UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RAFAEL COIMBRA PINTO

Online Incremental One-Shot Learning of Temporal Sequences

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Paulo Martins Engel
Advisor

Porto Alegre, December 2011

CIP – CATALOGING-IN-PUBLICATION

Pinto, Rafael Coimbra

Online Incremental One-Shot Learning of Temporal Sequences / Rafael Coimbra Pinto. – Porto Alegre: PPGC da UFRGS, 2011.

106 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2011. Advisor: Paulo Martins Engel.

1. Neural networks. 2. Spatio-temporal Pattern Processing. 3. Gaussian mixtures. 4. Reservoir computing. 5. Time-series prediction. I. Engel, Paulo Martins. II. Título.

"Time is the father of truth, its mother is our mind."
— GIORDANO BRUNO

ACKNOWLEDGEMENTS

Firstly, I thank my family (this includes my girlfriend Cíntia) for always supporting me, even without having any clue about what I do. Then, I would like to thank the people at LIAC (Connectionist Artificial Intelligence Lab) for the long and (often) fruitful discussions and brainstorms, it has been really great. Specifically, I want to thank João Henrique Flores for his patient consulting in statistical matters, without which I would be lost from time to time (actually, sometimes... well, many times, I get even more lost after talking to him, but the huge difference between our fields is the only one to blame). Thanks too to Renato Pereira for (at least unconsciously) spotting that decisive and almost invisible bug in my code, and also for providing inspiration for writing better looking and organized code and, of course, for setting up the group source code repository (it was my first contact with a versioning system and I am enjoying it a lot). Thanks also to Milton Heinen for his never ending "paper writing tips", for his source code, for including me in his "mafia" and for providing the base of this work. I also thank Guilherme Maron (not from LIAC) for pointing me to non-linear time-series analysis tools and some concepts. Many thanks as well to my bosses Doris Fridman and Leonardo Callero at MPF (Ministério Público Federal) for approving my license from work to finish this dissertation, which I hope can be of some use to our daily tasks. Last but not least, I thank my advisor Paulo Engel for letting me work with this wonderful group and for wisely guiding us without restricting or limiting our ideas and, of course, for providing the base of the base of this work. After last, but not lesser too, I thank the II (Informatics Institute) for being such a great place to study and meet smart people!

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

BMU	Best Matching Unit
EM	Expectation-Maximization
ESIGMN	Echo-State Incremental Gaussian Mixture Network
ESN	Echo-State Network
IGMM	Incremental Gaussian Mixture Model
IGMN	Incremental Gaussian Mixture Network
LWR	Locally Weighted Regression
MAD	Median Absolute Deviations from the Median
MIGMN	Merge Incremental Gaussian Mixture Network
MLP	Multi-Layer Perceptron
MSOM	Merge Self-Organizing Map
POMDP	Partially Observable Markov Decision Process
RC	Reservoir Computing
RecIGMN	Recursive Incremental Gaussian Mixture Network
RecSOM	Recursive Self-Organizing Map
SOM	Self-Organizing Map
SRN	Simple Recurrent Network
TDIGMN	Time Delay Incremental Gaussian Mixture Network
TDNN	Time Delay Neural Network

LIST OF SYMBOLS

$\boldsymbol{\mu}$	IGMN component mean vector
$\boldsymbol{\sigma}$	IGMN component standard deviations vector
ϵ_{max}	IGMN parameter used as an error threshold for component creation
δ	IGMN parameter for scaling the initial size of newly created Gaussian components
v_j	Age of the jth Gaussian component
sp_j	Posterior probabilities accumulator of the jth Gaussian component
$p(j)$	Prior probability of the jth Gaussian component
$p(\mathbf{x}\|j)$	Likelihood of input **x** being generated by the distribution of the jth Gaussian component
$p(j\|\mathbf{x})$	Posterior probability for the jth Gaussian component given input **x**
α	MIGMN exponential moving average decay (merging) parameter / RecSOM past-present importance parameter
η	Echo-State Network learning rate parameter
β	MSOM/RecSOM past-present importance parameter
γ	SOM/RecSOM/MSOM learning rate parameter
$\|\lambda\|_{max}$	Maximum absolute eigenvalue (spectral radius) of the reservoir weight matrix
τ	Mackey-Glass time-series parameter

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This work introduces novel neural networks algorithms for online spatio-temporal pattern processing by extending the Incremental Gaussian Mixture Network (IGMN). The IGMN algorithm is an online incremental neural network that learns from a single scan through data by means of an incremental version of the Expectation-Maximization (EM) algorithm combined with locally weighted regression (LWR). Four different approaches are used to give temporal processing capabilities to the IGMN algorithm: time-delay lines (Time-Delay IGMN), a reservoir layer (Echo-State IGMN), exponential moving average of reconstructed input vector (Merge IGMN) and self-referencing (Recursive IGMN). This results in algorithms that are online, incremental, aggressive and have temporal capabilities, and therefore are suitable for tasks with memory or unknown internal states, characterized by continuous non-stopping data-flows, and that require life-long learning while operating and giving predictions without separated stages. The proposed algorithms are compared to other spatio-temporal neural networks in 8 time-series prediction tasks. Two of them show satisfactory performances, generally improving upon existing approaches. A general enhancement for the IGMN algorithm is also described, eliminating one of the algorithm's manually tunable parameters and giving better results.

Keywords: Neural networks, spatio-temporal Pattern Processing, Gaussian mixtures, reservoir computing, time-series prediction.

RESUMO

Este trabalho introduz novos algoritmos de redes neurais para o processamento online de padrões espaço-temporais, estendendo o algoritmo Incremental Gaussian Mixture Network (IGMN). O algoritmo IGMN é uma rede neural online incremental que aprende a partir de uma única passada através de dados por meio de uma versão incremental do algoritmo Expectation-Maximization (EM) combinado com regressão localmente ponderada (Locally Weighted Regression, LWR). Quatro abordagens diferentes são usadas para dar capacidade de processamento temporal para o algoritmo IGMN: linhas de atraso (Time-Delay IGMN), uma camada de reservoir (Echo-State IGMN), média móvel exponencial do vetor de entrada reconstruído (Merge IGMN) e auto-referência (Recursive IGMN). Isso resulta em algoritmos que são online, incrementais, agressivos e têm capacidades temporais e, portanto, são adequados para tarefas com memória ou estados internos desconhecidos, caracterizados por fluxo contínuo ininterrupto de dados, e que exigem operação perpétua provendo previsões sem etapas separadas para aprendizado e execução. Os algoritmos propostos são comparados a outras redes neurais espaço-temporais em 8 tarefas de previsão de séries temporais. Dois deles mostram desempenhos satisfatórios, em geral, superando as abordagens existentes. Uma melhoria geral para o algoritmo IGMN também é descrita, eliminando um dos parâmetros ajustáveis manualmente e provendo melhores resultados.

Palavras-chave: Redes neurais, processamento de padrões espaço-temporais, misturas de Gaussianas, computação por reservoir, previsão de séries temporais.

1 INTRODUCTION

Learning of temporal sequences is a task that applies to areas like time-series prediction and systems control. Sometimes it is necessary for the learned models to be always up to date (aggressive learning) with a constant real-valued data flow (incremental learning) while already in operation (online learning). This is the case with monitoring systems and robotics control, for instance.

Many approaches for learning of temporal data have been proposed, although none of them can cope with all the needs pointed out in the previous paragraph. Markov Chains and Hidden Markov Models (HMMs) (BAUM; PETRIE, 1966) (RABINER; JUANG, 1986) are statistical models in which the transition probabilities between each two states are modeled. To learn these probabilities, HMMs typically use the Baum-Welch algorithm (BAUM et al., 1970), which is not incremental nor online. Markov Chains' parameters can be learned incrementally in an online fashion, but only for discrete variables and using only current observations (unless higher order models are used). Other statistical tools for temporal data include the Auto Regressive Integrated Moving Average (ARIMA) models (BOX; JENKINS, 1994), which make explicit use of past observations in order to make predictions. Usually, non-linear least squares methods, like the Levenberg-Marquardt algorithm (HAGAN; MENHAJ, 1994), or maximum likelihood estimators such as the Kalman Filter (KALMAN, 1960) are used to learn the parameters of ARIMA models, making them unsuitable for aggressive, incremental and online learning. Other approaches include using recurrent neural networks like the Time-Delay Neural Networks (TDNN), Time-Delay Radial Basis Functions Neural Networks (TDRBFNN) (BERTHOLD, 1994), Elman Networks (ELMAN, 1999), NARX networks (LIN et al., 1996), Echo-State Networks (JAEGER, 2001) and Long Short-Term Memories (HOCHREITER; SCHMIDHUBER, 1997). All of them can be trained by online incremental algorithms, but in an iterative, slow manner, requiring many training epochs.

The Incremental Gaussian Mixture Model (IGMM) (ENGEL; HEINEN, 2010) (ENGEL; HEINEN, 2011) and the Incremental Gaussian Mixture Network (IGMN, previously known as IPNN or Incremental Probabilistic Neural Network) (HEINEN; ENGEL, 2010a) (HEINEN; ENGEL, 2011a) (HEINEN; ENGEL; PINTO, 2011) (HEINEN, 2011) were recently proposed as novel neural network based algorithms for clustering and classification/regression, respectively. Those algorithms allow for aggressive online incremental learning, while almost avoiding critical parameter manual tuning.

Albeit being successfully applied to many problems, including robotics control (HEINEN; ENGEL, 2011b) (HEINEN; ENGEL, 2010b), the IGMM and the IGMN are static or purely spatial algorithms, meaning that they can not handle problems with internal states or memory (unless those states are explicitly added by hand). By having a dynamic version of the IGMN algorithm, it would be possible to handle dynamic, nonmarkovian tasks in an online and incremental way, with a single scan through data, which is the goal of this work. There are at least two approaches for reaching this goal, namely, extending online incremental temporal algorithms (like recurrent neural networks) for aggressive learning, or extending aggressive online incremental algorithms for temporal processing, which is the approach used here.

1.1 Main Contributions

The present work proposes a few spatio-temporal variants for the IGMN algorithm. Since the IGMM can be seen essentially as an advanced clustering algorithm, it is expected that already proven techniques for extending clustering algorithms (specially the Self-Organizing Map) to the temporal domain should work too with the IGMM, and thus with the IGMN. In this sense, three novel algorithms are proposed: Echo-State IGMN (ESIGMN), Merge IGMN (MIGMN) and Recursive IGMN (RecIGMN). Also, the IGMN with time-delays is evaluated against them and other classic algorithms, and is called Time-Delay IGMN (TDIGMN) here. Besides the novel algorithms for temporally extending the static IGMN, a new technique is also introduced to improve it even in its static form, which also removes one of its parameters that, otherwise, should be manually selected. Summarizing the contributions:

- Echo-State Incremental Gaussian Mixture Network (ESIGMN)
- Merge Incremental Gaussian Mixture Network (MIGMN)
- Recursive Incremental Gaussian Mixture Network (RecIGMN)
- Outlier-based Component Creation

1.2 Dissertation Structure

This work will start by describing related algorithms in chapter 2, which will serve as the foundations for the algorithms proposed in this work, as well as to form a basis for comparison in the experiments later. This chapter is divided into sections for static (2.1) and temporal (2.2) algorithms. Then, moving to the proposed algorithms of this work, chapter 3 presents the TDNN, ESIGMN, MIGMN and RecIGMN algorithms, as well as the new component creation rule. These algorithms will be compared among themselves and with some related supervised temporal neural networks in various temporal tasks in chapter 4. Chapter 5 finishes this work with a short revision of achieved results, final thoughts and plans for future works.

