X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

RECURSIVE INCREMENTAL GAUSSIAN MIXTURE NETWORK FOR SPATIO-TEMPORAL PATTERN PROCESSING

Rafael C. Pinto, Paulo M. Engel and Milton R. Heinen
Informatics Institute, Universidade Federal do Rio Grande do Sul (UFRGS)
{rcpinto,engel,mrheinen}@inf.ufrgs.br

Abstract – This work introduces a novel neural network algorithm for online spatio-temporal pattern processing, called Recursive Incremental Gaussian Mixture Network (RecIGMN). The proposed algorithm adds a temporal context by means of self-reference to the Incremental Gaussian Mixture Network (IGMN), enabling it for spatio-temporal tasks. The algorithm was compared against other spatio-temporal pattern processing neural networks in time-series prediction tasks and showed competitive performance, with the advantage of fast, incremental learning with just one single scan through data.

Keywords – Artificial neural networks, incremental learning, Gaussian mixture models, spatio-temporal pattern processing

Resumo – Este trabalho introduz um novo algoritmo de redes neurais para processamento online de padrões espaço-temporais, chamado Recursive Incremental Gaussian Mixture Network (RecIGMN). O algoritmo proposto adiciona um contexto temporal através de auto-referência à Incremental Gaussian Mixture Network (IGMN), habilitando-a para tarefas espaço-temporais. O algoritmo foi comparado com outras redes neurais para processamento de padrões espaço-temporais em tarefas de previsão de séries temporais e demostrou desempenho competitivo, com a vantagem de um aprendizado rápido, incremental, com apenas uma passada nos dados.

Palavras-chave – Redes neurais artificiais, aprendizado incremental, modelos de mistura de Gaussianas, processamento de padrões espaço-temporais

1. Introduction

This work presents a novel neural network algorithm for online spatio-temporal pattern processing, called Recursive Incremental Gaussian Mixture Network (RecIGMN). The proposed algorithm adds a temporal context by means of self-reference [1] to the Incremental Gaussian Mixture Network (IGMN, previously known as Incremental Probabilistic Neural Network or IPNN on early versions) [2] [3], enabling it for spatio-temporal tasks.

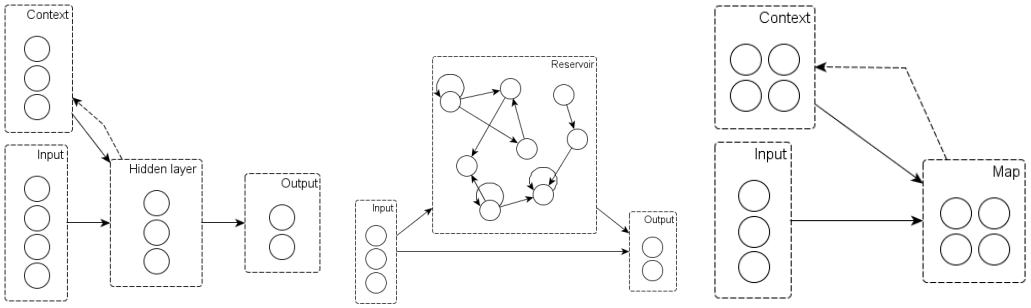
The IGMN is an one-shot incremental learning algorithm, needing only a single scan through the training data in order to build a consistent model. But the IGMN is essentially a static algorithm, meaning that it basically works only for static tasks, where any pattern is independent of past ones.

The RecIGMN uses feedback connections to provide the IGMN with information about its own state, producing a temporal context.

This work is structured as follows: The IGMN is described in section 2. Section 3 presents the new algorithm, the RecIGMN. In section 4, the RecIGMN is compared to other algorithms in time-series prediction tasks. Section 5 finishes this work with concluding remarks about the new algorithm and future works.

1.1 Related Works

Neural networks have been used for time-series prediction for a long time [4]. Some of the main algorithms used for this task are Elman Networks [5] (figure 1(a)), BPTT (Back-Propagation Through Time) [6], RTRL (Real-Time Recurrent Learning) [7] and ESN (Echo-State Networks) [8] (figure 1(b)). There are also unsupervised neural networks that can handle time-series, such as the RecSOM (Recursive Self-Organizing Map) [1] (figure 1(c)). Its approach for context learning is similar to the one used in the Elman Network, through recurrent connections providing self-reference. Since the IGMN was built upon an unsupervised algorithm (IGMM, see next section), a natural approach to extend it for spatio-temporal tasks is to take the same approach taken in the RecSOM.



