ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

Spiral Recurrent Neural Network for Online Learning

Huaien Gao[1], Rudolf Sollacher[2], Hans-Peter Kriegel[1]

1- University of Munich, Germany

2- Siemens AG, Corporate Technology, Germany

Abstract. Autonomous, self* sensor networks require sensor nodes with a certain degree of "intelligence". An elementary component of such an "intelligence" is the ability to learn online predicting sensor values. We consider recurrent neural network (RNN) models trained with an extended Kalman filter algorithm based on real time recurrent learning (RTRL) with teacher forcing. We compared the performance of conventional neural network architectures with that of spiral recurrent neural networks (Spiral RNN) - a novel RNN architecture combining a trainable hidden recurrent layer with the "echo state" property of echo state neural networks (ESN). We found that this novel RNN architecture shows more stable performance and faster convergence.

1 Introduction

Sensor networks are going to be deployed in many control and diagnosis applications where one expects solutions with low effort for installation, configuration and operational maintenance. Those "plug&play" requirements imply autonomous behavior and forbid extensive parameter tuning. This also requires a certain degree of self-organization with little or no a priori knowledge of the environment. Such kind of "intelligence" can be achieved with self-learning systems. One important component of such a self-learning system are prediction models for sensor values with the following properties: (1) They are able to learn in an online manner, (2) they have low computational complexity such that low price embedded systems can be used and (3) they are stable. The predicted values and associated errors are basic ingredients for diagnostic solutions and efficient energy management of battery powered sensor nodes.

The use of recurrent networks (RNN) is motivated by their successful application in many offline learning tasks [1]. Time delay neural networks (TDNN)[2], which basically are feed-forward neural networks, have the disadvantage of a prefixed number of historic inputs. Unlike a shift register in TDNN, recurrent neural network (RNN) models have recurrent couplings. These enable RNNs to embed - at least theoretically - an infinite history within its recurrent hidden layer. Examples are simple recurrent networks (SRN) by Elman [3], echo state neural networks (ESN) [4], long-short time memory networks (LSTM) [5], and block-diagonal recurrent neural network (BDRNN) [6]. SRNs sometimes show stability problems, mainly due to an unconstrained hidden matrix; ESNs require large recurrent network layers [4] increasing the computational complexity; they also tend to generalize not very well as shown in our result. To our best knowledge we are not aware of investigations involving LSTM networks in online learning tasks as the one we are considering here; moreover we met stability problems in online learning LSTM networks. In order to overcome the aforementioned deficiencies, a novel architecture called spiral recurrent neural network (Spiral RNN) is introduced.

This paper is organized as follows: in section 2 we consider the environment model in question and the generic online learning algorithm. Spiral RNNs will be introduced in section 3. Section 4 and 5 describe the simulation experiments and results. Finally, we draw a conclusion in section 6.

2 Online learning

The aforementioned requirements imply a generic online learning rule. Such a rule can be derived from Bayes' rule and leads to an extended Kalman filter. The learning system has to maintain and update a model of its environment allowing it to predict future observations. A typical ansatz for such a model describing a generic dynamical system is the following:

$$\begin{aligned} \mathbf{w}_t &= \mathbf{w}_{t-1} + \mu_t && (1) \\ \mathbf{s}_t &= \mathcal{G}(\mathbf{s}_{t-1}, \mathbf{w}_t, \mathbf{x}_{t-1}) && (2) \\ \mathbf{x}_t &= \mathcal{H}(\mathbf{s}_t, \mathbf{w}_t) + \nu_t && (3) \end{aligned}$$

Eq.(1) holds as we assume a stationary dynamics of environment, *i.e.* static model parameters $\mathbf{w}_t$ with random fluctuations μ_t obeying Gaussian statistics with a *p.d.f.* $f_{\mu_t} = \mathcal{N}(\mu_t|0, Q_t)$ and covariance matrix Q_t. *Eq.*(2) describes the dynamic evolution of the hidden state vector which depends on the previous state $\mathbf{s}_{t-1}$, as well as previous observation $\mathbf{x}_{t-1}$ and $\mathbf{w}_t$. In *eq.*(3), we assume that the current observation is based on $\mathbf{s}_t$ and $\mathbf{w}_t$. ν_t is the measurement noise, satisfying a *p.d.f.* $f_{\nu_t} = \mathcal{N}(\nu_t|0, R_t)$ where R_t is the covariance matrix.

