ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 357-368

Tutorial: Perspectives on Learning with RNNs

B. Hammer[1] and Jochen J. Steil[2]

[1] Dept. of Math./Comp.Science, University of Osnabrück, Germany
hammer@informatik.uni-osnabrueck.de,
[2] Neuroinformatics Group, University of Bielefeld, Germany
jsteil@techfak.uni-bielefeld.de, www.jsteil.de

Abstract. We present an overview of current lines of research on learning with recurrent neural networks (RNNs). Topics covered are: understanding and unification of algorithms, theoretical foundations, new efforts to circumvent gradient vanishing, new architectures, and fusion with other learning methods and dynamical systems theory. The structuring guideline is to understand many new approaches as different efforts to regularize and thereby improve recurrent learning. Often this is done on two levels: by restricting the learning objective by constraints, for instance derived from stability conditions or weight normalization, and by imposing architectural constraints as for instance local recurrence.

1 Introduction

In many fields, the sequential nature of either discrete or continuous inputs and outputs makes RNNs attractive for the general tasks of sequence prediction, sequence transduction, or sequence generation. Typical well known applications include time-series prediction and generation, speech recognition, adaptive control, or biological modeling. While many architectures ranging from fully connected to partially or locally RNNs have been proposed, there are common problems: to analyze the resulting dynamical network behavior and to devise learning algorithms in order to cope with the complexity introduced by these dynamics. Thus, intensive research on dynamical network properties and the corresponding learning algorithms has achieved a good theoretical grounding of many popular algorithms.

In this contribution, we review recent developments in the theoretical foundations of recurrent learning algorithms, new algorithms, and architectures to enforce that a specified desired dynamic behavior of the RNN is achieved. Hereby, we focus on the concept of regularization and follow the hypothesis that many new architectures and methods can be understood as restricting the huge search spaces for weight optimization, either by constraining the learning algorithm or the network architecture. As the RNN field is fast and dynamically changing, we restrict ourselves to recent work of the last few years. Some topics not covered are: RNNs for optimization, stability theory of RNNs, oscillator and spiking networks.

In Section 2 we ask how, what, and why to learn in order to summarize work in unification of algorithms, knowledge on network capabilities, to sketch a profile of

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 357-368

recent applications, and to discuss RNN learning from the point of view of learning theory. In Section 3 we review recent algorithms and architectures under the regularization paradigm, including stability issues, connections to automata, and locally RNNs. Section 4 highlights approaches not relying on gradient learning and connections to other techniques like self organized learning and dynamical systems theory.

2 Understanding and unification of algorithms

Common RNN models have the following discrete or continuous dynamics, respectively, or can be simulated within these dynamics (Kremer [2001]):

$$\mathbf{x}(k+1) = \sigma(\mathsf{W} \cdot (\mathbf{x}(k), \mathbf{i}(k))), \quad \mathbf{o}(k) = \mathsf{V} \cdot \mathbf{x}(k), \quad k \in \mathbb{N} \tag{1}$$

$$\dot{\mathbf{x}}(k) = \sigma(\mathsf{W} \cdot (\mathbf{x}(k), \mathbf{i}(k))), \quad \mathbf{o}(k) = \mathsf{V} \cdot \mathbf{x}(k), \quad k \geq 0 \tag{2}$$

starting with a fixed initial state $\mathbf{x}(0)$ of the network. Hereby, $\mathbf{x}(k)$ denotes the state of the network at time k, $\mathbf{i}(k)$ the respective vector of inputs, V and W are weight matrices, σ denotes the element-wise application of the node output function (usually a sigmoid such as the hyperbolic tangent), and $\mathbf{o}(k)$ is the vector of outputs at time k.

2.1 How to learn: understanding algorithms

A fundamental aspect of recurrent learning is to understand how algorithms solve the credit assignment problem, i.e. how the network prediction error is used to change the weights. The main body of such research is centered around the two key concepts of (global) gradient decent (see Atiya and Parlos [2000] and Pearlmutter [1995] for reviews) and (local) Hebbian learning. Although a lot of flavors of the basic rules were proposed, progress has been made to rederive many of these algorithms in common frameworks. Here, we concentrate on two such approaches to gain fundamental insight into the concepts before reviewing some extensions and new architectures in later sections. In the following, we assume for simplicity that the outputs $\mathbf{o}(k)$ are part of the state variables $\mathbf{x}(k)$ such that only W has to be determined.