2 RELATED WORKS

The IGMN, albeit being a supervised algorithm, is very akin to unsupervised neural networks, due to its origin in the IGMM algorithm. This makes it more suitable for temporal extensions based on the ones already proposed for unsupervised neural networks. Having this in mind, it is interesting to analyze different proposed temporal neural networks, both supervised and unsupervised, in order to adopt similar strategies for extending the IGMN temporally.

2.1 Static Algorithms

The algorithms presented in this section are purely spatial, and are explained as requirements for their spatio-temporal variants to be seen in section 2.2.

2.1.1 Self-Organizing Map (SOM)

The SOM (KOHONEN, 1982) is an unsupervised neural network, meaning it does not learn from input-output example pairs, instead just learning the structure of the input space, in this case in the form of data clusters. It consists of a single competitive layer (besides the input layer) neural network, and this layer is spatially organized. It is usually a 2D lattice (could be 1D, 3D or any dimension) and, therefore, the relative position of the neurons is important, in contrast to more conventional neural networks (a 2D input space representation in this architecture can be seen in figure 2.1). The weights from the input layer to the competitive layer represent the learned input vectors' prototypes and, therefore, have the same dimensionality as the input space. The input vectors are presented one-by-one to the algorithm, which finds the most similar prototype so far,

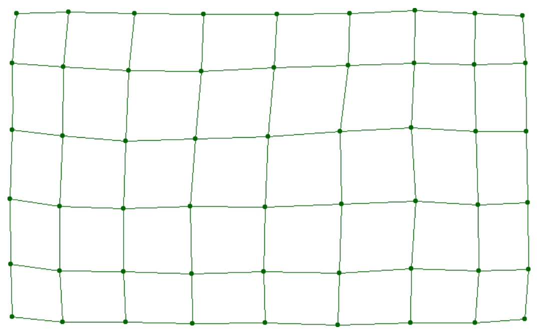


Figure 2.1: SOM in an advanced training state, with bidimensional inputs taken from an uniform distribution. Each node represents a data cluster, and the edges represent the neighborhood relations between them.

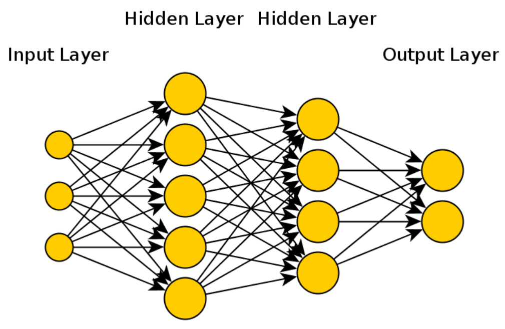


Figure 2.2: An example of multilayer perceptron architecture with 3 inputs, 2 hidden layers with 5 and 4 neurons respectively and 2 output neurons.

based on a distance metric (usually Euclidian or Manhatan distance). The best matching neuron is then selected as the winner neuron or best matching unit (BMU), according to the following equation:

$$b(t) = \arg\min_{i \in V_O} \{\|\mathbf{x}(t) - \mathbf{w}_i\|\} \tag{2.1}$$

where $\mathbf{x}(t)$ is the input vector at time t, $b(t)$ is the index (position) of the winner neuron at time t in the output / competitive layer space, $\mathbf{w}_i(t)$ is a prototype to be compared to the input, also at time t. After finding the winner neuron, its corresponding weights $\mathbf{w}_b$ (its prototype vector elements) are adjusted, as well as the weights of its neighbors, according to the following update rule:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \gamma(t) h_{ib}(t)(\mathbf{x}(t) - \mathbf{w}_i(t)) \tag{2.2}$$

where γ is a learning rate between 0 and 1 and h_{ib} is a neighborhood function such as:

$$h_{ib}(t) = \exp\left(\frac{-\|\mathbf{I}_i - \mathbf{I}_b\|^2}{2\sigma(t)^2}\right) \tag{2.3}$$

where $\mathbf{I}_i$ and $\mathbf{I}_b$ are neurons *i* and *b* indexes (positions) on the competitive layer, and $\sigma(t)$ is the Gaussian standard deviation at time t. Note that γ and σ are time dependent, and are usually implemented with some decay during the algorithm's execution time.

2.1.2 Multi-Layer Perceptron

The multilayer perceptron (MLP) (RUMELHART; MCCLELLAND, 1986) is a supervised feedforward artificial neural network model that learns non-linear mappings from input vectors to output vectors (discrete or continuous). It is composed by multiple layers of non-linear neurons (computational units) which propagate signals until they reach the output layer. Each neuron computes the scalar product between its inputs and its incoming weights and then applies some non-linear transformation, usually a sigmoid function. Consecutive layers are fully connected and the number of neurons in each of them (as well as the number of layers) must be manually specified (other similar neural networks like Cascade Correlation (FAHLMAN, 1990) and NEAT (STANLEY; MIIKKULAINEN, 2001) can learn the number of layers and neurons as well). This architecture can be seen in figure 2.2.

Once the signal reaches the output layer, the error of the computed values in relation to the given output example (the target values) is back-propagated in order to update the network weights. This training procedure can be performed in various ways, but here we will talk about the Stochastic Gradient Descent (SGD) procedure, since it can be used for online learning, i.e. updating the model after each example pair is presented to the network. The specialized version of SGD for multilayer perceptrons is called Stochastic Backpropagation (WERBOS, 1974). Backpropagation tries to minimize a cost function, usually the mean squared error, in relation to the network weights between the neurons. Unlike single layer neural networks (e.g. perceptron, adaline), there is no guarantee of finding the global minimum, just local minima, since the error surface is not convex. Also, this kind of training procedure requires many epochs for convergence, i.e. scanning the entire dataset many times for reducing error at small steps.

It is also proven that multilayer perceptrons are universal function approximators (HORNIK MAXWELL; WHITE, 1989) (CYBENKO, 1989), given necessary number of neurons. Nevertheless, it is not guaranteed that backpropagation can find those approximations even with enough neurons.

2.1.3 Incremental Gaussian Mixture Network

The IGMN (Incremental Gaussian Mixture Network) is a supervised algorithm (HEINEN; ENGEL, 2010c) that uses an incremental approximation of the EM algorithm (DEMPSTER et al., 1977), the IGMM (Incremental Gaussian Mixture Model) (ENGEL; HEINEN, 2011). 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 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, like with autoassociative neural networks (RUMELHART; MCCLELLAND, 1986)). This architecture is depicted on figure 2.3. Next subsections describe the algorithm in more detail.

2.1.3.1 Learning

The algorithm starts with no components, which are created as necessary (see subsection 2.1.3.2). Given input $\mathbf{x}$, the IGMN algorithm processing step is as follows. First, the likelihood for each component j is computed:

$$\bar{p}(\mathbf{x}|j) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\mathbf{C}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right) \tag{2.4}$$

$$p(\mathbf{x}|j) = \frac{\bar{p}(\mathbf{x}|j)}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_j|}} \tag{2.5}$$

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:

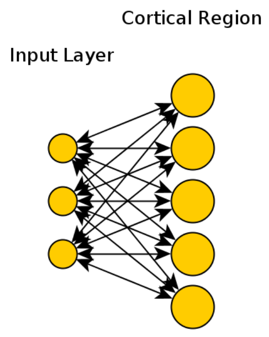


Figure 2.3: An example of IGMN with 3 input nodes and 5 Gaussian components. Any input element can be predicted by using any other element, which means that the input vector can actually be divided into input and output elements.

$$p(j|\mathbf{x}) = \frac{p(\mathbf{x}|j)p(j)}{\sum\limits_{q=1}^{M} p(\mathbf{x}|q)p(q)} \qquad \forall j \tag{2.6}$$

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

$$v_j(t) = v_j(t-1) + 1 \tag{2.7}$$

$$sp_j(t) = sp_j(t-1) + p(j|\mathbf{x}) \tag{2.8}$$

$$\mathbf{e}_j = \mathbf{x} - \boldsymbol{\mu}_j \tag{2.9}$$

$$\omega_j = \frac{p(j|\mathbf{x})}{sp_j} \tag{2.10}$$

$$\Delta\boldsymbol{\mu}_j = \omega_j \mathbf{e}_j \tag{2.11}$$

$$\boldsymbol{\mu}_j(t) = \boldsymbol{\mu}_j(t-1) + \Delta\boldsymbol{\mu}_j \tag{2.12}$$

$$\mathbf{C}_j(t) = \mathbf{C}_j(t-1) - \Delta\boldsymbol{\mu}_j \Delta\boldsymbol{\mu}_j^T + \omega \left[\mathbf{e}\mathbf{e}^T - \mathbf{C}_j(t-1)\right] \tag{2.13}$$

$$p(j) = \frac{sp_j}{\sum\limits_{q=1}^{M} sp_q} \tag{2.14}$$

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

2.1.3.2 Creating New Components

In order to create new components, the cortical region must reconstruct its input $\mathbf{x}$ based on the posterior probabilities obtained in equation 2.6. The reconstruction of the unknown elements $\mathbf{x}_t$ is obtained by the following equation:

$$\hat{\mathbf{x}}_t = \sum_{j=1}^{M} p(j|\mathbf{x}_i)\boldsymbol{\mu}_{j,t} \tag{2.15}$$

in the naïve approach (only diagonal covariance matrixes are used), where $\mathbf{x}_i$ is the input vector without the unknown elements and $\boldsymbol{\mu}_{j,t}$ is the jth component mean without the unknown elements. Note that it is only an average of the region's means weighted by their posterior probabilities. This approach will be called here IGMNn (naïve). The full multivariate version (IGMN) is as follows:

$$\hat{\mathbf{x}}_t = \sum_{j=1}^{M} p(j|\mathbf{x}_i)(\boldsymbol{\mu}_{j,t} + \mathbf{C}_{j,ti}\mathbf{C}_{j,i}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_{j,i})) \tag{2.16}$$

where $\mathbf{C}_{j,ti}$ is the submatrix of the jth component covariance matrix associating the unknown and known parts of the data, $\mathbf{C}_{j,i}$ is the submatrix corresponding to the known part only and $\boldsymbol{\mu}_{j,i}$ is the jth's component mean without the unknown element.