Following the extended Kalman filter (EKF) algorithm [7], we obtain the following update scheme for model parameters:

$$\begin{aligned} \bar{\mathbf{s}}_t &= \mathcal{G}(\bar{\mathbf{s}}_{t-1}, \bar{\mathbf{w}}_t, \hat{\mathbf{x}}_{t-1}) && (4) \\ O_t &= \mathcal{H}(\bar{\mathbf{s}}_t, \bar{\mathbf{w}}_t) \\ P_t &= P_t^\dagger - P_t^\dagger H_t^T (H_t P_t^\dagger H_t^T + R_t)^{-1} H_t P_t^\dagger \\ \bar{\mathbf{w}}_t &= \bar{\mathbf{w}}_{t-1} + P_t H_t^T R_t^{-1} (\hat{\mathbf{x}}_t - O_t) \end{aligned}$$

where P_t represents the covariance matrix of $\mathbf{w_t}$, $P_t^\dagger$ is the a priori covariance matrix of $\mathbf{w_t}$ satisfying $P_t^\dagger = P_{t-1} + Q_t$, H_t represents the gradient of the output *w.r.t.* $\mathbf{w}_{t-1}$ (notice the indexing):

$$\begin{aligned} H_t &= \frac{dO_t}{d\bar{\mathbf{w}}_{t-1}} = \frac{\partial O_t}{\partial \bar{\mathbf{s}}_t}\frac{d\bar{\mathbf{s}}_t}{d\bar{\mathbf{w}}_{t-1}} + \frac{\partial O_t}{\partial \bar{\mathbf{w}}_{t-1}} \\ \text{with } \frac{d\bar{\mathbf{s}}_t}{d\bar{\mathbf{w}}_{t-1}} &= \frac{\partial \bar{s}_t}{\partial \bar{\mathbf{w}}_{t-1}} + \frac{\partial \bar{s}_t}{\partial \bar{s}_{t-1}}\frac{d\bar{s}_{t-1}}{d\bar{\mathbf{w}}_{t-1}} \simeq \frac{\partial \bar{s}_t}{\partial \bar{\mathbf{w}}_{t-1}} + \frac{\partial \bar{s}_t}{\partial \bar{s}_{t-1}}\frac{d\bar{s}_{t-1}}{d\bar{\mathbf{w}}_{t-2}} && (5) \end{aligned}$$

Using the data $\hat{\mathbf{x}}_{t-1}$ as input in *eq.*(4) corresponds to teacher forcing. The latter expression (multiplication term) in *eq.*(5) is the gradient calculated in real time recurrent learning (RTRL) [8] with teacher forcing. Instability problems may occur, if the norm of some eigenvalues of the Jacobian matrix $\partial s_t / \partial s_{t-1}$ exceeds one; this may lead to an unlimited amplification of gradient components. In a recurrent neural network, this Jacobian matrix is closely related to the hidden weight matrix.

3 The Spiral Recurrent Neural Network

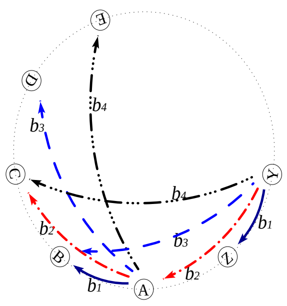
Similar as other RNN models, Spiral RNNs satisfy the following model equations:

$$\begin{aligned} \mathbf{s}_t &= g(W_{hid}\mathbf{s}_{t-1} + W_{in}\hat{\mathbf{x}}_{t-1} + \mathbf{b}_1) \\ \mathbf{x}_t &= h(W_{out}\mathbf{s}_t + \mathbf{b}_2) \end{aligned}$$

Here, $b1$ and $b2$ are bias vectors, matrices W_{in}, W_{hid}, W_{out} represent synaptic connections between layers and g and h are activation functions.