For gradient methods, Atiya and Parlos [2000] show that many algorithms can be derived from a constrained optimization problem for the accumulated quadratic error

$$\text{Minimize } E \doteq \frac{1}{2} \sum_{k=1}^{K} \sum_{i \in O} (x_i(k) - d_i(k))^2, \tag{3}$$

$$\text{subject to } g(k+1) \doteq \sigma(\mathsf{W}\mathbf{x}(k)) - (\mathbf{x}(k+1)) = 0, \tag{4}$$

where $d_i(k)$ is the target output at time k for a number of output units from the set O of output nodes. Denote by $\mathbf{w}$ the vector of weights in W and by $\mathbf{x}$ the vector of activations $\mathbf{x}(k)$. Taking the derivatives of (3) and (4) with respect to $\mathbf{w}$ we get

$$\frac{dE}{d\mathbf{w}} = \frac{\partial E}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{w}}, \quad \frac{dg}{d\mathbf{w}} = \frac{\partial g}{\partial \mathbf{w}} + \frac{\partial g}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{w}} = 0 \tag{5}$$

and elimination of $\partial \mathbf{x} / \partial \mathbf{w}$ yields the gradient of the error function

$$\frac{dE}{d\mathbf{w}} = -\frac{\partial E}{\partial \mathbf{x}} \left(\frac{\partial g}{\partial \mathbf{x}} \right)^{-1} \frac{\partial g}{\partial \mathbf{w}}. \tag{6}$$

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 357-368

With respect to the matrix equation (6), backpropagation through time (BPTT) computes $\partial E/\partial \mathbf{x}\,(\partial g/\partial \mathbf{x})^{-1}$ first and postmultiplies $\partial g/\partial \mathbf{w}$ whereas real time recurrent learning (RTRL) starts with $(\partial g/\partial \mathbf{x})^{-1}\,\partial g/\partial \mathbf{w}$. Atiya and Parlos further introduce new batch and on-line algorithms based on the idea to interchange the role of the network states $\mathbf{x}$ and the weights $\mathbf{w}$. The key element is to compute the gradient of E with respect to $\mathbf{x}$ first and then adjust the weights accordingly to generate the desired change in $\mathbf{x}$. Their work shows, that from the point of view of unification of the standard algorithms the formulation of learning as an optimization theoretic problem is very fruitful. It is intuitive that from this starting point also other non-gradient optimization approaches can be employed, which are reviewed in Section 4.1.

The main alternative to gradient algorithms is to use the locally available pre- and postsynaptic states x_i^- and x_i^+. In this case the generalized recirculation algorithm introduced by O'Reilly [1996] provides a common framework to understand the popular delta rule, contrastive Hebbian learning, and classical recirculation as different approximations of Almeida-Pineda recurrent backpropagation when employing the cross-entropy error function as in Hinton [1989].

2.2 What to learn: dynamical capabilities

From well known approximation results for feedforward networks (Scarselli and Tsoi [1998]) it follows that discrete-time RNNs can approximate every given dynamics with continuous transition function up to any desired accuracy on compact sets and a finite time horizon (Sontag [1992]). Funahashi and Nakamura [1993] extended this result to the continuous case. Hence for every given finite set of input/output pairs in (3, 4) a network which produces this behavior can be found in principle.

Naturally, the long-term behavior of dynamic systems is also of particular interest. With respect to their capability, discrete-time RNNs can be related to classical symbolic sequence processing tools: Kilian and Siegelmann [1996] prove the Turing-universality of discrete time RNNs and Siegelmann and Sontag [1994] relate RNNs to the even more powerful Boolean circuits. However, these proofs of the capacity of RNNs do not yield benefits for practical learning tasks. On the other hand, connections to finite automata come together with mechanisms for rule extraction and injection which can increase the performance of gradient based learning algorithms (Carrasco et al. [2000], Omlin and Giles [1996a,b]).

It is also well known that even in two neuron RNNs there exists the whole spectrum of qualitatively different trajectory behavior from stable, over periodic and quasi-periodic to a chaotic regime. An analytical proof was given by Wang [1991], the location of stable states has been studied in depth in Tiňo et al. [2001], and input dependent bifurcation curves have been analyzed by Hoppensteadt and Izhikevich [1997] in the continuous and Haschke et al. [2001] in the discrete case. Many numerical studies support these results for larger networks, see e.g. the review in Dayhoff et al. [1999].

2.3 Why to learn: some recent applications

In time series prediction, a variety of benchmarks and competitions (McNames et al. [1999]) from dynamical systems, weather forecast, or the financial data exist (Aussem [1999]). Relevant real world financial forecasting data sets are often small and noisy

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 357-368

and it is not clear which trends are predictable at all (de Bodt et al. [2001]). Hence regularization methods involve e.g. ideas from symbol processing (Giles et al. [2001]), dimensionality reduction, or restricted training as proposed by (Zimmermann,Neuneier,-Grothmann) and (Zimmermann,Tietz,Grothmann) in this volume.

