ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

Conditional Prediction of Time Series Using Spiral Recurrent Neural Network

Huaien Gao[1,2], Rudolf Sollacher[2]

1- University of Munich, Germany

2- Siemens AG, Corporate Technology, Germany

Abstract. Frequently, sequences of state transitions are triggered by specific signals. Learning these triggered sequences with recurrent neural networks implies storing them as different attractors of the recurrent hidden layer dynamics. A challenging test and also useful for application is conditional prediction of sequences giving just the trigger signal as an input and letting the recurrent neural network evolve the sequences automatically. This paper addresses this problem with the spiral recurrent neural network (SpiralRNN) architecture.

1 Introduction

There are many applications where a system evolves differently depending on one or more external triggering "context" signals. A typical example in discrete manufacturing is a robot manipulating bottles arriving on a conveyor belt: Depending on sensor signals indicating (i) the presence of a bottle, (ii) whether the bottle is already filled or empty and (iii) which liquid to fill in, the robot starts different sequences of actions like filling liquid from different sources or ignoring already filled bottles. A controller which can be trained easily to perform these sequences depending on the sensor information represents a significant simplification for installing and reconfiguring automation systems.

Here, we consider another application, namely a warehouse in figure *1(a)* for many commodities which arrive at the warehouse in random order rather than regular one. When a trolley is assigned a job to convey a product to its respective cabin, a suitable sensor provides the associated information ϕ indicating, for example, the destination cabin for the product. During the transportation, the trajectory of trolley and product is recorded and used for on-line training of a dynamical trajectory model that, given the previous positions on a grid of suitable spatial resolution, predicts the next position. After sufficient time of learning, the model should be able to reproduce the trajectories of previously seen products given the associated ϕ-values.

In the next section, Spiral recurrent neural network architecture is introduced for the conditional prediction problem. *Section-3* and *section-4* will present the simulations, results and discussion, where performance of different neural network models will be addressed. Conclusion will be given in the last section.

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

2 Conditional Prediction with SpiralRNN

2.1 Spiral Recurrent Neural Network

Spiral recurrent neural network (SpiralRNN) [3] is a novel neural network architecture for efficient and stable on-line time series prediction. Forward evolution of a SpiralRNN model is very similar to conventional recurrent neural networks, which follows equation *(1)* and equation *(2)*. Here, x is the data whose dynamics is to be trained, s refers to the hidden state vector, $\mathscr{H}$ and $\mathscr{G}$ are the activation functions in hidden layer and output layer respectively, W_{in} W_{hid} and W_{out} are connection-weight matrices in the input, hidden and output layers, and b_1 and b_2 are the respective bias vectors in hidden and output layer.

$$s_{t+1} = \mathscr{H}(W_{hid}s_t + W_{in}x_t + b_1) \quad (1)$$

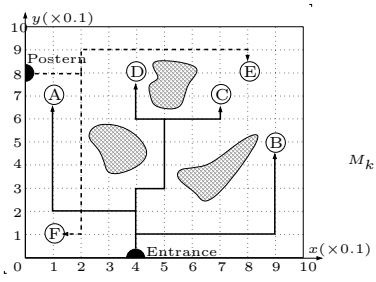
$$x_{t+1} = \mathscr{G}(W_{out}s_{t+1} + b_2) \quad (2)$$

SpiralRNN models differ from conventional RNNs in their special structure of the hidden layer weight matrix W_{hid}, which is a block-diagonal matrix with μ sub-blocks. Each sub-block matrix M_k is determined by an associated vector $\beta^{(k)} = \{\beta_1^{(k)}, \beta_2^{(k)}, \ldots, \beta_{u-1}^{(k)}\}^T$ as shown in figure *1(b)*, with $k \in [1, \mu]$ being the indices of sub-blocks and $u \in \mathbf{N}$ indicating the size of M_k. Furthermore the value of β satisfies the condition: $\beta_i^{(k)} = tanh(\gamma_i^{(k)})$. With the special structure as well as the squash function of vector β, the maximum absolute eigenvalue of k-th sub-block matrix is bounded as shown in equation *(3)*. Because of the block-diagonal structure, matrix W_{hid}'s maximum absolute eigenvalue λ_{max} satisfies equation *(4)*.