In order to avoid instability problems during online learning, we parametrize the recurrent hidden matrix such that its eigenvalue spectrum is bounded. A suitable choice for one-dimensional data is shown in (6). With all neurons lying on a circle as in *Fig.*1(a), all clockwise nearest-neighbor links have the same weight β_1. Similarly, all clockwise next-to-nearest-neighbor links share the same weight β_2, and so on for longer range links.

$$\begin{pmatrix} 0 & \beta_{l-1} & \dots & \beta_1 \\ \beta_1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \beta_{l-1} \\ \beta_{l-1} & \dots & \beta_1 & 0 \end{pmatrix}_{l \times l} \tag{6}$$



(a) Hidden layer

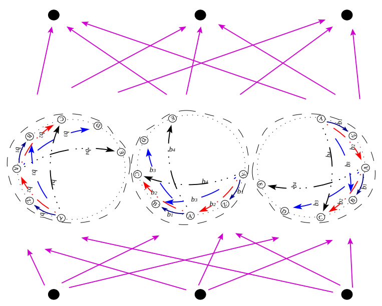

(b) Spiral RNN for 3-dimensional data

Fig. 1: *(a) Sample structure of hidden layer of 1-dimensional data where only part of connections coming out from nodes A and Y are shown. (b) Spiral RNN for 3-dimensional data with isolated but similar structure groups of hidden nodes.*

Thus, for a hidden layer with l neurons, in the hidden weight matrix $W_{hid} \in \mathbb{R}^2_{l\times l}$ in (6) we use a representation $\beta_k = \gamma\ tanh(\xi_k), k \in [1, l-1]$, where $\gamma \in \mathbb{R}^+$

is a given fixed value. This representation introduces the unconstrained parameters ξ_k. Such hidden matrix can realize the fading memory feature because the maximum absolute eigenvalue is bounded:

$$|\lambda^{\vec{\beta}}| \leq \sum_{k=1}^{l-1} |\beta_k| = \gamma \sum_{k=1}^{l-1} tanh(|\xi_k|) \leq \gamma(l-1) \tag{7}$$

For d-dimensional data, the recurrent hidden layer is repeated d times, but input and output neurons are fully connected to all hidden neurons, as in $Fig.1$(b). Thus the hidden weight matrix of Spiral RNNs in the multi-dimensional case is a block diagonal matrix and the maximum absolute eigenvalue of W_{hid} of d-dimensional data will be one of the d maximum absolute eigenvalues in $eq.(7)$ for each dimension: $|\lambda_{max}| = max\{|\lambda^{\beta^{(:,i)}}|\}, i \in [1, d]$.

With this parametrization, Spiral RNNs have a trainable hidden weight matrix but with a pre-definable bound on the eigenvalue spectrum.

4 Experimental Settings

To examine the generalization ability of networks, comparisons were run among BDRNN, SRN, ESN and Spiral RNN with two time series: spike time series with period 21 (each period contains 20 zeros and 1 spike of value 1) and chaotic Lorentz time series (with parameters s=16, r=40, b=6, $x_0 = y_0 = 0.1$, $z_0 = -0.1$, data was scaled down by factor of 20 after being generated). Both time series are corrupted by uniformly distributed noise in the range [0, 0.01]. Training of all networks was based on teacher forcing RTRL-based EKF algorithm introduced in section 2. Note that, in ESN model which has fixed hidden weights, $\frac{d\bar{\mathbf{s}}_t}{d\bar{\mathbf{w}}_{t-1}}$ in $eq.(5)$ equals to zero and thus no RTRL gradient calculation is applied. The covariance matrix R_t for the measurement noise ν_t is the filtered estimate of model error with filter coefficient 0.1. We choose the size of the different RNN models such that they have roughly the same number of model parameters, determining the computational cost (mainly due to EKF training). We randomly initialized the weights of the networks because of lack of a priori information.