(a) (b) (c)

Figure 1: Recurrent neural networks. (a) An Elman Network with 4 inputs, 3 hidden neurons and 2 outputs The context is a copy of the last hidden layer activation. (b) An Echo-State Network with 4 inputs, 8 reservoir neurons and 2 outputs. (c) A RecSOM with 3 inputs and 4 map neurons (2x2). The context is a copy of the last map activation.

2 Incremental Gaussian Mixture Network

The IGMN (Incremental Gaussian Mixture Network) algorithm [2] uses an incremental approximation of the EM algorithm [9], the IGMM (Incremental Gaussian Mixture Model) [10]. It creates and continually adjusts probabilistic models consistent to all sequentially presented data, after each data point presentation, and without the need to store any past data points. Its learning process is aggressive, or "one-shot", meaning that only a single scan through the data is necessary to obtain a consistent model.

IGMN (and IGMM) adopts a gaussian mixture model of distribution components (known as a *cortical region*) that can be expanded to accommodate new information from an input data point, or reduced if spurious components are identified along the learning process. Each data point assimilated by the model contributes to the sequential update of the model parameters based on the maximization of the likelihood of the data. The parameters are updated through the accumulation of relevant information extracted from each data point.

Differently from IGMM, however, the IGMN is capable of supervised learning, simply by assigning any of its input vector elements as outputs (any element can be used to predict any other element). This architecture is depicted on figure 2.

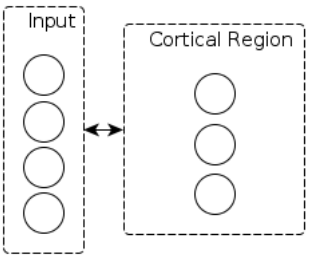


Figure 2: An example of IGMN with 4 input nodes and 3 gaussian components. Any input element can be predicted by using any other element, which means that the input vector can actually be divided into input and output elements.

2.1 Learning

The algorithm starts with no components, which are created as necessary (see subsection 2.2). Given input $\mathbf{x}$, the IGMN algorithm processing step is as follows. First, a gaussian function for each component j and input $\mathbf{x}$ is computed:

$$g(x,j) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\mathbf{C}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right) \tag{1}$$

then it's used to obtain the likelihood for each component in the following way:

$$p(\mathbf{x}|j) = \frac{g(x,j)}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_j|}} \tag{2}$$

where D is the input dimensionality, $\boldsymbol{\mu}_j$ the j^{th} component mean and $\mathbf{C}_j$ its covariance matrix.

After that, posterior probabilities are calculated for each component as follows:

$$p(j|\mathbf{x}) = \frac{p(\mathbf{x}|j)p(j)}{\sum_{q=1}^{M} p(\mathbf{x}|q)p(q)} \qquad \forall j \tag{3}$$

where M is the number of components. Now, parameters of the algorithm must be updated according to the following equations:

$$v_j(t) = v_j(t-1) + 1 \tag{4}$$

$$sp_j(t) = sp_j(t-1) + p(j|\mathbf{x}) \tag{5}$$

$$\mathbf{e}_j = \mathbf{x} - \boldsymbol{\mu}_j \tag{6}$$

$$\omega_j = \frac{p(j|\mathbf{x})}{sp_j} \tag{7}$$

$$\Delta\boldsymbol{\mu}_j = \omega_j \mathbf{e}_j \tag{8}$$

$$\boldsymbol{\mu}_j(t) = \boldsymbol{\mu}_j(t-1) + \Delta\boldsymbol{\mu}_j \tag{9}$$

$$\mathbf{C}_j(t) = \mathbf{C}_j(t-1) - \Delta\boldsymbol{\mu}_j\Delta\boldsymbol{\mu}_j^T + \omega\left[\mathbf{e}\mathbf{e}^T - \mathbf{C}_j(t-1)\right] \tag{10}$$

$$p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q} \tag{11}$$

where sp_j and v_j are the accumulator and the age of component j, respectively, and $p(j)$ is its prior probability.