Discrete or continuous time RNNs can be found in the context of dynamical systems and control: adaptive filtering (Parlos et al. [2001]), noise reduction, and system identification (Giannakakis [1999]), channel equalization and signal separation in mobile communication (Benson and Carrasco [2001]). RNNs are designed for simulating simple linear control in biological systems (Ferree and Lockery [1998]), complex biotechnological processes (Patnaik [1997]) or for adaptive control of technical systems (Baratti et al. [2000]). A variety of RNN architectures can be found in robot control, e.g.for adaptive gait generation (Kolb et al. [1998]), generation of periodic movements (Ruiz et al. [1998]), or autonomous adaptive behavior (Ziemke [2000]).

Recurrent architectures are used in object recognition by integrating time-context information in a sequence of visual images (Ziemke et al. [1997]). Wersing et al. [2001] use the dynamics of the competitive layer model, a recurrent linear threshold network, for complex binding and grouping tasks such as contour extraction, object localization, and cell segmentation (Nattkemper et al. [2000]).

RNNs have been successfully applied for speech recognition, prediction and classification (Trentin and Giuliani [2001],Baltersee and Chambers [1998], Lawrence et al. [2000]), to name just a few recent approaches. They have also been trained on artificial grammars, where the internal processing dynamics can be understood more easily (Omlin and Giles [1996a]). Remarkably, RNNs can also recognize some context sensitive languages, as demonstrated in two different approaches within this volume by (Bodén) and (Gers,Pérez-Ortiz,Eck,Schmidhuber), respectively.

A couple of generalizations of RNNs have been proposed to process structured data, among them complex systems for propositional reasoning like LISA (Hummel and Holyoak [1997]). Other approaches only slightly enlarge RNNs such that the structure can be processed or identified more easily, e.g. bicausal RNNs (Baldi et al. [1999]), recursive networks (Frasconi et al. [1997], see also the article by (Hammer,Micheli,Sperduti) in this volume), simple synchrony networks (Lane and Henderson [2001]), or recurrent autoassociative networks (Stoianov [2000]). Applications for these architectures can be found in language parsing (Costa et al., Lane and Henderson [2001], see also the article by (Henderson) in this volume), chemistry (Bianucci et al. [2000]), or bioinformatics (Baldi et al. [1999]).

2.4 How to learn (revisited): theoretical issues

Two fundamental questions arise with respect to the theory of learning: can the generalization ability of RNNs be guaranteed? Which is the complexity of training?

Possible formalizations of the generalization ability are offered by the notion of PAC-learnability and identification in the limit, respectively (Anthony and Bartlett [1999], Natarajan [1991]). The statistical notion of PAC-learnability ensures that, given enough training patterns, the error of the trained network is with high probability small for all inputs in a set of high probability. It is often tested limiting the VC-dimension, which for RNNs depends on the maximum length of input sequences (Koiran and Sontag [1997]) and is infinite for priorly unlimited inputs. Hence unlike standard feedforward networks, distribution independent bounds on the generalization

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 357-368

error of recurrent architectures cannot be found within the PAC framework. Weaker bounds are obtained if additional information on the input distribution or the training is available (Hammer [2001]). Alternatively, the architecture can be restricted, e.g. to contractive transition functions (Hammer and Tiňo [2002]).

Identification in the limit ensures that the underlying regularity can be precisely identified for all possible inputs after presentation of enough training examples (Natarajan [1991]). This characterization is suited e.g. for grammar inference, where the RNN should adequately model the behavior for inputs of arbitrary length. It can be shown that in the Chomsky hierarchy only regular and context free languages can be identified in the limit from positive and negative training examples. If only positive examples are available, as is often the case in language processing, the class of learnable languages reduces to finite languages. Hence language processing is an inherently difficult task for all learning mechanisms including neural networks (Kremer [2000]). However, natural languages as well as recurrent architectures stratified e.g. according to the number and size of weights are orthogonal to the Chomsky hierarchy such that investigating the learnability of important settings remains a challenging task.

The complexity of training can be treated within classical complexity theory, but several results show that optimizing the training error is NP-hard even for feedforward networks (DasGupta and Hammer [2000]). Further, regularization methods which yield feasible feedforward network training need not have the same effect for their recurrent counterparts (Suykens and Vandewalle [2000]). On the other hand, weight sharing induces further structure into the training problem which might prove beneficial in specific training scenarios (Frasconi et al. [2000]). In summary, the basic complexity of RNN training is widely unsolved.

One specific numerical problem can be observed for gradient based training algorithms of RNNs: the so-called problem of long-term dependencies (Bengio et al. [1994], Hochreiter and Schmidhuber [1997]). This refers to the fact that information has to be latched through several recursive time steps within network training. Unfortunately, the error signals vanish while propagating through the nonlinear network and hence gradient based training methods without using regularization are likely to fail.