After reconstructing the input, the reconstruction error can be obtained by:

$$\epsilon = \max_{i \in D}\left[\frac{(x_i - \hat{x}_i)^2}{max(x_i) - min(x_i)}\right] \tag{2.17}$$

Note that $max(x_i) - min(x_i)$ is simply the range of dimension i in the dataset and is used here for the purpose of rescaling the error. This can be just an estimate (since the algorithm is incremental, the true range may not be available). If there are no components or ϵ is greater than a manually chosen threshold ϵ_{max} (e.g., 0.1), then a new component j is created and initialized as follows:

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

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

$$\sigma_{ini} = diag(\delta[max(\mathbf{x}) - min(\mathbf{x})]) \tag{2.18}$$

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.

2.1.3.3 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 2.14. In other words, each component is given some time v_{min} to show its importance to the model in the form of an accumulation of its posterior probabilities sp_j.

2.1.3.4 Recalling

In IGMN, any element can be predicted by any other element. This is done by reconstructing data from the target elements ($\mathbf{x}_t$) by estimating the posterior probabilities using only the given elements, as follows:

$$p(j|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|j)p(j)}{\sum\limits_{q=1}^{M} p(\mathbf{x}_i|q)p(q)} \qquad \forall j \tag{2.19}$$

It is similar to equation 2.6, except that it uses a modified input vector $\mathbf{x}_i$ with the target elements $\mathbf{x}_t$ removed from calculations. After that, $\mathbf{x}_t$ can be reconstructed using equation 2.15 or 2.16.

2.1.3.5 Confidence Intervals

Besides being able to estimate any missing values in its input, the IGMN can also estimate their variances and covariances. This is useful for estimating confidence intervals when using the algorithm for decision making. For a given input $[\mathbf{x}_i; \mathbf{x}_t]$, the variances for $\mathbf{x}_t$ can be estimated as follows:

$$\hat{\boldsymbol{\sigma}}_t^2 = \sum_{j=1}^{M} p(j|\mathbf{x}_i)(\boldsymbol{\sigma}_j^t + \|\boldsymbol{\mu}_j^t - \hat{\mathbf{x}}_t\|^2) \tag{2.20}$$

for the naïve case, where $\hat{\mathbf{x}}_t$ is the estimated output, and

$$\hat{\mathbf{C}}_t = \sum_{j=1}^{M} p(j|\mathbf{x}_i)(\mathbf{C}_j^{tt} - \mathbf{C}_j^{ti}\mathbf{C}_j^{ii-1}\mathbf{C}_j^{tiT} + \|\bar{\mathbf{x}}_j^t - \hat{\mathbf{x}}_t\|^2) \tag{2.21}$$

where $\mathbf{C}^{tt}$ is the covariance matrix portion corresponding to the outputs, $\mathbf{C}^{ti}$ corresponds to the portion corresponding to the covariances between inputs and outputs, $\mathbf{C}^{ii-1}$ corresponds to the input portion and $\bar{\mathbf{x}}^t$ is given as follows:

$$\bar{\mathbf{x}}_j^t = \boldsymbol{\mu}_j^t + \mathbf{C}_j^{ti}\mathbf{C}_j^{ii-1}(\mathbf{x}_j^i - \boldsymbol{\mu}_j^i) \tag{2.22}$$

then the confidence margins are given by $\hat{\mathbf{x}}_t \pm d\hat{\boldsymbol{\sigma}}_t$ (in the multivariate case, $\hat{\boldsymbol{\sigma}}$ is the diagonal of $\hat{\mathbf{C}}$), where d is the number of standard deviations (e.g., 1.96 for 95% confidence). $\hat{\boldsymbol{\sigma}}_i$ can be obtained similarly by just swapping input and output portions in previous equations.

2.2 Temporal Algorithms

The algorithms presented in this chapter have spatio-temporal capabilities, some of them being temporal counterparts of the ones presented in the previous section.

2.2.1 Time-Delay Neural Network (TDNN)

The time delay neural network (TDNN) extends the simple multilayer perceptron with a sliding window in its input. It means that the last input signals are stored and presented to the neural network with current input altogether as an augmented input. How many past inputs are stored is a design choice, but some techniques from classical statistics could be used for this decision, like the partial autocorrelation function (BOX; JENKINS,

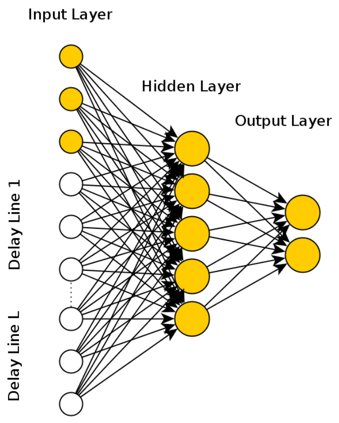


Figure 2.4: Example TDNN architecture with originally 3 inputs, plus L delay lines, each with a past copy of the inputs (empty nodes). It can be seen as a $3(L+1)$ inputs MLP.

1994). This forms a kind of exact short-term memory (could be long-term, but the increase in complexity may render it impractical, especially for high dimensional input spaces). Other than that, the algorithm is exactly the same as the multilayer perceptron and can use all training algorithms and optimizations available to it. An example architecture for a TDNN can be seen in figure 2.4.

2.2.2 Elman Network

The Elman Network (ELMAN, 1999) also augments the MLP inputs with a kind of temporal context, but instead of using past copies of the input in a sliding window like the TDNN, it stores just the last activation vector from the hidden layer. Therefore, it produces a self-referencing temporal context by learning its own internal state. It can also be seen as a MLP where its inputs are augmented with the previous hidden neuron activations, and the same training algorithms and optimizations apply. Another possible view is that of a dynamical system where the hidden layer is the internal state, which is affected by past states and external inputs, and some output signal extracted from the state. In fact, Extended Kalman Filter (which is used for doing exact inference in dynamical systems) can be used to train an Elman Network (WILLIAMS, 1992). Figure 2.5 shows a possible architecture for an Elman Network.

Other learning algorithms that can be used with the Elman Networks include Backpropagation Through Time (BPTT) (WERBOS, 1974), in which the architecture is unfolded in time and trained by conventional backpropagation, and Real-Time Recurrent Learning (RTRL) (WILLIAMS; ZIPSER, 1989), in which the gradients of the internal state (temporal context neurons) in relation to their incoming weights are computed and used in the update equations, in order to avoid unfolding the network.

2.2.3 Echo-State Network (ESN)

Reservoir Computing (RC) (LUKOŠEVIČIUS; JAEGER, 2009) is a recently coined term for a (not so recent) neural pattern processing paradigm where a random, non-linear, fixed and large hidden layer with recurrent connections, called a reservoir, is used as an excitable medium where interesting dynamic features of the data stream can be extracted. It is similar to a random filter bank, producing transformations over the input data. Albeit the fixed reservoir weights, its output states are sufficient to successfully train linear

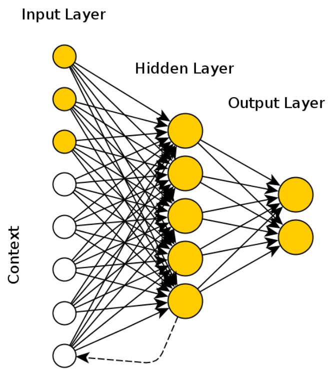


Figure 2.5: Example Elman Network architecture with originally 3 inputs augmented by hidden layer (with 5 neurons) activations from the previous time-step (empty nodes). It can be seen as an 8 inputs MLP.

regression / classification algorithms on non-linear dynamic tasks, thus potentially turning any static linear algorithm into a non-linear dynamic one. But since the reservoir is random, large reservoirs are required, to increase the chances of obtaining useful transformations.

This paradigm was found independently by different researchers at different time points, also in distinct research fields like computational neuroscience and machine learning: Temporal Recurrent Neural Network (DOMINEY, 1995); Liquid State Machines (NATSCHLÄGER; MAASS; MARKRAM, 2002); Echo State Networks (JAEGER, 2001); Decorrelation-Backpropagation Learning (STEIL, 2004).

This work will incorporate the ESN as the base for the new algorithm, since it is composed by the default neuron model used in artificial neural networks (like the Multi-Layer Perceptron), is very simple to implement, is probably the most widely used RC artificial neural network in computer science and gave excellent results in previous works, e.g., predicting chaotic dynamics (three orders of magnitude improved accuracy (JAEGER; HAAS, 2004)), nonlinear wireless channel equalization (two orders of magnitude improvement (JAEGER; HAAS, 2004)), the Japanese Vowel benchmark (zero test error rate, previous best was 1.8% (JAEGER et al., 2007)), financial forecasting (winner of the international forecasting competition NN3), and in isolated spoken digits recognition (improvement of word error rate on benchmark from 0.6% of previous best system to 0.2% (VERSTRAETEN; SCHRAUWEN; STROOBANDT, 2006)).

The ESN consists basically of an input layer, a reservoir and an output layer. The weights between input layer and reservoir (here denoted as $\mathbf{W}_{in}$) as well as the recurrent reservoir weights ($\mathbf{W}$) are randomly chosen and fixed – no training is necessary. The weights from input layer and reservoir to the output layer ($\mathbf{W}_{out}$) are trained in batch mode by linear Least Squares Fitting, or in incremental mode by Recursive Least Squares (RLS) or any other incremental linear learning algorithm, like conventional Least Mean Squares (LMS). This work will focus on the incremental mode, since we need to compare the ESN to an incremental algorithm (ESIGMN). The reservoir activation function ($f(.)$) is the hyperbolic tangent, and the output layer activation function ($g(.)$) is usually the identity, although any differentiable function could be used. An example of an ESN can be seen in figure 2.6.

In order to have a stable reservoir, however, the echo state property must be assured. It means that when a null vector is continually fed into the reservoir as its input, its state

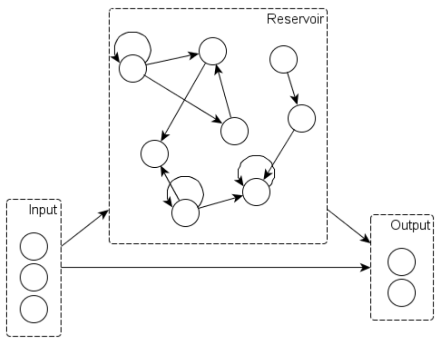


Figure 2.6: An example of Echo State Network with 3 input nodes, 2 output nodes and 8 reservoir nodes. All weights going to the output layer ($\mathbf{W}_{out}$) must be trained, while every other weight is fixed.

vector must decay to a null vector too, as time tends towards infinity. In practical terms, this can be ensured by rescaling the largest absolute eigenvalue $|\lambda|_{max}$ of the reservoir recurrent weight matrix **W**, i.e. its spectral radius, to a value in the interval (0, 1). There is controversy about the necessity or sufficiency of this condition to ensure the echo state property (BUEHNER; YOUNG, 2006), but it is known to work well in practice (JAEGER, 2001). A proven and stronger sufficient condition is to rescale the largest singular value of the weight matrix to the interval (0, 1) (JAEGER, 2001).