In each network, hidden and output activation functions are set to hyperbolic tangent function "*tanh*" and linear map respectively. For ESN, the hidden matrix W_{hid} is scaled such that the maximum absolute eigenvalue $|\lambda_{max}|$ is equal to 0.95; the sparsity of W_{hid} is set to 0.9. For Spiral RNNs, setting $\gamma = \frac{1}{l-1}$ can theoretically ensure $|\lambda_{max}| \leq 1$, but experiments have shown better performance by setting $\gamma = 1$. This is due to the nonlinear "*tanh*" activation function and also the fact that the limit in $eq.(7)$ holds only if $\xi_k = \infty, \forall k \in [1, l-1]$.

Within time intervals [1000:1500], [5000:5500] and [9000:9500], long-term prediction tests were performed using the previous output as the input for the next step. Maximum prediction step $n = 45$ for spike time series, and $n = 15$ for Lorentz time series. The normalized square error $\varepsilon_\tau|\hat{x}_t$ was calculated for each τ-step prediction, given the target $\hat{x}_t$ and variance of data: $\varepsilon_\tau|\hat{x}_t = (x_\tau - \hat{x}_\tau)^2/\sigma^2$. The n-step iterative prediction error (IPE) is defined as: $E_t^{(n)} = \frac{1}{n}\sum_{\tau=1}^{n} \varepsilon_\tau|\hat{x}_t$.

For multidimensional data, IPE value is averaged over all dimensions. In the next section, we will take IPE as the criterion for the comparisons.

5 Result and Discussion

The following tables show the averaged IPE values and respective standard deviation over 30 simulations for different prediction steps n (2^{nd} column) in each time interval. The best solution of each condition/row is marked in bold font.

5.1 Test with spike time series

The comparison results of networks with 100 trainable synaptic weights are listed in *Table*-1. ESN shows good performance for short-term prediction, but falls back for long-term prediction. This already indicates poor generalization behavior. BDRNN cannot recognize any of the spike even in 1-step prediction. The Spiral RNN shows good performance also for long-term prediction, even for a 45-step prediction. The Spiral RNN requires less than 50 spike periods to converge. Note that, due to instability problems, SRN model failed to provide useful output in 8 out of 30 simulations.

	# pred.	Spiral		ESN		SRN		BDRNN	
		mean	std.	mean	std.	mean	std.	mean	std.
$E^{(n)}_{t_{1000}}$	n=1	3e-2	6e-2	**2e-3**	2e-2	3e-2	8e-2	5e-2	8e-3
	n=15	**3e-2**	9e-3	0.55	0.46	3e-2	1e-2	5e-2	2e-2
	n=30	**3e-2**	5e-3	1.2	0.49	**3e-2**	6e-3	5e-2	2e-2
	n=45	4e-2	1e-2	1.6	0.42	**3e-2**	3e-3	5e-2	2e-3
$E^{(n)}_{t_{5000}}$	n=1	2e-2	7e-2	**5e-4**	5e-3	3e-2	8e-2	5e-2	2e-2
	n=15	**2e-2**	1e-2	0.46	0.7	3e-2	2e-2	5e-2	3e-3
	n=30	**2e-2**	8e-3	1.1	0.51	3e-2	1e-2	5e-2	2e-3
	n=45	**2e-2**	8e-3	1.6	0.41	3e-2	8e-3	5e-2	6e-4
$E^{(n)}_{t_{9000}}$	n=1	1e-2	7e-2	**2e-4**	1e-3	2e-2	8e-2	5e-2	2e-2
	n=15	**2e-2**	9e-2	0.49	1.1	3e-2	2e-2	5e-2	4e-3
	n=30	**2e-2**	1e-2	1.3	1	3e-2	1e-2	5e-2	4e-3
	n=45	**2e-2**	2e-2	1.9	0.93	3e-2	9e-3	5e-2	4e-3

Table- 1: *IPE comparisons on spike time series with 100 weights. Note that IPE values tend to be small because of the characteristics of spike time series.*

5.2 Test with Lorentz time series

Chaotic time series like the Lorentz time series are excellent examples for testing the models' ability of generalization. Results shown in *Table*-2 confirm the impression from the tests with spike time series: Spiral RNNs converge fast, generalize well and show excellent performance in long-term prediction tasks. SRN and BDRNN improve from test interval to test interval although it converges not as fast as Spiral RNN. ESN is unable to make accurate predictions, at least for this small network size with 100 trainable weights. Similar results have been achieved in simulations with 200 and 300 trainable parameters.