2.2 Creating New Components

In order to create new components, the cortical region must reconstruct its input **x** based on the posterior probabilities obtained in equation 3. Let **x** be a concatenation of two vectors **a**, the actual input or known part, and **b**, the target / desired value ($\mathbf{x} = [\mathbf{a}; \mathbf{b}]$). Then the reconstruction of **b** given **a** is obtained by the following equation:

$$\hat{\mathbf{b}} = \sum_{j=1}^{M} p(j|\mathbf{a})(\boldsymbol{\mu}_{j,b} + \mathbf{C}_{j,ba}\mathbf{C}_{j,a}^{-1}(\mathbf{a} - \boldsymbol{\mu}_{j,a})) \tag{12}$$

where $\boldsymbol{\mu}_{j,b}$ is just the target part of the jth component's mean vector, $\mathbf{C}_{j,ba}$ is the submatrix of the jth component covariance matrix associating the input and output parts of the data, $\mathbf{C}_{j,a}$ is the submatrix corresponding to the input part only and $\boldsymbol{\mu}_{j,a}$ is just the input part of the jth component's mean vector. After reconstructing the input, the reconstruction error can be obtained by:

$$\epsilon = \max_{i \in D}\left[\frac{|x_i - \hat{x}_i|}{X_{max,i} - X_{min,i}}\right] \tag{13}$$

where $X_{max,i}$ and $X_{min,i}$ are the ith column's maximum and minimum values expected for the entire dataset (just approximated values are needed, since IGMN-based algorithms don't require availability of the entire dataset beforehand). If there are no components or ϵ is greater than a manually chosen threshold ϵ_{max} (e.g., 0.1), then a new component is created and initialized as follows:

$$\boldsymbol{\mu} = \mathbf{x}; \qquad sp = 1; \qquad v = 1; \qquad p(j) = \frac{1}{\sum_{i=1}^{M} sp_i}; \qquad \mathbf{C} = \sigma_{ini}^2$$

where M already includes the new component and σ_{ini} can be obtained by:

$$\sigma_{ini} = diag(\delta[X_{max} - X_{min}]) \tag{14}$$

where δ is a manually chosen scaling factor (e.g., 0.1) and diag returns a diagonal matrix having its input vector in the main diagonal.

2.3 Removing Spurious Components

A component j is removed whenever $v_j > v_{min}$ and $sp_j < sp_{min}$, where v_{min} and sp_{min} are manually chosen (e.g., 5.0 and 3.0, respectively). In that case, also, $p(q)$ must be adjusted for all $q \in M$, $q \neq j$, using equation 11.

2.4 Recalling

In IGMN, any element can be predicted by any other element. This is done by reconstructing data from the target elements (**b**) by estimating the posterior probabilities using only the input elements, as follows:

$$p(j|\mathbf{a}) = \frac{p(\mathbf{a}|j)p(j)}{\sum_{q=1}^{M} p(\mathbf{a}|q)p(q)} \qquad \forall j \tag{15}$$

It's similar to equation 3, except that it uses the actual input vector **a** instead of the full **x** vector, with the target elements **b** removed from calculations. After that, **b** can be reconstructed using equation 12.

3 The Recursive Incremental Gaussian Mixture Network

The RecIGMN is a temporal extension of the IGMN algorithm, which is augmented with feedback connections from its cortical region. This architecture can be seen in figure 3. Its feedback connections are analogue to the ones of the Elman Network and RecSOM, creating additional input in the form of a context vector, which is processed together with data inputs.

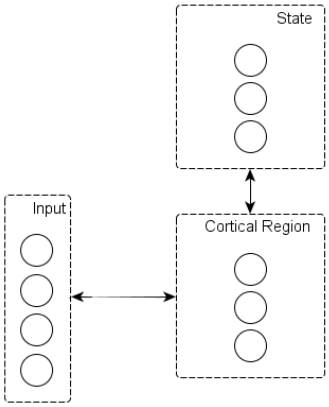


Figure 3: An example of RecIGMN with 4 inputs and 3 gaussian components.

Therefore, all IGMN equations from the previous section apply, except that the input is augmented with a context vector **c** obtained from each gaussian component in the last processing step, resulting in a new input vector $\mathbf{u} = [\mathbf{c}; \mathbf{x}]$ ($[;]$ is the vector concatanation operation and **c** is initialized as the null vector when the algorithm starts). This has a huge implementation impact, since the number of gaussian components in a IGMN is variable, meaning that the algorithm must cope with variable size inputs now. In the next subsections, the modifications to the original IGMN are described.