Following, a brief explanation of the ESN algorithm is shown. Given input **x** at time t, a processing step of the ESN is computed as follows:

The new state **s** of the reservoir at time t is obtained by the following equation:

$$\mathbf{s}(t) = f(\mathbf{W}_{in}\mathbf{x}(t) + \mathbf{W}\mathbf{s}(t-1)) \tag{2.23}$$

while the output **y** at time t of the ESN is given by (assuming identity activation function):

$$\mathbf{y}(t) = \mathbf{W}_{out}[\mathbf{x}(t); \mathbf{s}(t)] \tag{2.24}$$

where $[.;.]$ is the vector concatenation operation. The prediction error **e** at time t is given by:

$$\mathbf{e}(t) = \mathbf{d}(t) - \mathbf{y}(t) \tag{2.25}$$

where $\mathbf{d}(t)$ is the target vector (teacher signal) at time t. The output weights $\mathbf{W}_{out}$ are modified, in the online case, by stochastic gradient descent as follows:

$$\Delta\mathbf{W}_{out} = \eta\mathbf{e}(t)[\mathbf{x}(t); \mathbf{s}(t)]^T \tag{2.26}$$

where η is a learning rate in the interval [0, 1].

The ESN can also be seen as an Elman Network with fixed input-hidden and hidden-hidden weights and shortcut input-output connections. The kind of temporal context used by the ESN (the reservoir) is the base of the ESIGMN algorithm, to be seen in section 3.3.

2.2.4 RecSOM

The temporal context of RecSOM (VOEGTLIN, 2002) is inspired by the Elman Network (section 2.2.2). Its temporal capabilities are achieved through feedbacks from the hidden layer itself, processing it together with the input signal. To transpose that idea

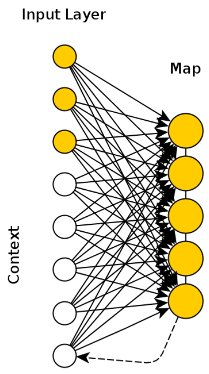


Figure 2.7: Example RecSOM architecture with 3 inputs and an 1-dimensional map with 5 neurons.

from the supervised to the unsupervised domain, it is enough to remove the output layer and see the SOM as the hidden layer, as is shown in figure 2.7.

Then, the RecSOM is capable of self-organizing its own neuron activity, which is now its temporal context. This capability gives it a potentially long-term memory without ambiguities (there is no combination of signals). In fact, it is possible to store arbitrary sequences of any finite length, given sufficient neurons (STRICKERT; HAMMER; BLOHM, 2005). However, for the algorithm to be stable, it is necessary to apply an activation function at the map outputs, one possibility suggested by Voegtlin being:

$$y_i(t+1) = \exp\{-\alpha\|\mathbf{x}(t) - \mathbf{w}_i^x(t)\|^2 - \beta\|\mathbf{y}(t) - \mathbf{w}_i^y(t)\|^2\} \tag{2.27}$$

where $y_i(t)$ is the activation of neuron i at time t, α is a constant between 0 and 1 which defines the contribution from the current input pattern $\mathbf{x}(t)$ to the activation, $\mathbf{w}_i^x(t)$ is the prototype (mean) vector of neuron i in relation to input patterns, β is a constant between 0 and 1 which defines the contribution of the current context and $\mathbf{w}_i^y(t)$ is the prototype vector of neuron i in relation to the context space. Therefore, in the RecSOM algorithm, each neuron has 2 prototypes instead of 1: the input prototype and the context prototype. The new equations of winner selection and weight updates are as follows:

$$b(t+1) = \arg\max_{i \in V_O}\{y_i(t+1)\} \tag{2.28}$$

$$\mathbf{w}_i^x(t+1) = \mathbf{w}_i^x(t) + \gamma(t)h_{ib}(t)(\mathbf{x}(t) - \mathbf{w}_i^x(t)) \tag{2.29}$$

$$\mathbf{w}_i^y(t+1) = \mathbf{w}_i^y(t) + \gamma(t)h_{ib}(t)(\mathbf{y}(t) - \mathbf{w}_i^y(t)) \tag{2.30}$$

So, for each input vector presented, not only the most similar input prototype is verified, but also the current map context (previous time step activations) must be compared with the context prototypes of the neurons. Therefore, each neuron defines a pattern referring to the input space and a context referring to the map space. It means that some identical input pattern could select different winner neurons, depending on the input history.

The RecSOM could be implemented as a conventional SOM by concatenating input and context, but in this case the capability of weighting the input and context contributions with the α and β parameters is lost.

This kind of temporal context is the base for the RecIGMN algorithm to be seen in section 3.5, which is also implemented without the weighting parameters as suggested in the previous paragraph.

2.2.5 MSOM

The RecSOM is a very powerful temporal SOM variant, but its time complexity is very high, due to the huge number of additional inputs and prototype sizes (the context) equal to the number of neurons in the map. The MSOM (STRICKERT; HAMMER, 2005) can be seen as a compressed version of the RecSOM, with lower time complexity. Its temporal context is given by an exponential moving average of the winning neurons' prototypes over time, and thus has the same dimensionality as the input space. The new equations for this algorithm are as follows:

$$\mathbf{C}(t) = \alpha \mathbf{w}_b^c(t-1) + \beta \mathbf{w}_b^x(t-1) \tag{2.31}$$

$$y_i(t+1) = \exp\{-(1-\eta)\|\mathbf{x}(t) - \mathbf{w}_i^x(t)\|^2 - \eta\|\mathbf{C}(t) - \mathbf{w}_i^c(t)\|^2\} \tag{2.32}$$

$$b(t) = \arg\max_{i \in V_O}\{y_i(t+1)\} \tag{2.33}$$

$$\mathbf{w}_i^x(t+1) = \mathbf{w}_i^x(t) + \gamma(t)h_{ib}(t)(\mathbf{x}(t) - \mathbf{w}_i^x(t)) \tag{2.34}$$

$$\mathbf{w}_i^c(t+1) = \mathbf{w}_i^c(t) + \gamma(t)h_{ib}(t)(\mathbf{C}(t) - \mathbf{w}_i^c(t)) \tag{2.35}$$

Where α and β now weight the last winning neuron weight vectors (context and input prototypes, $\mathbf{w}_b^c$ and $\mathbf{w}_b^x$, respectively) in order to obtain a context vector $\mathbf{C}$. η plays the role of the α and β parameters in the RecSOM algorithm, weighting the present in relation to the past, but now using only a single parameter and its complement.

This kind of temporal context is the base for the MIGMN algorithm, to be seen in section 3.4.

3 ENHANCING THE INCREMENTAL GAUSSIAN MIXTURE NETWORK

In this chapter, four temporal extensions for the IGMN algorithm are presented. Each one has a different kind of temporal context with their own advantages and disadvantages: sliding windows, reservoir layer, exponential moving average and self-referencing. All of them introduce relatively small modifications to the original IGMN (section 2.1.3), so only the differences will be highlighted. Also, as they are inspired by temporal extensions for other algorithms, section 2.2 already covers those approaches in more detail. But before presenting the temporal extensions, an improvement which applies to both the static and temporal versions of IGMN is presented in the next section.

3.1 Outlier-Based Component Creation

The default component creation rule in the IGMN algorithm has its drawbacks. First, it needs the ϵ_{max} parameter to be set manually beforehand. Second, this parameter is used as a fraction of the estimated data range, resulting in an error threshold, but the estimated data range may not be a good starting point for defining maximum acceptable errors. This can be observed by trying to approximate the following function:

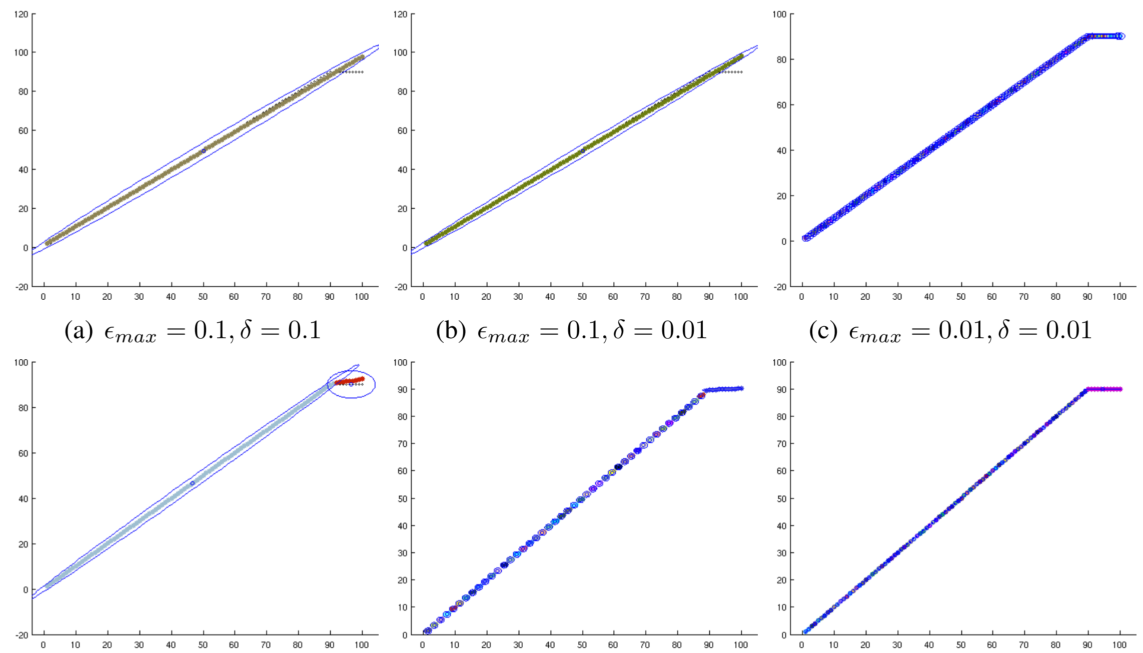
$$f(x) = \begin{cases} x & : x < 90 \\ 90 & : x >= 90 \\ \forall x \in [1; 100] \end{cases} \tag{3.1}$$

When the IGMN algorithm reaches point $(91, 90)$ (and all subsequent points), the maximum normalized (by range) error is not sufficient to trigger component creation for ϵ_{max} (it is less than 0.1). The result can be seen in figure 3.1(a). Figure 3.1(b) shows that changing the δ parameter is not sufficient to avoid this problem. In 3.1(c) the problem is solved, but 1 component is created for each point in the dataset. It happens because the ϵ_{max} parameter controls component creation by assuming the same error threshold for the entire dataset, while we clearly need it to be adaptive.