Comparisons were also performed on a 4-dimensional chaotic time series (extension of Lorentz data). Again, the Spiral RNN has achieved better results than the other models.

Besides these benchmark experiments, Spiral RNN has been successfully employed in predicting quasi-periodical trajectories of computer mouse moved by user.

	# pred.	Spiral		ESN		SRN		BDRNN	
		mean	std.	mean	std.	mean	std.	mean	std.
$E_{t_{1000}}^{(n)}$	n=1	**5e-3**	3e-3	0.4	4e-2	8e-2	0.15	8e-2	3e-2
	n=5	**5e-2**	2e-2	2.3	0.26	0.23	0.14	0.88	0.3
	n=10	**0.16**	4e-2	4.5	0.72	0.69	0.36	2.7	0.84
	n=15	**0.33**	0.17	6.2	1	1.5	0.83	4.9	1.2
$E_{t_{5000}}^{(n)}$	n=1	**6e-4**	5e-4	3e-1	8e-2	6e-3	2e-2	9e-3	6e-3
	n=5	**7e-3**	6e-3	1.6	0.2	3e-2	2e-2	0.11	0.11
	n=10	**2e-2**	2e-2	4.2	0.53	0.16	0.17	0.53	0.64
	n=15	**4e-2**	3e-2	6	0.68	0.57	0.66	1	1.2
$E_{t_{9000}}^{(n)}$	n=1	**2e-4**	4e-5	0.19	6e-2	1e-3	2e-3	1e-3	1e-3
	n=5	**1e-3**	4e-4	1	0.21	1e-2	6e-3	2e-2	1e-2
	n=10	**3e-3**	1e-3	3	0.42	5e-2	3e-2	7e-2	8e-2
	n=15	**8e-3**	3e-3	5	0.6	0.15	0.12	0.2	0.43

Table- 2: *IPE comparisons on Lorentz time series with 100 weights*

6 Conclusion

In this paper, we considered the use of RNNs in applications like self-organizing sensor networks. Our focus is on autonomous online-learning required to be stable in performance, fast in convergence and cheap in computation.

We introduced the Spiral RNN model in order to overcome several drawbacks of traditional models and to combine advantageous features: trainable recurrent hidden layer as in SRNs together with a constraint on the eigenvalue spectrum of the recurrent weight matrix as in ESNs. This novel architecture has been proven to provide stable and accurate results with fast convergence. Thus, it is a suitable candidate for online learning tasks in embedded systems.

References

[1] H.G Zimmermann, R. Grothmann, A.M. Schaefer, and Ch Tietz. Dynamical consistent recurrent neural networks. *Int. Joint Conference on Neural Networks (IJCNN)*, 2005.

[2] D. E. Rumelhart and et al. J. L. McClelland. *Parallel distributed processing: explorations in the microstructure of cognition.* Cambridge, Mass.,MIT Press, 1986.

[3] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[4] Herbert Jaeger. Adaptive nonlinear system identification with echo state networks. *Advances in Neural Information Processing Systems*, 15:593–600, 2003.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput*, 9(8):1735–1780, Nov 1997.

[6] P.A. Mastorocostas and J.B. Theocharis. On stable learning algorithm for block-diagonal recurrent neural networks, part 1: the RENNCOM algorithm. *IEEE International Joint Conference on Neural Networks*, 2:815– 820, 2004.

[7] F.L. Lewis. *Optimal Estimation: With an Introduction to Stochastic Control Theory.* A Wiley-Interscience Publication, 1986. ISBN: 0-471-83741-5.

[8] R.J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computtion*, 1:270–280, 1989.