3 Regularization

Regularization techniques are a major source of new ideas and approaches which either incorporate additional terms or constraints into the optimization procedure or put certain restrictions on the architecture of the network. In the field of feedforward networks, regularization usually aims at improving generalization (Wahba [1995], Leung et al. [2001]). In the recurrent domain, the goals of regularization are much more widespread and include the incorporation of dynamical properties like stability or, on the architectural level, of semantic restrictions like approximation within a certain class of automata. A striking example of regularization with respect to function is provided by Bakker et al. [2000]. The idea is to stop training if the error is low and an additional test is passed, which measures the network's sequence generating capabilities with respect to the training data sequence.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 357-368

3.1 Incorporation of stability

RNNs can in general have arbitrary unstable behavior and in the course of learning stability may be not preserved (Pearlmutter [1995]). Therefore several approaches link results from stability theory directly to the learning process. For input-output mappings, Suykens et al. [2000] use results from robust stability theory for closed loop systems to derive constraints in form of linear matrix inequalities (LMI). These impose and maintain local stability at the origin (also Suykens et al. [1997] and the references therein). A less direct approach is taken by Steil and Ritter [1999a], Steil [2001], where global and local LMI conditions are given to determine a stable region in which arbitrary weight changes can be tolerated. Learning is then restricted to that region (Steil and Ritter [1999b]). Both methods increase the complexity and the numerical burden of the weight optimization process and improvements towards computational efficiency remain a challenge.

Jin and Gupta [1999] enforce stability of equilibria to store patterns by incorporation of additional terms in the error functional and the incremental weight update formulas, which are derived from matrix conditions for global stability. Park et al. [1999] obtained suitable weights by solving a set of LMIs, which encode that a number of patterns have to be stored as stable states of a BSB (brain state in a box) associative memory. In the competitive layer model for sensory segmentation and feature binding, stable states represent optimal bindings of the input data into segmented groups (Wersing et al. [2001]). The corresponding weights, which express mutual compatibility between pairs of inputs, can be obtained by supervised learning using inequalities derived from stability conditions (Wersing [2001]).

3.2 Automata

In grammatical inference tasks and language processing, symbolic approaches and RNNs constitute two alternative paradigms. Hence restriction of RNNs to the dynamics of symbolic mechanisms such as finite automata with a restricted number of states offers a natural possibility of regularization. If no further knowledge is available, RNNs are initialized with small random weights. It can be shown that this initialization introduces an architectural bias towards very simple mechanisms, finite memory machines, such that good generalization ability can be expected and structural differentiation can be observed even prior to training (Hammer and Tiňo [2002], Kolen [1994]). The simplest class in the Chomsky hierarchy are finite automata. Various different approaches show how automata rules can be encoded in a recurrent architecture (Carrasco et al. [2000], Frasconi et al. [1995]) as an alternative initialization method for RNNs. This explicit rule integration allows a very fast training since the inserted rules need only be refined (Frasconi et al. [1995], Sundareshan and Condarcure [1998], Omlin and Giles [1996b]). Extraction of automata rules from a trained neural network based on clustering of the state space is described by Omlin and Giles [1996a]. The extracted rules are subject to less noise and offer a better generalization (Giles et al. [2001]). Note that inserted automata rules might get lost during this procedure unless training involves additional constraints on the weights (Kremer et al. [1998]). For natural language processing, context free and context sensitive aspects play a role. Various approaches successfully learned some context free or context sensitive languages (Rodriguez et al. [1999], Hochreiter and Schmidhuber [1997], see

the articles of (Bodén) and (Gers,Pérez-Ortiz,Eck,Schmidhuber) in this volume). It is not yet possible to extract rules for these languages, however, one can identify typical behavior such as counting mechanisms (Rodriguez et al. [1999]). Hence inserting and extracting rules constitutes a valuable contribution to efficient training, although the capacity of rule sets in existing approaches is still very limited.

3.3 Locally recurrent architectures

Another possibility of regularization is offered by architectural constraints. Various methods are reviewed e.g. by Kremer [2001]. Time delay networks restrict to a finite time window in all layers such that standard backpropagation together with weight sharing can be used for training. NARX networks use only recursive values in a time window of the input and output layer, such that an open loop approximation transforms training to an essentially feedforward task. LSTM networks combine recurrent linear units with various nonlinear control gaits. Error backpropagation is truncated except for the linear units where a constant error signal flow can be guaranteed and hence information be latched. The resulting algorithm is local in time and space but modifications which hurt these conditions might prove beneficial as demonstrated by (Gers,Pérez-Ortiz,Eck,Schmidhuber) in this volume. Locally RNNs restrict recurrent connections to self-connections of the units such that backpropagation formulas can be computed in a particularly efficient way or, alternatively, iterative construction of the architecture and units as in recurrent cascade correlation can be achieved. Alternatively, local feedback can take the form of IIR, FIR, or γ-filters, which store in each step an appropriately weighted history of previous activations of the neurons. Campolucci et al. [1999] offer a self-contained derivation of the BPTT formulas adapted to these specific architectures. Furthermore, an online modification of the standard learning rule is proposed which covers as a special case various previous approaches: updates can be made causal via truncating the dependence of weight changes at time step k on the error signals such that they come from a fixed finite number of further time steps. Note that the capacity of local RNNs is strictly smaller than the capacity of general RNNs (Frasconi and Gori [1996]). Therefore the appropriate choice of the architecture for the specific learning task is crucial for success. It is valuable to combine efficient local training algorithms for networks with restricted architectures with global optimization algorithms for finding the optimal architecture. For instance, Baratti et al. [2000] obtain very good results by a combination of Tabu search with the above mentioned modification of BPTT.