The proposed solution is to use the IGMN error margin (section 2.1.3.5) as an adaptive threshold, thus creating a new component whenever a new data point is considered to be an outlier (we call an "outlier" any point which lies outside the error margins). Thus, the component creation condition becomes

$$\exists i, \frac{|x_i - \hat{x}_i|}{max(x_i) - min(x_i)} > d\sigma_i \tag{3.2}$$

where the left part is the rescaled absolute error of a single element of input vector **x** in



(a) $\epsilon_{max} = 0.1, \delta = 0.1$ (b) $\epsilon_{max} = 0.1, \delta = 0.01$ (c) $\epsilon_{max} = 0.01, \delta = 0.01$

(d) $\delta = 0.1$ (e) $\delta = 0.01$ (f) $\delta = 0.001$

Figure 3.1: Top: IGMN with default component creation rule. Bottom: IGMN with outlier-based component creation.

relation to its reconstructed value (equation 2.15 or 2.16) and the right part is the margin obtained by equation 2.20 or 2.21. In other words, whenever any element of **x** has an absolute error greater than the margin value, a component is created. It changes along the dataset according to the size of each component and hence it can solve that problem (and others) more efficiently, as can be seen in figures 3.1(d), 3.1(e) and 3.1(f). With $\delta = 0.1$, only 2 components are created, which is intuitively expected since the function is composed by only 2 straight lines. This behavior persists even when we decrease δ and just changes when it reaches 0.01, when 45 components are created (still less than half the components created with the same value of δ by the default rule). Increasing δ upto 0.2 also keeps the same behavior. With $\delta = 0.001$ it still produces less components (90) than the default rule with $\delta = 0.01$, as it continues to use just 1 component for the final portion of the function.

Thus, using outlier-based component creation has at least 2 advantages over the default rule: first, it eliminates the need to set the ϵ_{max} parameter. Second, it can adapt to datasets where the data distribution changes within different regions.

3.2 Time-Delay Incremental Gaussian Mixture Network (TDIGMN)

This IGMN variant was already presented in (HEINEN, 2011), but it was seen as just an IGMN with extra inputs. Here we will explicitly distinguish it from the static IGMN by calling it TDIGMN. The TDIGMN, as the TDNN, uses a sliding window of its inputs (a buffer) as a temporal context, allowing it to use information from past steps (as long as the size of the window) in posterior computations. It is a time-limited context, but has perfect memory inside the sliding window range. Besides its limited range nature, another problem with this kind of approach is the choice of sliding window, which can be defined by its lag size l:

$$\hat{\mathbf{x}}(t) = [\mathbf{x}(t); \mathbf{x}(t-1); ...; \mathbf{x}(t-l)] \tag{3.3}$$

where $\hat{\mathbf{x}}$ is the augmented input vector. As in classical autoregressive techniques, the partial autocorrelation function could be used to choose a reasonable value for l (BOX; JENKINS, 1994), but this sliding window does not even need to be built from consecutive values (as with seasonal approaches), leaving it open for yet more fine tuning. Trying to alleviate the limited range and sliding window choice problems, 3 new temporal extensions are proposed in the next sections, using contexts with unbounded (albeit inexact) memory range.

3.3 Echo-State Incremental Gaussian Mixture Network (ESIGMN)

The ESIGMN (PINTO; ENGEL; HEINEN, 2011a) augments the IGMN with an ESN-style reservoir between its input and cortical region. The reservoir is responsible for mapping the input space into a feature space which captures temporal dynamics of the data, as with the ESN. But instead of feeding the input and the reservoir state into a linear output layer, they are fed into an unmodified IGMN in the ESIGMN algorithm. The IGMN does its usual spatial processing, but its inputs already incorporate temporal information. Feedback connections from the outputs to the reservoir are also possible, but they are not explored in this work. This architecture can be seen in figure 3.2.

Therefore, all IGMN equations from section 2.1.3 apply, with a difference only in the source of the IGMN input. In ESIGMN, the input data $\mathbf{x}$ is divided into input $\mathbf{x}_i$ and output/target elements $\mathbf{x}_t$, and is processed in the following way:

$$\mathbf{s}(t) = f(\mathbf{W}_{in,r}\mathbf{x}_i(t) + \mathbf{W}\mathbf{s}(t-1)) \tag{3.4}$$

$$\bar{\mathbf{x}}(t) = [\mathbf{s}(t); \mathbf{x}(t)] \tag{3.5}$$

where $\mathbf{s}$ is the reservoir state, $\mathbf{x}_i$ is the known part of the input (the actual input portion of the data), excluding the target values $\mathbf{x}_t$, and $\mathbf{W}_{in,r}$ are the input weights (without the target values) to the reservoir (equation 3.4 is analogous to equation 2.23 for the ESN). The IGMN is trained by receiving $\bar{\mathbf{x}}$ (the concatenation of the full input and reservoir state) as its input. By omitting $\mathbf{x}_t$ and using equation 2.19 for recalling, it is possible to predict the output $\mathbf{x}_t$.

3.4 Merge Incremental Gaussian Mixture Network (MIGMN)

The MIGMN is inspired by the MSOM using an exponential moving average of the network reconstructed input as its temporal context. This context is computed as follows:

$$\mathbf{c}(t) = \alpha\mathbf{c}(t-1) + (1-\alpha)\hat{\mathbf{x}}(t) \tag{3.6}$$

where $\hat{\mathbf{x}}$ is the reconstructed full input (including the portion used as a target/output) obtained from equation 2.16, α is a merging parameter between 0 and 1 and $\mathbf{c}(0) = \mathbf{0}_D$ (the null vector with dimension D, the same as the input dimension). Hence, the context dimensionality in the MIGMN is equal to the input dimensionality. In other words, the input size doubles from the point-of-view of the static IGMN (which can be very onerous

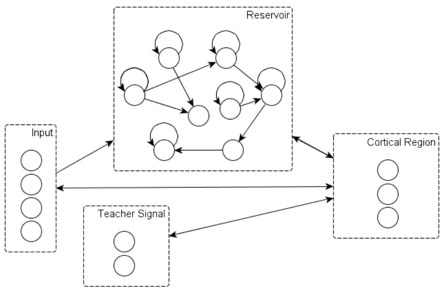


Figure 3.2: An example of ESIGMN with 4 inputs ($\mathbf{x}_i$), 2 targets/outputs ($\mathbf{x}_t$), 3 Gaussian components and 8 reservoir neurons. Only the actual input portion of the data is used to update the reservoir, while the full input data (with target values / teacher signal) is fed into the IGMN together with the reservoir state **s** in order to train it. By omitting $\mathbf{x}_t$, the IGMN can be used in recalling mode to predict $\mathbf{x}_t$.

for high dimensionalities, due to the matrix inversions inside the IGMN), since the context is concatenated to the original input:

$$\bar{\mathbf{x}}(t) = [\mathbf{c}(t-1); \mathbf{x}(t)] \tag{3.7}$$

where $\bar{\mathbf{x}}$ is the augmented input. It can also be noted that when $\alpha = 0$, the context will simply store the last input vector reconstruction, while with $\alpha = 1$ the context will be a constant null vector, turning the MIGMN into almost an IGMN (with extra useless inputs). Intermediary values will store exponential moving averages of the reconstruction vectors, which can contain longer term information (the higher the α, longer will be the memory). Figure 3.3 shows obtained context values for five different α values on a time-series. The MIGMN's context can also be seen as a compressed form of the RecIGMN's context, to be seen in the next section. Other than that simple modification, the MIGMN works exactly like the static IGMN. Unlike the MSOM, we do not weight contributions of the context and original input separately by some β parameter, trying to let the algorithm figure it out by its own, but it will be included in future works.

3.5 Recursive Incremental Gaussian Mixture Network (RecIGMN)

The RecIGMN augments the IGMN with feedback connections from its cortical region. This architecture can be seen in figure 3.4. 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.

Therefore, all IGMN equations from section 2.1.3 apply, except that the input is augmented with a context vector obtained from each Gaussian component's unnormalized likelihood (equation 2.4) in the last processing step, resulting in a new input vector

$$\bar{\mathbf{x}}(t) = [\bar{p}(\mathbf{x}(t-1)|j)_{t-1}; \mathbf{x}(t)] \tag{3.8}$$

where $[;]$ is the vector concatanation operation and $\bar{p}(\mathbf{x}|j)$ is initialized as the null vector with dimension 0 when the algorithm starts. The unnormalized likelihoods are used as the temporal context instead of the likelihoods or posterior probabilities for two reasons: 1) the range of likelihood values is not [0;1], which is not good for the scaling operations inside the IGMN, like error computations and initial Gaussian component sizes;

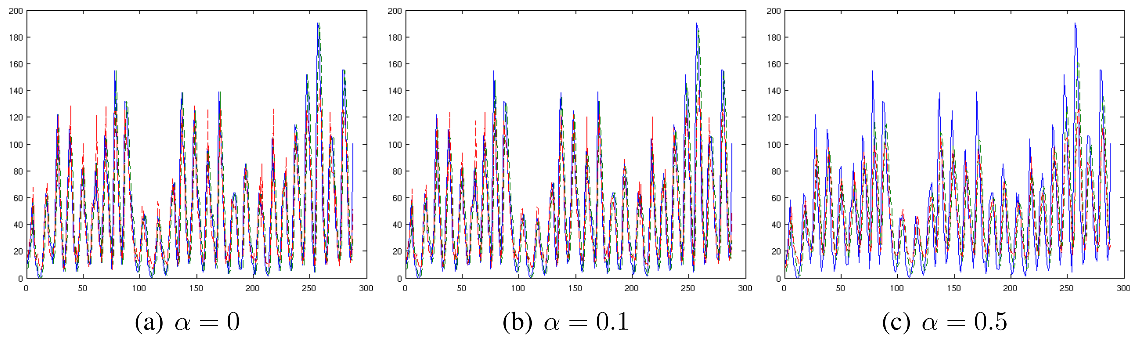
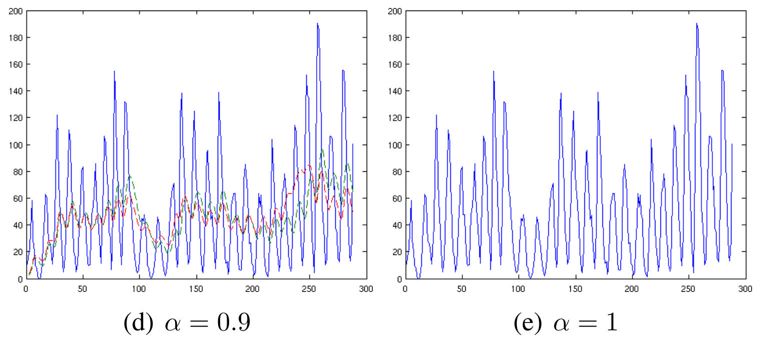



Figure 3.3: MIGMN temporal context along the yearly sunspot numbers time-series. Original series in solid blue, states in dashed green and red (the two state dimensions correspond to reconstructed input and output dimensions of the network).