$$|\lambda^{(k)}| \leq \sum_{i=1}^{u-1} |\beta_i^{(k)}| = \sum_{i=1}^{u-1} |tanh(\gamma_i^{(k)})| < u - 1 \quad (3)$$

$$\lambda_{max} = \max_{k \in [1,\ldots,\mu]} \{\lambda^{(k)}\} \quad (4)$$

With such constraint on the eigenvalue spectrum of hidden-weight matrix, as well as the fact that matrix W_{hid} is determined by vectors which can be optimized, SpiralRNN model enjoys the advantage of constraint eigenvalue spectrum



(a) Schematic Diagram of warehause

$$M_k = \begin{pmatrix} 0 & \beta_{u-1}^{(k)} & \beta_{u-2}^{(k)} & \cdots & \beta_1^{(k)} \\ \beta_1^{(k)} & 0 & \beta_{u-1}^{(k)} & \cdots & \vdots \\ \beta_2^{(k)} & \beta_1^{(k)} & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \beta_{u-1}^{(k)} \\ \beta_{u-1}^{(k)} & \cdots & \cdots & \beta_1^{(k)} & 0 \end{pmatrix}_{u \times u}$$

(b) Structuer of sub-block M_k

Fig- 1: *(a) Schematic diagram of warehouse. Product cabins are denoted by capital letters and obstacles are in shaded area. (b) Structure of sub-block matrix M_k.*

like in echo state neural network (ESN) [1] and the advantage of rich complexity of dynamics modeling ability like in simple recurrent net (SRN) [2], and is hence suitable for on-line time series prediction. For more details, refer to [3].

2.2 Implementation of SpiralRNN in Conditional Prediction

In the conditional prediction problem of trajectory reproduction, data tuple $x_t = [p_t, \phi]$ consist of the trajectory data p_t (which is two dimensional in this case) and the corresponding ϕ-value (which is one dimensional in this case), where $t \in [1, l_p]$ is the index for time step and l_p is the length of the particular trajectory pattern. Here, we assume that ϕ is unique and constant for one particular trajectory/product; this can be realized by storing the initial trigger signal (given by sensor) until another signals terminating the trajectory, *i.e.* product has been conveyed to the destination. At $t = 0$, different trajectory patterns will be separately assigned a virtual starting point p_0. The value of p_0 is different for each pattern but is randomly initialized the first time a specific pattern occurs.

Initialization: Every time before training or testing with any trajectory pattern $\vec{p}$, the hidden-state vector of the neural network will be reset to zero in order to avoid influence from previous data sequence. More important, the tuple $x_0 = [p_0, \phi]$ at time $t = 0$ is initialized such that p_0 value is unique but constant for different product/trajectory, as is the ϕ value. In such a way, the starting data item of the same product/trajectory, which is to be used in training and testing, is the constant.

Training: At time step $t + 1$, for instance, the target tuple is constructed with the latest data: $\hat{x}_{t+1} = [p_{t+1}; \phi_i]$. By setting the previous tuple $\hat{x}_t$ as input, the neural network iterates according to equation *(1)* and equation *(2)*. By teacher forcing, x_t is replaced by $\hat{x}_t$ in equation *(1)*. Different from batch training, the network parameters are updated each time step upon comparing the output x_{t+1} with the new tuple $\hat{x}_{t+1}$. The on-line training is based on an extended Kalman filter (EKF)[4] with a real-time recurrent learning gradient; this combination can be justified theoretically and has proven successful in earlier tests [3]. Training will be stopped as soon as the last tuple for the particular trajectory of a product is given; this can be realised by a sensor signal indicating arrival at the target position.

Testing: In testing, the production type is known, so is the ϕ-value, and the task is to reproduce the corresponding trajectory for monitoring and diagnosis purposes. Again, the initialization step will be taken before the starting of the testing. Having the initial tuple x_0 in initialization step, the neural network iterates according to equation *(1)* and equation *(2)* as it does in training step. The output of network in testing will be set as the input in the next iteration of the autonomous test. Such autonomous iterations continue until the stopping criterion is satisfied, usually the length of autonomous test l_p is predefined.