3.1 Learning

The modifications to the learning algorithm in the RecIGMN consist in weighting context and current data by an α parameter. Equation 1 must be replaced by

$$g_{j,c} = \exp\left(-\frac{1}{2}(\mathbf{c}(t-1) - \boldsymbol{\mu}_{j,c})^T \mathbf{C}_{j,c}^{-1}(\mathbf{c}(t-1) - \boldsymbol{\mu}_{j,c})\right) \tag{16}$$

$$g_{j,x} = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{j,x})^T \mathbf{C}_{j,x}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{j,x})\right) \tag{17}$$

$$c_j(t) = \alpha g_{j,c} + (1-\alpha) g_{j,x} \tag{18}$$

where **c** is the context vector, $\boldsymbol{\mu}_{j,c}$ is the context portion of the jth component's mean, while $\boldsymbol{\mu}_{j,x}$ is the current input portion of the same component's mean. $\mathbf{C}_{j,c}$ is the context portion of the jth component's covariance matrix, while $\mathbf{C}_{j,x}$ is its current input portion. Equation 2 is replaced by

$$p_{j,c} = \frac{1}{(2\pi)^{D_c/2}\sqrt{|\mathbf{C}_{j,c}|}} \exp\left(-\frac{1}{2}(\mathbf{c}(t-1) - \boldsymbol{\mu}_{j,c})^T \mathbf{C}_{j,c}^{-1}(\mathbf{c}(t-1) - \boldsymbol{\mu}_{j,c})\right) \tag{19}$$

$$p_{j,x} = \frac{1}{(2\pi)^{D_x/2}\sqrt{|\mathbf{C}_{j,x}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{j,x})^T \mathbf{C}_{j,x}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{j,x})\right) \tag{20}$$

$$p(\mathbf{u}|j) = \alpha p_{j,c} + (1-\alpha) p_{j,x} \tag{21}$$

where D_c is the size of the context vector and D_x is the size of the original input vector.

3.2 Creating New Components

Whenever a new component is created (subsection 2.2), all components means and covariance matrixes must be adjusted by adding a new element corresponding to the activation of the newly created component. For the covariance matrixes, it means adding 1 full column and 1 full row. The new element added to the mean vectors must be 0, since the new component wasn't activated at the previous time step. The new covariance matrixes elements must be all 0, except for the new element on the main diagonal, which is initialized as δ^2 (since the gaussian function ranges from 0 to 1, the internal term of equation 14 results in δ).

3.3 Removing Spurious Components

Whenever a component is removed (subsection 2.3), all components means and covariance matrixes must be adjusted by removing the corresponding element. For the covariance matrixes, it means removing 1 full column and 1 full row from the correct positions corresponding to the removed element.

3.4 Recalling

Recalling in the RecIGMN is done in the same way as the IGMN, just that the posterior probabilities are computed based on the new likelihood equations shown in subsection 3.1.

4 Experiments and Results

Experiments both with one-dimensional stochastic and chaotic time-series were performed. The task is to predict the scalar value $\mathbf{a}(t+1)$ given $\mathbf{a}(t)$ (the target value $\mathbf{b}(t)$ is $\mathbf{a}(t+1)$). The RecIGMN was compared to the ESN (Echo State Network), Elman Network (also known as Simple Recurrent Network or SRN [5]) and static IGMN (using only current input to predict the next one). The error measure used was the normalized MSE with respect to the trivial solution (always predicting the latest observation $\mathbf{a}(t)$ for the expected value $\mathbf{a}(t+1)$) and is defined as

$$NMSE_t = \frac{\frac{1}{N}\sum_{t=1}^{N} \|\mathbf{y}(t) - \mathbf{a}(t+1)\|^2}{\frac{1}{N}\sum_{t=1}^{N} \|\mathbf{a}(t) - \mathbf{a}(t+1)\|^2} \tag{22}$$