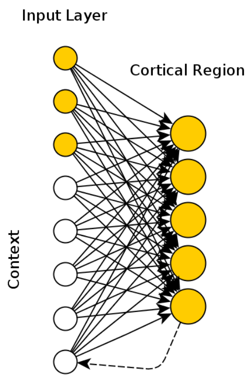


Figure 3.4: An example of RecIGMN with 4 inputs and 3 Gaussian components.

2) the posterior probabilities will be different for a same input in a same context previously seen, since a different number of Gaussian components results in different posterior probabilities.

This kind of context has a huge implementation impact: since the number of Gaussian components in a IGMN is variable, the RecIGMN context will also have variable size, meaning that the algorithm must cope with variable size inputs now. In the next subsections, the modifications to the original IGMN are described.

3.5.1 Creating New Components

Whenever a new component is created (subsection 2.1.3.2), all components means and covariance matrices must be adjusted by adding a new element corresponding to the activation of the newly created component. For the covariance matrices, 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 was not activated at the previous time step. The new covariance matrices 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, equation 2.18 results in δ).

3.5.2 Removing Spurious Components

Whenever a component is removed (subsection 2.1.3.3), all components means and covariance matrices must be adjusted by removing the corresponding element. For the covariance matrices, it means removing 1 full column and 1 full row from the correct positions corresponding to the removed element.

4 EXPERIMENTS AND RESULTS

Experiments both with one-dimensional stochastic (non-deterministic) and chaotic (deterministic with high sensibility to initial conditions) time-series were performed. The main task is to predict the scalar value $\mathbf{x}_i(t+1)$ given $\mathbf{x}_i(t)$ (the target value $\mathbf{x}_t(t)$ is $\mathbf{x}_i(t+1)$), a one-step prediction. A blind run (long-term prediction) experiment is also done (giving only the first data point from the test set and predicting the remaining time-series recursively). Both the naïve and full versions of each IGMN-based algorithm were compared to the ESN (Echo State Network), Elman Network (also known as Simple Recurrent Network or SRN (ELMAN, 1999)), TDNN (Time-Delay Neural Network) (those algorithms were selected due to their temporal online incremental learning capabilities, and that is why more classical algorithms like ARIMA (BOX; JENKINS, 1994) were not used) – referred as classic neural networks from now on – and static IGMN (using only current input to predict the next one; both naïve and full versions). The IGMN-based algorithms were tested with and without additional time-delay lines (the same delays used for TDNN and TDIGMN, and are called from now on TDESIGMN, TDMIGMN and TDRecIGMN). The used error measure for one-step prediction was the normalized MSE with respect to the trivial solution (always predicting the latest observation $\mathbf{x}_i(t)$ for the expected value $\mathbf{x}_i(t+1)$) and is defined as

$$NMSE_t = \frac{\frac{1}{N-1-s}\sum_{t=s+1}^{N-1} \|\mathbf{y}(t) - \mathbf{x}_i(t+1)\|^2}{\frac{1}{N-1-s}\sum_{t=s+1}^{N-1} \|\mathbf{x}_i(t) - \mathbf{x}_i(t+1)\|^2} \tag{4.1}$$

where N is the number of observations, s is the training set size, $\mathbf{x}_i(t)$ is the observation at time t, $\mathbf{y}(t)$ is the predicted value at time t and $\mathbf{x}_i(t+1)$ 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. For the long-term prediction experiments, the normalization is relative to the error that would be obtained by guessing all values using the training set mean, as follows:

$$M = \frac{1}{s}\sum_{t=1}^{s} \mathbf{x}_i(t) \tag{4.2}$$

$$NMSE = \frac{\frac{1}{N-1-s}\sum_{t=s+1}^{N-1} \|\mathbf{y}(t) - \mathbf{x}_i(t+1)\|^2}{\frac{1}{N-1-s}\sum_{t=s+1}^{N-1} \|M - \mathbf{x}_i(t+1)\|^2} \tag{4.3}$$

The runtime (in seconds) and number of epochs are also informed. The Gaussian components (clusters) information refers to the configuration at the end of training. The parameters of the IGMN based algorithms were the same for all experiments: δ was set to 0.1, v_{min} and sp_{min} were set to 2 and 3, respectively. Outlier-based component creation (section 3.1) was used instead of the ϵ_{max} parameter. Due to technical limitations of the Matlab Neural Networks Toolbox and time constraints, batch learning was used for the classic neural networks instead of incremental learning (batch algorithms, even more the ones used here, usually give much better results than the online ones and are slower, since their online counterparts give only approximations for the true gradients and use only first-order derivatives (LECUN et al., 1998), so this should be kept in mind in the comparisons). The ESN output layer was trained with the Conjugate Gradient backpropagation with Fletcher-Reeves updates algorithm (SCALES, 1985) ('traincgf' in Matlab, which does not 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. The TDNN used the Levenberg-Marquardt training algorithm (HAGAN; MENHAJ, 1994). All networks with hidden layers / reservoirs used 10 neurons in this layer, and all input and output layers had size 1 (plus time-delay lines), since one-dimensional data was used. All reservoirs, both for ESN and ESIGMN, were scaled to a spectral radius of 0.9. Each experiment was executed 20 times and results were summarized by medians and median absolute deviations from the median (MAD):

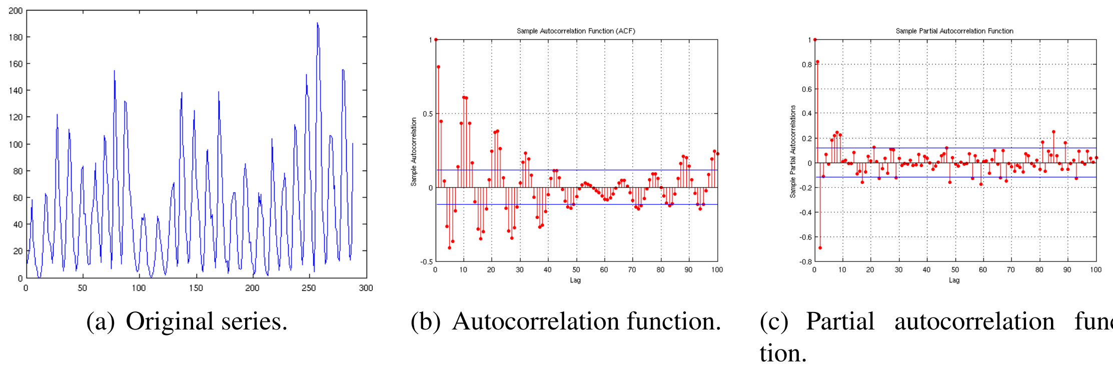
$$MAD = median(|x - median(X)|) \tag{4.4}$$

Those descriptive statistics measures were chosen due to their greater robustness in comparison to means and standard deviations (HUBER; RONCHETTI; MYILIBRARY, 1981). An Intel Core 2 Quad Q8200 with 4GB RAM was used for running all experiments on Matlab 2009b without parallelization. Boxplots (including medians, 1st and 3rd quartiles, 1.5 interquartile range intervals and outliers) were produced for each experiment and statistical significance was determined through the Kruskal-Wallis test (GIBBONS; CHAKRABORTI, 2003) and Wilcoxon Rank-Sum Test (WILCOXON, 1945).

A summary of features of the time-series used in the experiments can be seen in table 4.1. Correlation dimension is a measure of the dimensionality of the space occupied by a data set and can be seen as a type of fractal dimension (GRASSBERGER; PROCACCIA, 1983). Algorithmic complexity is an estimate of the complexity of a symbolic sequence (the time-series are discretized in equidistant bins) given by the Lempel-Ziv algorithm (LEMPEL; ZIV, 1976) (the real algorithmic complexity, or Kolmogorov complexity (KOLMOGOROV, 1965) is not computable and thus an estimate is used). Hjorth mobility and complexity (HJORTH, 1970) are measures of the signal mean frequency and its deviation from the sine shape, respectively. The Hurst Exponent is a measure of long-term memory of time-series (HURST; BLACK; SIMAIKA, 1965), but it is not suitable for non-stationary series, so we use here the Detrended Fluctuation Analysis (DFA) (PENG et al., 1994) descriptor, which is similar to the Hurst Exponent but works with non-stationary time-series. The other measures are simply descriptive statistics measures. All the values were obtained using MATS (Kugiumtzis; Tsimpiris, 2010), a MATLAB toolkit for computation of multiple measures on time-series. Besides those scalar measures, 2 vectorial measures are presented along with each time-series: their autocorrelation and partial autocorrelation functions, which are useful for verifying long-term dependence, seasonality and non-stationarity of the series (BOX; JENKINS, 1994).

	Google	Interest	MG17	MG30	Passengers	Passengers LD	Monthly Sunspots	Yearly Sunspots
Correlation Dimension	0.932	0.997	**0.998**	0.995	**0.826**	0.905	0.956	0.851
Algorithmic Complexity	**0.515**	0.957	0.652	0.875	0.927	**1.133**	0.699	0.860
Mean	**441.975**	1.706	0.919	0.895	281.475	**0.008**	51.925	48.764
Median	**467.215**	1.541	0.966	0.949	267	**0.011**	41.500	39.500
Variance	**17534.01**	0.58	0.05	0.07	14292.40	**0.01**	1966.24	1556.98
Standard Deviation	**132.416**	0.762	0.235	0.282	119.550	**0.106**	44.342	39.458
Interquartile Range	**166.900**	0.892	0.391	0.376	181.250	**0.186**	60.375	53.150
Skewness	-0.566	**1.169**	-0.445	-0.523	0.577	**-0.064**	1.094	1.022
Kurtosis	2.789	**4.011**	2.277	2.458	2.610	**1.941**	3.849	3.615
Hjorth Mobility	**0.021**	0.192	0.048	0.184	0.114	**1.258**	0.254	0.378
Hjorth Complexity	1.409	1.562	1.016	1.005	1.181	0.987	**1.595**	**0.879**
Detrended Fluctuation Analysis	**1.420**	1.274	0.568	0.286	1.372	**0.094**	0.964	0.623

Table 4.1: Descriptors summarizing the time-series used in the experiments. Largest and smallest absolute values in bold.



(a) Original series. (b) Autocorrelation function. (c) Partial autocorrelation function.