4 Other techniques

Naturally, the combination of RNNs with other techniques from computational neuroscience offers a variety of alternative learning methods. For example, Trentin and Giuliani [2001] combine RNNs in a mixture of experts architecture for speaker independent language recognition. Thereby, the credit assignment problem for recursive data is solved by embedding the sequences in a finite dimensional vector space and successive application of standard techniques for the reduced descriptions. The idea of self-organization constitutes another valuable paradigm for data processing and training which can also be transferred to recurrent data structures: based on metrical

information, a low-dimensional topological representation of the data is found with Hebbian learning in the self-organizing map (SOM). Thus the generalization ability is automatically taken into account due to the learning procedure. Some approaches transfer unsupervised learning algorithms to sequential data (Barreto et al. [2001] for an overview). A general framework for several recurrent SOM architectures is proposed in the article of (Hammer,Micheli,Sperduti) in this volume. Further methods involve global optimization techniques or explicit construction and modification of desired dynamic behavior.

4.1 Optimization without the Gradient

Starting from the learning problem formulation (3,4), other optimization methods not using the gradients can as well be applied. Obviously such techniques can often be combined with regularization methods discussed above.

A standard approach is so called second order learning with state constraints, i.e the application of multi-dimensional pseudo-Newton and conjugate gradient methods, for a review see dos Santos and Zuben [1999]. Galicki et al. [1999] use the Pontragyn maximum principle to minimize a rather general energy-functional. Their framework includes the standard quadratic error, but as well other control theory motivated performance indices and a regularizing weight sum, but the derivation is very complicated and suffers from the problem that suitable initial values have separately to be determined. Suykens and Vandewalle [2000] demonstrate that addition of the weight sum to the standard quadratic error together with the constraints (4) can be treated as a variation of support vector machines. Employing the kernel trick, they obtain a solution which is expressed in the example data and demonstrate an application of learning a double scroll attractor. However, the convexity of the standard SVM optimization and the sparseness of the support vectors are lost and non-convex optimization techniques have to be used. In Sundareshan et al. [1999] the simplex optimization method (not to be confused with the simplex algorithm of linear programming) for systematic search in high dimensional spaces is is applied to RNN training.

4.2 Oscillations and chaotic dynamics

The dynamical capabilities of RNNs have in various ways been used to detect, drive, or generate non-stationary motions. For instance, the autonomous reproduction of a figure eight has become a classical test problem for many standard training algorithms (Pearlmutter [1995], Hagner [1999]), but nevertheless a number of papers address special modifications for periodic or chaotic motions. Ruiz et al. [1998] learn periodic motions for driving repetitive robotic tasks in a ring structured network with a specialized algorithm. Feldkamp et al. [2001] show that a network can switch between different output sequences according to a conditioning input sequence, even with noise in between. Starting from chaotic dynamics, Daucé [2001] uses Hebbian learning to get into resonance with periodic input patterns. Also ideas from chaos control can be used, e.g. stabilization by delayed feedback (Tsui and Jones [1999]) and (Crook,olde-Scheper) in this volume.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 357-368

5 Conclusion

In the current overview we present a large variety of theoretical and practical RNN learning approaches. The amount and intensity of recent work in this field shows that it is worth to deal with the complexity and the computational burden of recurrent learning algorithms in order to profit from their sequence processing capabilities. Some well settled issues can be identified: gradient learning is well understood and algorithms are available but they suffer from the problem of long-term dependencies. The dynamical approximation capabilities and the relations to automata are well known, whereas learning complexity and generalization ability are partially unsolved problems. The classical application domains of language processing, adaptive control, and time series prediction remain popular and are enhanced by new application fields e.g. in bioinformatics, biotechnology, or process control.