where N is the number of observations, $\mathbf{a}(t)$ is the observation at time t, $\mathbf{y}(t)$ is the predicted value at time t and $\mathbf{a}(t+1)$ is the desired/target value at time t. It means that solutions worse than the trivial one will have $NMSE_t$ greater than 1, while better solutions will have $NMSE_t$ smaller than 1. The runtime (in seconds) and number of epochs are also informed. The gaussian components information refers to the configuration at the end of training. The parameters of the IGMN based algorithms were the default ones suggested in section 2.1, with ϵ_{max} and δ set to 0.1. v_{min} and sp_{min} were set to 5 and 3, respectively. The ESN output layer was trained with the Conjugate gradient backpropagation with Fletcher-Reeves updates algorithm ('traincgf' in Matlab, which doesn't allow using the Levenberg-Marquardt training algorithm for recurrent networks), with early stopping and default parameters, and this same configuration was used by both layers of the Elman Network. All networks used a hidden layer with 10 neurons, and all input and output layers had size 1, since one-dimensional data was used. The best α parameter of RecIGMN was found for each experiment by evaluating results with α in the interval between 0 and 1 with 0.01 increments. All experiments were averaged over 100 runs and executed on a Intel Core 2 Quad Q8400 with 4GB RAM on Matlab 2009b without parallelization.

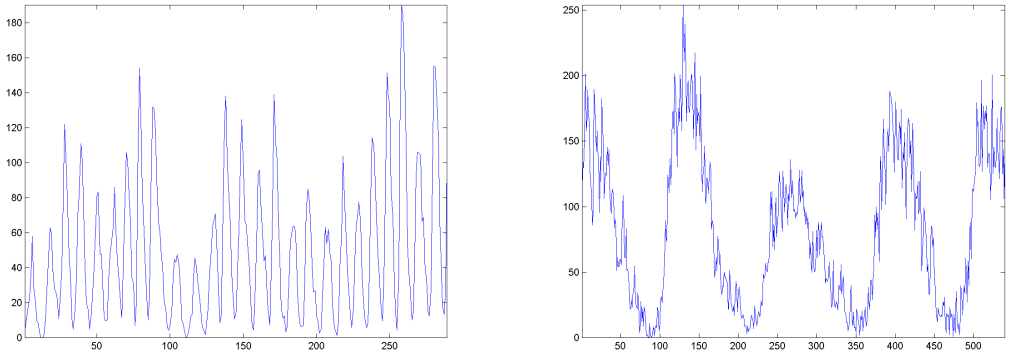
4.1 Yearly Mean Sunspot Numbers

This dataset consists of 289 yearly (mean) observations of a stochastic time-series, which can be seen in figure 4(a). The first 200 observations were used for training, while the remaining 89 were used for testing. The α parameter for the RecIGMN was set to 0.06. For this experiment, $NMSE_t$ = 1 corresponds to $NMSE$ = 0.35 (normalized MSE w.r.t. the observations mean). Results are summarized in table 1 with mean values and standard deviations between parenthesis.

	Elman	ESN	IGMN	RecIGMN
$NMSE_t$	0.97 (0.04)	0.92 (0.02)	0.94 (0.00)	0.72 (0.00)
Epochs	10.97 (5.32)	5.33 (2.26)	1.00 (0.00)	1.00 (0.00)
Runtime	0.98 (0.77)	0.55 (0.45)	0.29 (0.02)	0.46 (0.03)
Gaussian Components	-	-	4.00 (0.00)	6.00 (0.00)

Table 1: Results of the mean yearly sunspot numbers experiment. Mean values outside parenthesis, standard deviations inside.

The RecIGMN algorithm achieved very good results in this experiment, well ahead of its competitors. The static IGMN could solve the problem by exploiting the fact that when the time-series is low, it tends to increase, and vice-versa. Its four gaussian components encoded low, low-medium, high-medium and high values. A similar phenomenon happened in all experiments.



(a) The yearly mean sunspot numbers time-series.

(b) The first 550 data points of the monthly sunspot numbers time-series.

Figure 4: The two flavors of sunspot time-series.

4.2 Monthly Sunspot Numbers

This dataset consists of 2987 monthly observations of a stochastic time-series, which can be seen in figure 4(b). The first 2000 observations were used for training, while the remaining 987 were used for testing. The α parameter for the RecIGMN was set to 0.25. For this experiment, $NMSE_t$ = 1 corresponds to $NMSE$ = 0.1. Results are summarized in table 2.

	Elman	ESN	IGMN	RecIGMN
$NMSE_t$	1.02 (0.08)	0.99 (0.01)	0.99 (0.00)	0.96 (0.00)
Epochs	17.49 (16.79)	6.29 (3.83)	1.00 (0.00)	1.00 (0.00)
Runtime	4.74 (4.27)	1.65 (1.05)	3.03 (0.39)	9.68 (1.09)
Gaussian Components	-	-	4.00 (0.00)	18.00 (0.00)

Table 2: Results of the monthly sunspot numbers experiment.