Figure 4.1: The yearly mean sunspot numbers time-series

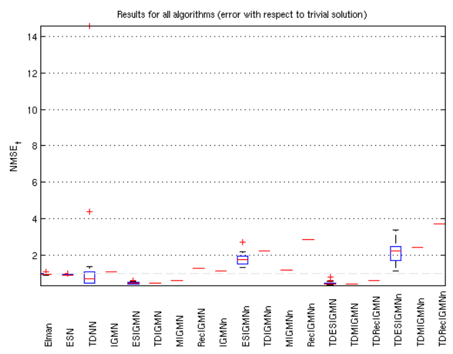
4.1 Yearly Mean Sunspot Numbers

This dataset consists of 289 yearly (monthly mean) observations of a stochastic time-series, which can be seen in figure 4.1, along with its autocorrelation and partial autocorrelation functions. The first 200 observations were used for training, while the remaining 89 were used for testing. For the TD algorithms, 5 extra time-delays were added to input (and 5 data points removed from the training set). For this experiment, $NMSE_t = 1$ corresponds to $NMSE = 0.35$. Results are summarized in table 4.2 with median values and MADs, and also in figure 4.2. Comparison of all algorithms accounting for statistical significance can be seen in tables B.1 for 1-step prediction errors, table B.2 for long-term prediction errors and table B.3 for runtimes (appendix B). Figures A.1 and A.2 show test results for all algorithms on the 1-step and long-term tasks, respectively (appendix A), while figures 4.3 and 4.4 show only best and worst algorithms' results.

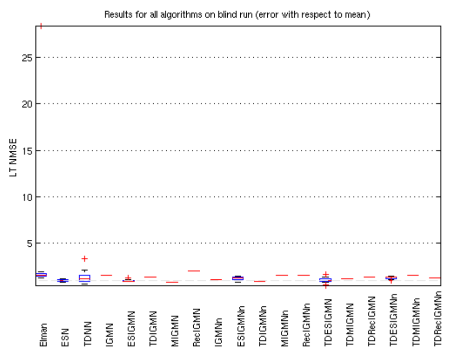
Except for the RecIGMN, all proposed algorithms using full covariance matrices performed statistically better than the trivial solution, the classic neural networks and the static IGMN. Also, all proposed algorithms with full covariance matrices and time-delays were better than the trivial solution, the classic neural networks, the static IGMN and their counterparts without time-delays, with the TDMIGMN being the best performing algorithm of all. The TDMIGMN was better than the TDIGMN and the MIGMN alone, meaning that their combination was beneficial for both. None of the naïve version algorithms could outperform the trivial solution, due to a lack of generalization at the highest peaks. For long-term prediction, the pure MIGMN got the best result. The fastest neural networks for this experiment were the ESN, TDNN and IGMNn, but taking into account only the IGMN-based algorithms better than the trivial solution, ESIGMN was the fastest one. Only the ESIGMN, TDESIGMN and the MIGMN achieved errors less than 1 for both short-term and long-term experiments, with only ESIGMN being significantly better

	$NMSE_t$	LT NMSE	Epochs	Runtime	Clusters
Elman	0.96 (0.0)	1.57 (0.1)	10.7 (2.5)	0.32 (0.1)	0.0 (0.0)
ESN	0.92 (0.0)	1.01 (0.1)	6.2 (0.0)	**0.16** (0.0)	0.0 (0.0)
TDNN	0.70 (0.2)	1.21 (0.3)	13.4 (1.0)	**0.26** (0.0)	0.0 (0.0)
IGMN	1.10 (0.0)	1.60 (0.0)	1.0 (0.0)	0.46 (0.0)	4.0 (0.0)
<u>ESIGMN</u>	0.46 (0.0)	0.93 (0.0)	1.0 (0.0)	0.39 (0.0)	1.0 (0.0)
TDIGMN	0.46 (0.0)	1.41 (0.0)	1.0 (0.0)	0.43 (0.0)	3.0 (0.0)
<u>MIGMN</u>	0.61 (0.0)	<u>**0.81**</u> (0.0)	1.0 (0.0)	0.54 (0.0)	5.0 (0.0)
RecIGMN	1.26 (0.0)	2.00 (0.0)	1.0 (0.0)	0.42 (0.0)	1.0 (0.0)
IGMNn	1.15 (0.0)	1.05 (0.0)	1.0 (0.0)	<u>**0.25**</u> (0.0)	3.0 (0.0)
ESIGMNn	1.74 (0.2)	1.26 (0.1)	1.0 (0.0)	0.30 (0.0)	4.0 (1.0)
TDIGMNn	2.25 (0.0)	0.89 (0.0)	1.0 (0.0)	0.28 (0.0)	3.0 (0.0)
MIGMNn	1.19 (0.0)	1.52 (0.0)	1.0 (0.0)	0.64 (0.1)	6.0 (0.0)
RecIGMNn	2.86 (0.0)	1.59 (0.0)	1.0 (0.0)	0.35 (0.1)	2.0 (0.0)
TDESIGMN	0.43 (0.0)	0.97 (0.1)	1.0 (0.0)	0.43 (0.1)	1.0 (0.0)
TDMIGMN	<u>**0.40**</u> (0.0)	1.21 (0.0)	1.0 (0.0)	0.62 (0.0)	5.0 (0.0)
TDRecIGMN	0.61 (0.0)	1.37 (0.0)	1.0 (0.0)	0.50 (0.0)	3.0 (0.0)
TDESIGMNn	2.22 (0.3)	1.32 (0.1)	1.0 (0.0)	0.33 (0.0)	5.5 (1.5)
TDMIGMNn	2.44 (0.0)	1.54 (0.0)	1.0 (0.0)	0.30 (0.0)	6.0 (0.0)
TDRecIGMNn	3.70 (0.0)	1.30 (0.0)	1.0 (0.0)	0.30 (0.0)	1.0 (0.0)

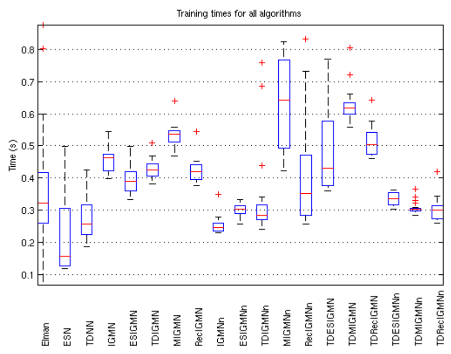
Table 4.2: Results of the mean yearly sunspot numbers experiment. Median values outside parenthesis, MADs inside. Best errors and runtimes in bold, accounting for statistical similarities. Each one-epoch algorithm better than the trivial solution (one-step) and the mean solution (long-term) is underlined, as well as the best errors and runtimes among the IGMN-based algorithms, also accounting for statistical similarities.



(a) One-step prediction $NMSE_t$ boxplot.

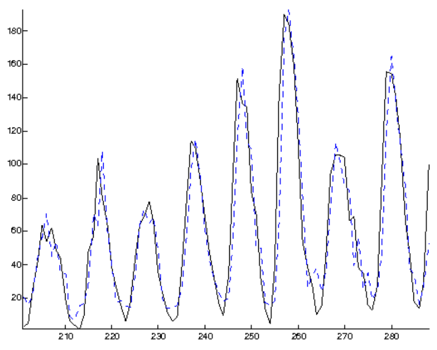


(b) Long-term prediction NMSE boxplot.

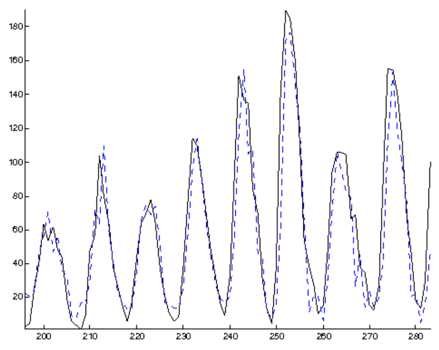


(c) Training time boxplot.

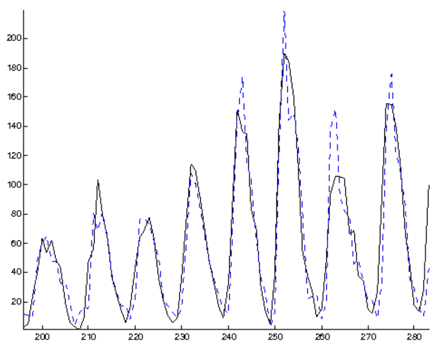
Figure 4.2: Boxplots for the results from all algorithms on the mean yearly sunspot numbers experiment.



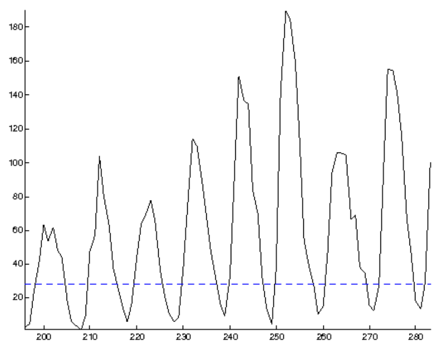
(a) ESIGMN



(b) TDESIGMN

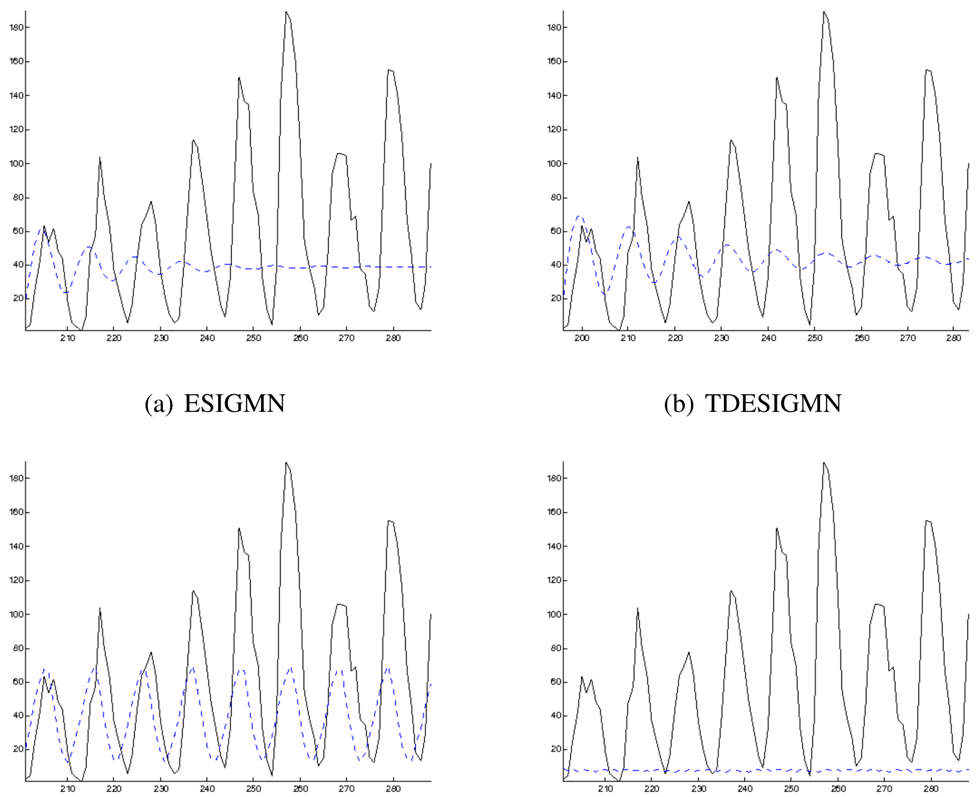


(c) TDMIGMN



(d) TDRecIGMNn

Figure 4.3: One-step prediction example outputs of best and worst algorithms on the yearly mean sunspot numbers time-series (test set only). ESIGMN and TDESIGMN are also shown due to their good performance at this task for both 1-step and long-term predictions. Original series in black, estimates in blue.