With respect to the inherent difficulties of recurrent learning, many researchers propose to use either problem specific algorithms, or specialized architectures, or both. In our opinion, most of these approaches can be interpreted as regularization techniques, either on the algorithmic level or on the structural level of the architecture design. The existence of a large number of such methods gives evidence that the theoretically known problems are severe as well in practice. On the other hand, this indicates as well that successful solutions with respect to concrete tasks can be found. As it is very difficult to measure the generalization capabilities of RNNs, it remains an open question, how to quantify the success of such techniques. To this aim more strict taxonomies of regularization could be developed in the future.

References

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

A. B. Atiya and A. G. Parlos. New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE Trans. Neural Networks*, 11(9):697–709, 2000.

A. Aussem. Dynamical RNNs towards prediction and modeling of dynamical systems. *Neurocomputing*, 28:207–232, 1999.

R. Bakker, J. C. Schouten, C. L. Giles, F. Takens, and C. M. van den Bleek. Learning chaotic attractors by neural networks. *Neural Computation*, 12:2355–2383, 2000.

P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, and G. Soda. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11), 1999.

J. Baltersee and J. Chambers. Non-linear adaptive prediction of speech signals using a pipelined recurrent network. *IEEE Trans. on Signal Proc.*, 1998.

R. Baratti, B. Cannas, A. Fanni, and F. Pilo. Automated RNN design to model the dynamics of complex systems. *Neural Computation & Applications*, 9:190–201, 2000.

G. Barreto, A. Araújo, and H. Ritter. Time in self-organizing maps: An overview of models. *Int. Journ. of Computer Research*, 10(2):139–179, 2001.

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

M. Benson and R. A. Carrasco. Application of a RNN to space diversity in SDMA and CDMA mobile communication systems. *Neur. Comp. and Appl.*, 10:136–147, 2001.

A. M. Bianucci, A. Micheli, A. Sperduti, and A. Starita. Application of cascade correlation networks for structures to chemistry. *Journ. of Appl. Int.*, 12:117–146, 2000.

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 357-368

P. Campolucci, A. Uncini, F. Piazza, and B. D. Rao. On-line learning algorithms for locally RNNs. *IEEE Trans. Neural Networks*, 10(2):253–271, 1999.

R. C. Carrasco, M. L. Forcada, M. Ángeles Valdés-Muñoz, and R. P. Ñeco. Stable encoding of finite-state machines in discrete-time recurrent neural nets with sigmoid units. *Neural Computation*, 12:2129–2174, 2000.

F. Costa, P. Frasconi, V. Lombardo, and G. Soda. Towards incremental parsing of natural language using recursive neural networks. *Applied Intelligence*. accepted for publication.

B. DasGupta and B. Hammer. On approximate learning by multi-layered feedforward circuits. In H. Arimura, S. Jain, and A. Sharma, editors, *ALT'2000*, pages 264–278. Springer, 2000.

E. Daucé. Learning from chaos: A model of dynamical perception. In *Proc. ICANN*, Springer LNCS 2130, pages 1129–1134, Wien, Austria, 2001.

J. E. Dayhoff, P. J. Palmadesso, and F. Richards. *RNNs: Design and Applications*, chapter Oscillation Responses in a Chaotic Recurrent Network. CRC Press, 1999.

E. de Bodt, J. Rynkiewicz, and M. Cottrell. Some known facts about financial data. In M. Verleysen, editor, *9th Europ. Symp. on Art. Neural Networks*. D facto, 2001.

E. P. dos Santos and F. J. V. Zuben. *RNNs: Design and Applications*, chapter Efficient Second-Order Learning Algorithms for Discrete-Time Recurrent Neural Networks. CRC Press, 1999.

L. A. Feldkamp, D. V. Prokhorov, and R. M. Feldkamp. Conditioned adaptive behavior from a fixed neural network. In *Proc. 11. Yale Workshop on Adaptation and Learning Systems*, pages 78–83, New Haven, 2001.

T. C. Ferree and S. R. Lockery. Chemotaxis control by linear recurrent networks. In J. Bower, editor, *Comp. Neuroscience: Trends in Research*, pages 373–377. Plenum Press, 1998.

P. Frasconi and M. Gori. Computational capabilities of local-feedback recurrent networks acting as finite-state machines. *IEEE TNN*, pages 1521–1525, 1996.

P. Frasconi, M. Gori, and G. Soda. RNNs and prior knowledge for sequence processing: A constrained nondeterministic approach. *Knowl.-Based Syst.*, 8(6):313–332, 1995.

P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE TNN*, 9(5):768–786, 1997.

P. Frasconi, M. Gori, and A. Sperduti. Learning efficiently with neural networks: a theoretical comparison between structured and flat representations. In W. Horn, editor, *ECAI 2000*, pages 301–305. IOS Press, 2000.

K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806, 1993.

M. Galicki, L. Leistritz, and H. Witte. Learning continuous trajectories in RNNs with time-dependent weights. *IEEE-NN*, 38(10):714–755, 1999.