The results were very similar to the previous ones, except that RecIGMN wasn't too much better this time. A better tuning of the α parameter should be needed to achieve better results.

4.3 Airline Passengers

This dataset consists of 143 monthly observations of a stochastic time-series with seasonal and trend components, which can be seen in figure 5. The first 100 observations were used for training, while the remaining 43 were used for testing. The α parameter for the RecIGMN was set to 0.17. For this experiment, $NMSE_t$ = 1 corresponds to $NMSE$ = 0.0536. Results are summarized in table 3.

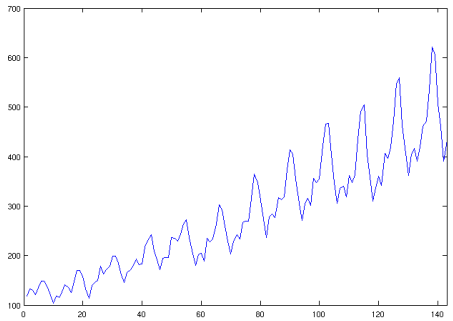


Figure 5: The airline passengers time series.

	Elman	ESN	IGMN	RecIGMN
$NMSE_t$	1.54 (0.67)	1.02 (0.08)	1.63 (0.00)	0.96 (0.00)
Epochs	11.76 (6.22)	6.04 (2.67)	1.00 (0.00)	1.00 (0.00)
Runtime	0.73 (0.57)	0.36 (0.35)	0.08 (0.01)	0.08 (0.01)
Gaussian Components	-	-	3.00 (0.00)	2.00 (0.00)

Table 3: Results of the airline passengers experiment.

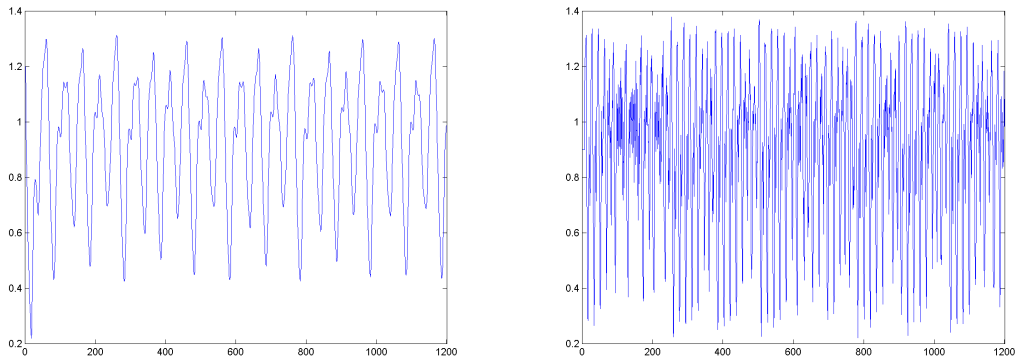
The RecIGMN was the only algorithm to overcome the trivial solution in this task, with excellent speed. It shows that the RecIGMN can handle time series with seasonal and trend components without any data preprocessing.

4.4 Mackey-Glass (τ=17)

This dataset consists of 1201 observations of a chaotic time-series defined by the equation

$$\frac{dx}{dt} = 0.2\frac{x_{t-\tau}}{1 + x_{t-\tau}^n} - 0.1x_t \tag{23}$$

with $\tau = 17$, which can be seen in figure 6(a). The first 1000 observations were used for training, while the remaining 201 were used for testing. The α parameter for the RecIGMN was set to 0.02. For this experiment, $NMSE_t$ = 1 corresponds to $NMSE$ = 0.0196. Results are summarized in table 4.



(a) The Mackey-Glass (τ=17) time-series.

(b) The Mackey-Glass (τ=30) time-series.

Figure 6: The two flavors of Mackey-Glass time-series.

	Elman	ESN	IGMN	RecIGMN
$NMSE_t$	1.06 (0.19)	1.93 (0.18)	0.99 (0.00)	0.27 (0.00)
Epochs	49.81 (31.30)	7.42 (4.60)	1.00 (0.00)	1.00 (0.00)
Runtime	7.79 (5.37)	1.01 (0.84)	1.00 (0.06)	2.60 (0.49)
Gaussian Components	-	-	2.00 (0.00)	5.00 (0.00)

Table 4: Results of the Mackey-Glass (τ = 17) experiment.