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

3 Experimental Settings

Data The trajectory data set is extracted from the trajectory coordinates of different products in the warehouse scenario shown in Figure *1(a)*. It is assumed that, in each time step, the trolley moves from one grid point to the neighbor grid point. Values of ϕ for trajectory patterns A to F are respectively set to $\{1, -1, 0.5, -0.5, 2, -2\}$. The initial virtual starting point p_0 is randomly initialized within range $(-1, 1)$ for different patterns. The length of each trajectory (excluding the virtual starting point) is set as $l_p = 11$. The number of trajectories n_p in each simulation varies, ranging from 2 to 6. Choice of data-sequences of trajectories will orderly start from product A, *e.g.* trajectories of products A, B and C will be chosen when $n_p = 3$. Data are corrupted by normally distributed noise $\mathcal{N}(0, 0.01^2)$ with mean 0 and standard deviation 0.01.

Comparison Models The performance of SpiralRNN model will be compared to that of conventional neural network models including: echo state network (ESN), block diagonal recurrent net (BDRNN)[5] and the classical multi-layer perceptron (MLP). For sake of fair comparison, number of trainable parameters (*i.e.* the *complexity* of model) in the models will keep to the same level, as the computational cost of extended Kalman filter (EKF) training is mainly dependent on this value. As the number of input/output neurons of models is dependent on the dimension of data and is therefore fixed, one can change the number of hidden nodes in order to change the complexity of the particular model.

Model parameters in most cases were initialized according to the normal distribution $\mathcal{N}(0, 0.01^2)$. For reason of easy deployment, construction of SpiralRNN model follows [3] and number of hidden units will be the same as number of input nodes where hidden units have the same size and structure. Configuration of ESN model follows [1], in such a way that the hidden weight matrix W_{hid} is initialized randomly over range $[-1, 1]$ with the sparsity value 95%, afterward the matrix is rescaled such that the maximum eigenvalue of W_{hid} is equal to 0.8. The BDRNN model will take the scaled orthogonal version, where the entries of the 2×2 sub-block matrix is determined by two variables w_1 and w_2 and they satisfy $w_1^2 + w_2^2 \leq 1$ (for details refer to [5]).

Training and Testing Training was implemented in rounds, where in each round the network model was trained with all trajectory time series separately. Note that entries of the hidden state are re-set to zeros before the training with any trajectory. After each 10 rounds of such training, the trained model produces corresponding autonomous output given different initial inputs. For more details, refer to *section-2.2*.

Evaluation Evaluation focuses on the output vector components corresponding to trajectory, namely Γ. The evaluation error ε is then calculated according

to equation *5* and equation *6*. Note that $\overline{\epsilon_{i,t,k}}$ in equation *6* refers to taking the mean value of ϵ over all indices i, t and k, and that when $t = l_p$ ϵ refers to the error of l_p-step-ahead prediction. The final result takes the average value over 30 simulations.

$$\epsilon_{i,t,k} = (\hat{x}_{i,t,k} - x_{i,t,k})^2, \quad i \in \Gamma, t \in [1, l_p], k \in [1, n_p] \tag{5}$$

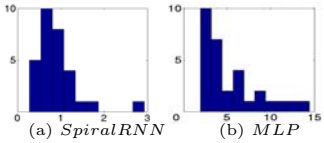
$$\varepsilon = \overline{\epsilon_{i,t,k}} \tag{6}$$

4 Results

Performance result from competing models of different size in different tasks are given in *table-1*, where models have been trained $n_t = 20$ rounds. Figure *2* illustrates the histogram plot of ε value over 30 simulations with n_p=4 training patterns and 200 parameters in model. *Table-2* shows the improvement of performance of SpiralRNN model with more training data. The histogram of ε over 30 runs of 200-weights SpiralRNN model is given in figure *3*. All numbers in tables are base-10 logarithm values of the respective ε. In histogram figures, X-axis represents the value of ε with unit of 10^{-3} and Y-axis shows the occurrence frequency of respective ε value.

Model	100 weights			200 weights			300 weights	
	n_p=2	n_p=3	n_p=4	n_p=4	n_p=5	n_p=6	n_p=5	n_p=6
SpiralRNN	-3.23	-2.84	-2.44	-3.04	-2.76	-2.53	-2.97	-2.73
ESN	-1.15	-0.82	-0.66	-0.77	-0.64	-0.50	-0.70	-0.57
BDRNN	-1.52	-2.17	-1.98	-2.25	-2.24	-1.97	-2.27	-2.23
MLP	-2.11	-2.01	-1.10	-2.27	-1.73	-1.29	-1.76	-1.51

Table- 1: *Evaluation error ε (in logarithmic scale) of models after $n_t = 20$ training rounds in different network size in different tasks.*

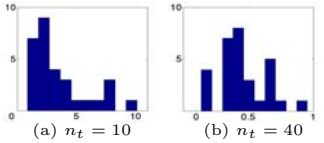


(a) *SpiralRNN* (b) *MLP*

Fig- 2: *Histograms of ε from 200-weight models with n_p=4 and n_t=20*

training rounds	100 weights			200 weights			300 weights	
	n_p=2	n_p=3	n_p=4	n_p=4	n_p=5	n_p=6	n_p=5	n_p=6
n_t=10	-2.76	-2.46	-1.99	-2.47	-2.23	-2.03	-2.39	-2.02
n_t=20	-3.23	-2.84	-2.44	-3.04	-2.76	-2.53	-2.97	-2.73
n_t=30	-3.43	-3.00	-2.54	-3.19	-2.99	-2.76	-3.17	-3.04
n_t=40	-3.63	-3.04	-2.63	-3.34	-3.11	-2.84	-3.31	-3.21

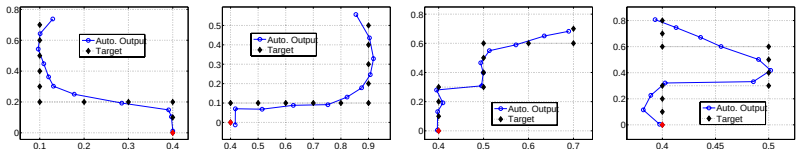
Table- 2: *Evaluation error ε (in logarithmic scale) of SpiralRNN model in different network size in different tasks.*



(a) $n_t = 10$ (b) $n_t = 40$

Fig- 3: *Histograms of ε from 200-weight SpiralRNN with $n_p = 4$*

The results in these tables clearly show that (i) different trajectories can be stored by recurrent neural networks simultaneously and (ii) that SpiralRNN outperforms other approaches in different prediction tasks and with different size of network. It is worth mentioning that SRN shows similar or even slightly better results than SpiralRNN but sometimes suffer from instability effects such that training fails completely (see also [3]). On the other hand, this result indicates that the constraints on the hidden layer structure of a SpiralRNN do not severely constrain the modeling power of this architecture compared to the unconstrained SRN. The worse performance of ESN indicates that their reduced variability (only the output weights can be trained) imposes too many constraints. BDRNN and MLP models have achieved better results than ESN, but suffer from slow convergence. The histogram plot figure *2* illustrates the improved stability of the SpiralRNN compared with *e.g.* MLP.



(a) trajectory A (b) trajectory B (c) trajectory C (d) trajectory D

Fig- 4: *Autonomous outputs on* $n_p = 4$ *trajectory patterns from* 200*-weight SpiralRNN model after* $n_t = 20$ *training rounds.*

Typical examples of trajectory prediction by SpiralRNN are given in figure *4*. Note that in each sub-plot the output trajectory is the autonomous result given initial starting value but no further data input.

Increasing the number of trajectories to be predicted will generally require either longer training time or larger neural network model in order to obtain the same level of performance. *Table-2* reports the performance of a SpiralRNN architecture of different network size and in different prediction tasks with varying number of trajectory patterns. Figure *3* which compares the histogram of error from SpiralRNN model at $n_t = 10$ training rounds and that at $n_t = 40$ has confirmed the convergence of learning. Because of the grid, one prediction can be consider to be matched if the absolute prediction error is smaller than 0.05, therefore the threshold value is defined as $\theta = \log 10(0.05^2) \simeq -2.6$. It is shown in *table-2* that, after $n_t = 20$ or even $n_t = 10$ training rounds, SpiralRNN model is able to achieve an evaluation error smaller or close to this threshold in all mentioned tasks.

5 Conclusion

This paper addresses the conditional prediction of data time series with a SpiralRNN model. We simulate the problem with a warehouse scenario where a trigger signal depending on product type and thus destination provides a value of the additional input ϕ. The task is to reproduce the trajectory of transportation of the respective product. SpiralRNN model has proven successful in such conditional prediction tasks compared to other state-of-the-art approaches.

References

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

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

[3] Huaien Gao, Rudolf Sollacher, and Hans-Peter Kriegel. Spiral recurrent neural network for online learning. In *15th European Symposium On Artificial Neural Networks Advances in Computational Intelligence and Learning*, Bruges (Belgium), April 2007.

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

[5] 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.