G. B. Giannakakis. Highlights of signal processing for communication. *Signal Proc. Magazine*, 16:14–50, 1999.

C. L. Giles, S. Lawrence, and A. C. Tsoi. Noisy time series prediction using a RNN and grammatical inference. *Machine Learning*, 44(1/2):161–183, July/August 2001.

C. G. Hagner. *RNNs: Design and Applications*, chapter Comparison of Recurrent Networks for Trajectory Generation. CRC Press, 1999.

B. Hammer. Generalization ability of folding networks. *IEEE TKDE*, 13(2):196–206, 2001.

B. Hammer and P. Tiňo. Neural networks with small weights implement finite memory machines. Technical Report P-241, Dept. of Math./Comp.Science, Univ. of Osnabrück, 2002.

R. Haschke, J. J. Steil, and H. Ritter. Controlling oscillatory behavior of a two neuron recurrent neural network using inputs. In *Proc. of the Int. Conf. on Artificial Neural Networks(ICANN)*, Springer LNCS 2130, pages 1109–1114, Wien, Austria, 2001.

G. E. Hinton. Connectionist learning procedures. *Aritificial Intelligence*, 40:185–234, 1989.

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

F. C. Hoppensteadt and E. M. Izhikevich. *Weakly Connected Neural Networks*. Springer, 1997.

J. E. Hummel and K. J. Holyoak. Distributed representation of structure: A theory of analogical access and mapping. *Psych. Review*, 104(3):427–466, 1997.

L. Jin and M. M. Gupta. Stable dynamic backpropagation learning in RNNs. *IEEE Trans. Neural Networks*, 10(6):1321–1333, 1999.

J. Kilian and H. T. Siegelmann. The dynamic universality of sigmoidal neural networks. *Inf. and Comp.*, 128:48–56, 1996.

P. Koiran and E. D. Sontag. Vapnik-Chervonenkis dimension of RNNs. In *Proc. of the 3rd Eur. Conf. on Comp. Learning Theory*, pages 223–237, 1997.

T. Kolb, W. Ilg, and J. Wille. Using time-discrete RNNs in nonlinear control. In *Proc. of the World Congress on Comp. Int. '98*, pages 1367–1371, 1998.

J. Kolen. Recurrent networks: state machines or iterated function systems? In *Proc. of the 1993 Conn. Models Summer School*, pages 203–210. Lawrence Erlbaum Associates, 1994.

S. C. Kremer. Lessons from language learning. In L. R. Medsker and L. C. Jain, editors, *Recurrent Neural Networks, Design and Appl.*, pages 179–204. CRC Press, 2000.

S. C. Kremer. Spatio-temporal connectionist networks: A taxonomy and review. *Neural Comp.*, 13(2):249–306, 2001.

S. C. Kremer, R. P. Ňeco, and M. L. Forcada. Constrained second-order recurrent networks for finite-state automata induction. In *Proc. ICANN*, volume 2, pages 529–534, 1998.

P. C. R. Lane and J. Henderson. Incremental syntactic parsing of natural language corpora with simple synchrony networks. *IEEE TKDE*, 13(2):219–231, 2001.

S. Lawrence, C. L. Giles, and S. Fong. Natural language grammatical inference with recurrent neural networks. *IEEE TKDE*, 12(1):126–140, 2000.

C.-S. Leung, A.-C. Tsoi, and L. W. Chan. Two regularizers for recursive least squared algorithms in feedforward multilayered neural networks. *IEEE Trans. Neural Networks*, 12(6): 1314–1332, 2001.

J. McNames, J. A. K. Suykens, and J. Vandewalle. Winning entry of the K.U.Leuven time-series prediction competition. *Int. Journ. of Bifurcation and Chaos*, 9(6):1485–1500, 1999.

B. K. Natarajan. *Machine learning: A theoretical approach*. Morgan Kaufmann, 1991.

T. W. Nattkemper, H. Wersing, W. Schubert, and H. Ritter. A neural network architecture for automatic segmentation of fluorescence micrographs. In M. Verleysen, editor, *Proc. of the 8th Europ. Symp. on Art. Neur. Netw.*, pages 177–182. D facto, 2000.

C. Omlin and C. Giles. Extraction of rules from discrete-time RNNs. *Neural Networks*, 9(1): 41, 1996a.

C. Omlin and C. Giles. Rule revision with RNN. *IEEE TKDE*, 8(1):183–188, 1996b.

R. C. O'Reilly. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, 8(5):895–938, 1996.

J. Park, H. Cho, and D. Park. On the design of BSB neural associative memories using semidefinite programming. *Neural Computation*, 11:1985–1994, 1999.

A. G. Parlos, S. K. Menon, and A. F. Atiya. An algorithmic approach to adaptive state filtering using recurrent neural networks. *IEEE Trans. Neural Networks*, 12(6):1411–1432, 2001.