Here, RecIGMN achieved the best result by a large margin again. The similarity of this experiment and the yearly mean sunspot one is that both need shorter term memories to succed, while the other two need longer term memories.

4.5 Mackey-Glass (τ=30)

This dataset with 1500 observations is generated by the same equation 23 but with $\tau = 30$, and can be seen in figure 6(b). The first 1000 observations were used for training, while the remaining 500 were used for testing. The α parameter for the RecIGMN was set to 0.04. For this experiment, $NMSE_t$ = 1 corresponds to $NMSE$ = 0.359. Results are summarized in table 5.

	Elman	ESN	IGMN	RecIGMN
$NMSE_t$	0.92 (0.02)	0.94 (0.01)	0.91 (0.00)	0.93 (0.00)
Epochs	29.10 (15.25)	6.39 (4.21)	1.00 (0.00)	1.00 (0.00)
Runtime	4.61 (2.69)	0.87 (0.82)	2.07 (0.18)	3.73 (0.47)
Gaussian Components	-	-	6.00 (0.00)	8.00 (0.00)

Table 5: Results of the Mackey-Glass (τ = 30) experiment.

This is the only experiment where the RecIGMN achieved worse results than its competitors, although not with a statistically significant difference. It was even worse than the static IGMN, meaning that the constructed temporal context was actually misguiding the network. Better parameter tuning would be needed in order to try to remedy this. However, it should be noted that the results are still competitive, and were achieved with just one single scan through data.

5 Conclusions

This work presented the RecIGMN, an one-shot, incremental and spatio-temporal learning algorithm, which has spatio-temporal capabilities.

Five experiments were executed, with 3 stochastic (1 with seasonal and trend components) and 2 chaotic time-series. RecIGMN got the best results in four of them, showing that the IGMN is capable of spatio-temporal learning through self-reference, even in the presence of seasonal and trend components. Parameter tuning for α was needed, however, to correctly set the amount of importance of the context in relation to current inputs.

The worst execution time for RecIGMN was of $\sim$5ms for each one-dimensional data point (plus context values), running on Matlab without any parallelization. This paves the way to real-time learning and prediction of sequences, which can be very useful in robotics, games, reinforcement learning, embedded systems and monitoring tools.

In future works, these other IGMN temporal extensions will be explored:

- Adding tapped delay lines [11]
- Using differentiator-integrator neurons [12];
- Using recurrent connections in various other ways [13]

References

[1] T. Voegtlin. "Recursive self-organizing maps". *Neural Networks*, vol. 15, no. 8-9, pp. 979–991, 2002.

[2] M. Heinen and P. Engel. "An Incremental Probabilistic Neural Network for Regression and Reinforcement Learning Tasks". *Artificial Neural Networks–ICANN 2010*, pp. 170–179, 2010.

[3] M. Heinen. "A Connectionist Approach for Incremental Function Approximation and On-line Tasks". Ph.D. thesis, Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação., 2011.

[4] T. Kolarik and G. Rudorfer. "Time series forecasting using neural networks". In *ACM SIGAPL APL Quote Quad*, volume 25, pp. 86–94. ACM, 1994.

[5] J. Elman. "Finding structure in time* 1". *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[6] P. Werbos. "Backpropagation through time: what it does and how to do it". *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[7] R. Williams and D. Zipser. "A learning algorithm for continually running fully recurrent neural networks". *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[8] H. Jaeger. "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'". Technical report, Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.

[9] A. Dempster, N. Laird, D. Rubin *et al.*. "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[10] P. Engel and M. Heinen. "Incremental learning of multivariate gaussian mixture models". *Advances in Artificial Intelligence–SBIA 2010*, pp. 82–91, 2011.

[11] J. Kangas. "Phoneme recognition using time-dependent versions of self-organizing maps". In *icassp*, pp. 101–104. IEEE, 1991.

[12] L. Moser. "Modelo de um neurônio diferenciador-integrador para representação temporal em arquiteturas conexionistas", 2004. Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação.

[13] R. Pinto. "Um Estudo de Redes Neurais Não-Supervisionadas Temporais", 2010. Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação.