has been investigated in this article constitutes one aspect among others which might account for good generalization ability of RNNs in practice. We have derived explicit prior bounds on the generalization ability for this case and we have established an equivalence of the dynamics to the well understood dynamics of DMMs. As a consequence small weights constitute *one sufficient* condition for valid generalization of RNNs, among other well known guarantees. The concrete effect of the small weight restriction and other aspects as mentioned above has to be further investigated in experiments. Two preliminary experiments for time series prediction have shown that small recurrent weights have a beneficial effect on the generalization ability of RNNs. Thereby, we tested randomly generated RNNs in order to rule out numerical effects of the training algorithm. We varied only the size of the recurrent connections to rule out the beneficial effect of small weights in standard feedforward networks (Bartlett, 1997). For randomly chosen small networks, the percentage of networks with small weights which generalize well to unseen examples is larger than the percentage among RNNs initialized with larger weights. Thereby, the increase of the generalization error is smooth compared to the size of the weights, i.e. networks with particularly bad generalization ability for larger weights can hardly be found by random choice. Since efficient training of RNNs is still an open problem, we did not incorporate the effects of training in our experiments which might introduce additional regularization into learning such that the effect of small weights might vanish. Nevertheless, restriction to the smallest possible weights for a given task seems one possible strategy to achieve valid generalization and we have derived explicit mathematical bounds for this setting.

In (Tiňo, Čerňanský, Beňušková, 2002a; Tiňo, Čerňanský, Beňušková, 2002b) we extracted from recurrent networks predictive models that operated on the network dynamics. The networks were first randomly initialized with small weights and then input-driven with training sequences. The resulting clusters of recurrent activations were labeled with (cluster conditional) empirical next-symbol distributions calculated on the training stream. Hence training takes place in one epoch on the output level only. No optimization of the representation of the sequences in the hidden neurons was done but the sequence representation provided by the randomly initialized recurrent network dynamic was used. By performing experiments on symbolic sequences of various memory and subsequence structure we showed that predictive models extracted from these networks where internal representation of the sequences is given by randomly initialized (with small weights) networks achieved performance very similar to that of variable memory length Markov models (VLMM). Obviously, recurrent networks have a potential to outperform finite memory models and they indeed did so after a careful and (often rather lengthy) training process. But, since the predictive models extracted from networks with untrained recurrent connections initialized with small weights[4] correspond to VLMM, depending on the nature of the data, the performance gain resulting from training the appropriate recursive representation in the hidden neurons of recursive neural networks can be quite small. In (Tiňo, Čerňanský, Beňušková, 2002b) we argue that to appreciate how much information has really been induced during the training, the network performance should always be compared with that of VLMM and predictive models extracted before training as the 'null' base mod-

[4]training is performed in one epoch to adjust hidden-layer-to-output mapping

els.

Interestingly enough, the contractive nature of recurrent networks initialized with small weights enables us to perform a rigorous fractal analysis of the state-space representations induced by such networks. The first results in that direction can be found in (Tiňo and Hammer, 2002).

References

Anthony, M., and Bartlett, P.L. (1999). *Neural Network Learning: Theoretical Foundations.* Cambridge University Press.

Baldi, P., Brunak, S., Frasconi, P., Pollastri, G., and Soda, G. (2001). Bidirectional dynamics for protein secondary structure prediction. R. Sun, C.L. Giles (eds.), *Sequence Learning: Paradigms, Algorithms, and Applications*, pp. 80-104, Springer.

Bartlett, P.L. (1997). For valid generalization, the size of the weights is more important than the size of the network. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, Volume 9. The MIT Press, pp. 134-141.

Bartlett, P.L., Long P., and Williamson, R. (1994). Fat-shattering and the learnability of real valued functions. In *Proceedings of the 7th ACM Conference on Computational Learning Theory*, pp. 299-310.

Baum, E.B., and Haussler, D. (1989). What size net gives valid generalization? *Neural Computation*, 1(1):151-165.

Bengio, Y. and Frasconi, P. (1996). Input/output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231-1249.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157-166.

Bühlmann, P., and Wyner, A.J. (1999). Variable length Markov chains. *Annals of Statistics*, 27:480-513.

Carrasco, R.C., and Forcada, M.L. (2001). Simple strategies to encode tree automata in sigmoid recursive neural networks. *IEEE Transactions on Knowledge and Data engineering*, 13(2):148-156.

Christiansen, M.H., and Chater,N. (1999). Towards a connectionist model of recursion in human linguistic performance. *Cognitive Science*, 23:157-205.

D.S. Clouse, C.L. Giles, B.G. Horne, and G.W. Cottrell. Time-Delay Neural Networks: Representation and Induction of Finite State Machines. *IEEE Transactions on Neural Networks*, 8(5):1065, 1997.

Elman, J., Bates, E., Johnson, M., Karmiloff-Smith, A., Parisi, D., and Plunkett, K. (1996). *Rethinking Innateness: a Connectionist Perspective on Development.* MIT Press, Cambridge.

Frasconi, P., Gori, M., Maggini, M., and Soda, G. (1995). Unified integration of explicit rules and learning by example in recurrent networks. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):313-332.

Funahashi, K., and Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 12:831-864.

Giles, C.L., Lawrence, S., and Lin, T. (1995). Learning a class of large finite state machines with a recurrent neural network. *Neural Networks*, 8(0):1359-1365.

Giles, C.L., Lawrence, S., and Tsoi, A.C. (1997). Rule inference for financial prediction using recurrent neural networks. *Proceedings of the Conference on Computational Intelligence for Financial Engineering*, pp.253-259, New York City, NY.