P. R. Patnaik. A RNN for a fed-batch fermentation with recombinant *Escheria coli* subject to inflow disturbances. *Process Biochemistry*, 32(5):391-400, 1997.

B. A. Pearlmutter. Gradient calculations for dynamic RNNs: A survey. *IEEE Tansactions on Neural Networks*, 6(5):1212–1228, 1995.

P. Rodriguez, J. Wiles, and J. L. Elman. A RNN that learns to count. *Connection Science*, 11 (1):4–40, 1999.

A. Ruiz, D. H. Owens, and S. Townley. Existence, learning, and replication of periodic motions in recurrent neural networks. *IEEE Transactions on Neural Networks*, 9:651–661, 1998.

F. Scarselli and A. Tsoi. Universal approximation using feedforward neural networks: a survey

of some existing methods, and some new results. *Neural Networks*, 11(1):15–37, 1998.

H. T. Siegelmann and E. D. Sontag. Analog computation, neural networks, and circuits. *Theoretical Comp. Science*, 131:331–360, 1994.

E. D. Sontag. Neural nets as systems models and controllers. In *7th Yale Workshop on Adaptive and Learning Syst.*, pages 73–79, 1992.

J. J. Steil. Local structural stability of recurrent networks with time-varying weights. *Neurocomputing*. To appear.

J. J. Steil and H. Ritter. Maximization of stability ranges for RNNs subject to on-line adaptation. In *Proc. of ESANN 99*, pages 369–374, 1999.

J. J. Steil and H. Ritter. Recurrent Learning of Input-Output Stable Behavior in Function Space: A Case Study with the Roessler Attractor. In *Proc. ICANN 99*, pages 761-766, 1999.

I. Stoianov. Recurrent autoassociative networks: developing distributed representations of hierarchically structured sequences by autoassociation. In L. R. Medsker and L. C. Jain, editors, *Recurrent Neural Networks, Design and Appl.*, pages 205–242. CRC Press, 2000.

M. K. Sundareshan and T. A. Condarcure. Recurrent neural-network training by a learning automaton approach for trajectory learning and control system. *IEEE Trans. Neural Networks*, 9(3):354–368, May 1998.

M. K. Sundareshan, Y. C. Wong, and T. Condarcure. *RNNs: Design and Applications*, chapter Training Algorithms for Recurrent Neural Nets that Eliminate the Need for Computation of Error Gradients with Application to Trajectory Production Problem. CRC Press, 1999.

J. A. K. Suykens, B. D. Moor, and J. Vandewalle. Robust local stability of multilayer recurrent neural networks. *IEEE Trans. Neural Networks*, 11(1):222–229, 2000.

J. A. K. Suykens and J. Vandewalle. Recurrent least squares support vector machines. *IEEE Trans. Circuits Systems*, 2000.

J. A. K. Suykens, J. Vandewalle, and B. D. Moor. NL_q theory: Checking and imposing stability of recurrent neural networks for nonlinear modeling. *IEEE Trans. Signal Processing*, 45(11): 2682–2691, 1997.

P. Tiňo, B. G. Horne, and C. L. Giles. Attractive periodic sets in discrete-time recurrent networks (with emphasis on fixed-point stability and bifurcations in two-neuron networks). *Neural Computation*, 13:1379–1414, 2001.

E. Trentin and D. Giuliani. A mixture of RNNs for speaker normalization. *Neural Comput & Applications*, 10:120–135, 2001.

A. P. M. Tsui and A. J. Jones. Periodic response to external stimulation of a chaotic neural network with delayed feedback. *Int. J. Bifurcation and Chaos*, 9:713–722, 1999.

G. Wahba. Generalization and regularization in nonlinear learning systems. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 426–430, Cambridge, MA, 1995. The MIT Press.

X. Wang. Period-doublings to chaos in a simple neural network: An analytic proof. *Complex Systems*, 5:425–442, 1991.

H. Wersing. Learning lateral interactions for feature binding and sensory segmentation. In *Conference on Neural Information Processing: Natural and Synthetic NIPS*, 2001.

H. Wersing, J. J. Steil, and H. Ritter. A competitive layer model for feature binding and sensory segmentation. *Neural Computation*, 3(2):357–387, 2001.

T. Ziemke. Remembering how to behave: RNNs for adaptive robot behavior. In L. R. Medsker and L. C. Jain, editors, *Recurrent Neural Networks, Design and Appl.*, pages 355–389. CRC Press, 2000.

T. Ziemke, M. Bodén, and L. Niklasson. Oil spill detection: A case study of recurrent artificial neural networks. In A. Browne, editor, *Neural Network Analysis, Architectures and Appl.* IOP Press, 1997.