Guyon, I., and Pereira, F. (1995). Design of a linguistic postprocessor using variable memory length Markov models. *Proceedings of International Conference on Document Analysis and Recognition*, pp.454-457, Montreal, Canada, IEEE Computer Society Press.

Hammer, B. (2001). Generalization ability of folding networks. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):196-206.

Hammer, B. (1999). On the learnability of recursive data. *Mathematics of Control, Signals, and Systems*, 12:62-79.

Hammer, B. (1997). On the generalization of Elman networks. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicaud, editors, *Artificial Neural Networks – ICANN'97*. Springer, pp. 409-414.

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

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735-1780.

Hornik, K. (1993). Some new results on neural network approximation. *Neural Networks*, 6:1069-1072.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359-366.

Karpinski, M., and Macintyre, A. (1995). Polynomial bounds for the VC dimension of sigmoidal neural networks. In *Proceedings of the 27th annual ACM Symposium on the Theory of Computing*, pp. 200-208.

Kohavi, Z. (1978). *Switching and finite automata*. McGraw-Hill.

Kohonen, T. (1997). *Self-Organizing Maps*. Springer.

Koiran, P., and Sontag, E.D. (1997). Vapnik-Chervonenkis dimension of recurrent neural networks. In *Proceedings of the 3rd European Conference on Computational Learning Theory*, pp. 223-237.

Kolen, J.F. (1994). Recurrent networks: state machines or iterated function systems? *Proceedings of the 1993 Connectionist Models Summer School,* pp.203-210, Lawrence Erlbaum Associates, Hilsdale, NJ.

Kolen, J.F. (1994). The origin of clusters in recurrent neural state space. *Proceedings of the 1993 Connectionist Models Summer School,* pp.508-513, Lawrence Erlbaum Associates, Hilsdale, NJ.

Krogh, A. (1997). Two methods for improving performance of a HMM and their application for gene finding. *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology*, pp.179-186, Menlo Park, CA, AAAI Press.

Laird, P., and Saul, R. (1994). Discrete sequence prediction and its applications. *Machine Learning*, 15: 43-68.

Maass, W., and Orponen, P. (1998). On the effect of analog noise in discrete-time analog computation. *Neural Computation*, 10(5):1071-1095.

Maass, W., and Sontag, E.D. (1999). Analog neural nets with Gaussian or other common noise distributions cannot recognize arbitrary regular languages. *Neural Computation*, 11:771-782.

Mackey, M.C., and Glass, L. (1977). Oscillations and chaos in physiological control systems. *Science*, 197:287-289.

Nadas, J. (1984). Estimation of probabilities in the language model of the IBM speech recognition system. *IEEE Transactions on ASSP*, 4:859-861.

Neal, R., and Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants, in M. Jordan (ed.), Learning in Graphical Models, Kluwer, pp.355-368.

Omlin, C.W., and Giles, C.L. (1996). Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM*, 43(6):937-972.

Omlin, C.W., and Giles, C.L. (1996). Stable encoding of large finite-state automata in recurrent networks with sigmoid discriminants. *Neural Computation*, 8:675-696.

Robinson, T., Hochberg, M., and Renals, S. (1996). The use of recurrent networks in continuous speech recognition. C.-H. Lee and F.K. Song (eds.), *Advanced Topics in Automatic Speech and Speaker Recognition*, chapter 7, Kluwer.

Ron, D., Singer, Y., and Tishby, N. (1996). The power of amnesia. *Machine Learning*, 25:117-150.

Sejnowski, T., and Rosenberg, C. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145-168.

Shawe-Taylor, J., Bartlett, P.L., Williamson, R., and Anthony, M. (1998). Structural risk minimization over data dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5).

Siegelmann, H.T., and Sontag, E.D. (1994). Analog computation, neural networks, and circuits. *Theoretical Computer Science*, 131:331-360.

Siegelmann, H.T., and Sontag, E.D. (1995). On the computational power of neural networks. *Journal of Computer and System Sciences*, 50:132-150.

Sontag, E.D. (1998). VC dimension of neural networks. In C. Bishop, editor, *Neural Networks and Machine Learning*. Springer, pp. 69-95.

Sontag, E.D. (1992). Feedforward nets for interpolation and classification. *Journal of Computer and System Sciences*, 45:20-48.

Sun, R. (2001), Introduction to sequence learning. R. Sun, C.L. Giles (eds.), *Sequence Learning: Paradigms, Algorithms, and Applications*, pp. 1-10, Springer.

Tiňo, P., Čerňanský, M., and Beňušková, L. (2002). Markovian architectural bias of recurrent neural networks. P. Sinčák, J. Vaščák, V. Kvasnička and J. Pospichal (eds.), *Intelligent Technologies - Theory and Applications. Frontiers in AI and Applications* – 2nd Euro-International Symposium on Computational Intelligence, pp. 17-23, IOS Press, Amsterdam.

Tiňo, P., Čerňanský, M., and Beňušková, L. (2002). Markovian architectural bias of recurrent neural networks. Technical Report NCRG/2002/008, NCRG, Aston University, UK.

Tiňo, P., and Dorffner, G. (2001). Predicting the future of discrete sequences from fractal representations of the past. *Machine Learning*, 45(2):187-218.

Tiňo, P., and Hammer, B. (2002). Architectural bias of recurrent neural networks - fractal analysis. J. R. Dorronsoro (ed.), *Int. Conf. on Artificial Neural Networks (ICANN 2002)*, pp. 1359-1364, Springer.

Tiňo, P., and Šajda, J. (1995). Learning and extracting initial Mealy machines with a modular neural network model. *Neural Computation*, 4:822-844.

Vidyasagar, M. (1997). *A Theory of Learning and Generalization*. Springer.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328-339.