(a) ESIGMN (b) TDESIGMN

(c) MIGMN (d) RecIGMN

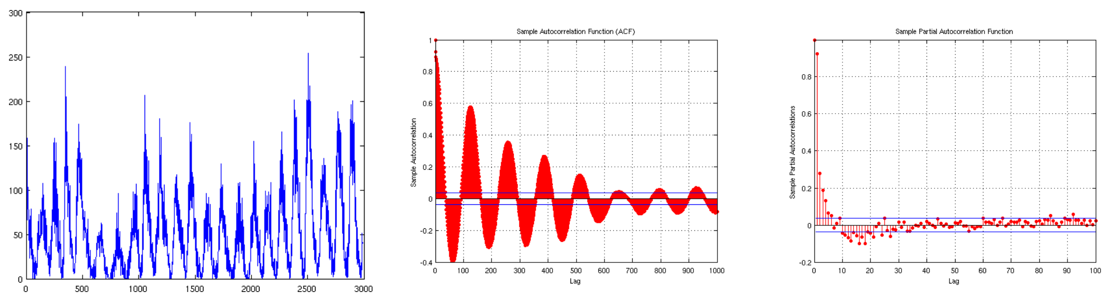
Figure 4.4: Long-term prediction example outputs of best and worst algorithms in the yearly mean sunspot numbers time-series (test set only). ESIGMN and TDESIGMN are also shown due to their good performance at this task for both 1-step and long-term predictions. Original series in black, estimates in blue.

than all classic neural networks and static IGMN too. Overall, the ESIGMN seems to be a good compromise between prediction performance and speed in this experiment.

4.2 Monthly Sunspot Numbers

This dataset consists of 2987 monthly observations of a stochastic time-series, which can be seen in figure 4.5 along with its autocorrelation and partial autocorrelation functions. The first 2000 observations were used for training, while the remaining 987 were used for testing. For the TD algorithms, 16 extra time-delays were added to input (and 16 data points removed from the training set). For this experiment, $NMSE_t = 1$ corresponds to $NMSE = 0.1$. Results are summarized in table 4.3 with medians and MADs, and also in figure 4.6. Comparison of all algorithms accounting for statistical significance can be seen in tables B.4 for 1-step prediction errors, table B.5 for long-term prediction errors and table B.6 for runtimes (appendix B). Figures A.3 and A.4 show test results for all algorithms on the 1-step and long-term tasks, respectively (appendix A), while figures 4.7 and 4.8 show only best and worst algorithms' results.

Very few algorithms managed to outperform the trivial solution in this experiment for one-step prediction: ESN, ESIGMN, RecIGMN and TDRecIGMN, which got the smallest error from all, although not much better than the trivial solution. The MIGMN

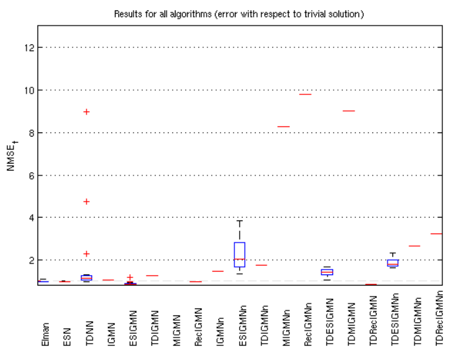


(a) Original series. (b) Autocorrelation function. (c) Partial autocorrelation function.

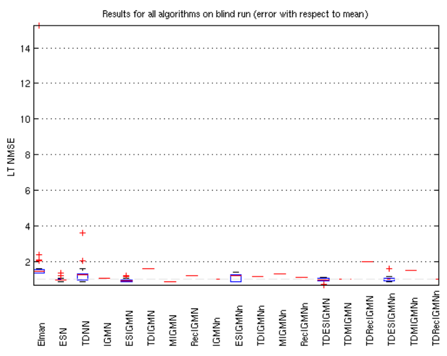
Figure 4.5: The monthly mean sunspot numbers time-series

	$NMSE_t$	LT NMSE	Epochs	Runtime	Clusters
Elman	1.01 (0.0)	1.45 (0.1)	13.1 (4.5)	1.26 (0.6)	0.0 (0.0)
ESN	0.99 (0.0)	0.99 (0.0)	5.0 (0.0)	**0.49** (0.1)	0.0 (0.0)
TDNN	1.14 (0.1)	1.25 (0.2)	12.0 (1.5)	1.07 (0.2)	0.0 (0.0)
IGMN	1.07 (0.0)	1.06 (0.0)	1.0 (0.0)	4.87 (0.1)	4.0 (0.0)
ESIGMN	0.86 (0.0)	0.94 (0.1)	1.0 (0.0)	4.21 (0.3)	1.0 (0.0)
TDIGMN	1.28 (0.0)	1.60 (0.0)	1.0 (0.0)	32.20 (3.1)	51.0 (0.0)
MIGMN	13.06 (0.0)	**0.86** (0.0)	1.0 (0.0)	6.59 (0.1)	9.0 (0.0)
RecIGMN	0.99 (0.0)	1.20 (0.0)	1.0 (0.0)	3.70 (0.1)	1.0 (0.0)
IGMNn	1.48 (0.0)	1.03 (0.0)	1.0 (0.0)	2.67 (0.1)	4.0 (0.0)
ESIGMNn	2.03 (0.5)	1.20 (0.2)	1.0 (0.0)	3.02 (0.1)	3.5 (0.5)
TDIGMNn	1.77 (0.0)	1.17 (0.0)	1.0 (0.0)	4.32 (0.2)	19.0 (0.0)
MIGMNn	8.26 (0.0)	1.33 (0.0)	1.0 (0.0)	2.78 (0.1)	5.0 (0.0)
RecIGMNn	9.81 (0.0)	1.11 (0.0)	1.0 (0.0)	2.60 (0.1)	1.0 (0.0)
TDESIGMN	1.45 (0.1)	0.98 (0.1)	1.0 (0.0)	28.05 (3.6)	36.0 (3.5)
TDMIGMN	9.01 (0.0)	1.03 (0.0)	1.0 (0.0)	8.51 (0.5)	4.0 (0.0)
TDRecIGMN	**0.84** (0.0)	1.98 (0.0)	1.0 (0.0)	9.35 (0.2)	7.0 (0.0)
TDESIGMNn	1.82 (0.1)	1.01 (0.1)	1.0 (0.0)	4.85 (0.4)	23.0 (4.0)
TDMIGMNn	2.65 (0.0)	1.49 (0.0)	1.0 (0.0)	5.03 (0.3)	26.0 (0.0)
TDRecIGMNn	3.21 (0.0)	1.00 (0.0)	1.0 (0.0)	3.93 (0.1)	12.0 (0.0)

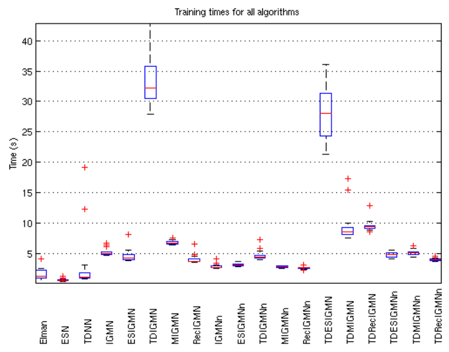
Table 4.3: Results of the monthly sunspot numbers experiment. Median values outside parenthesis, MADs inside. Best errors and runtimes in bold, accounting for statistical similarities. Each one-epoch algorithm better than the trivial solution (one-step) and the mean solution (long-term) is underlined, as well as the best errors and runtimes among the IGMN-based algorithms, also accounting for statistical similarities.



(a) One-step prediction $NMSE_t$ boxplot.

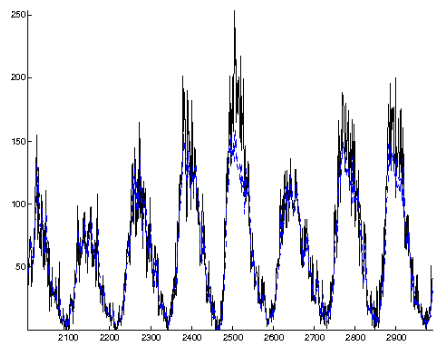


(b) Long-term prediction NMSE boxplot.

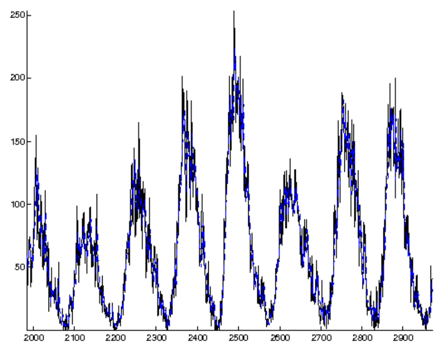


(c) Training time boxplot.

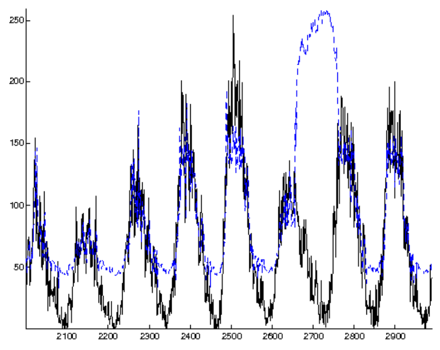
Figure 4.6: Boxplots for the results from all algorithms on the monthly sunspot numbers experiment.



(a) ESIGMN



(b) TDRecIGMN



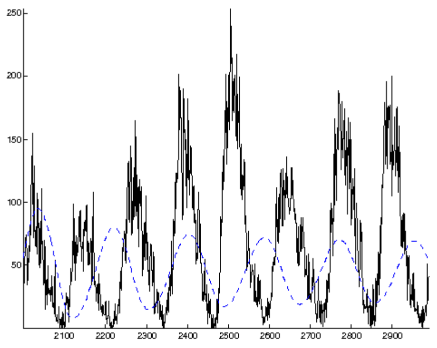
(c) MIGMN

Figure 4.7: One-step prediction example outputs of best and worst algorithms in the monthly mean sunspot numbers time-series (test set only). ESIGMN is also shown due to its good performance at this task for both 1-step and long-term predictions. Original series in black, estimates in blue.

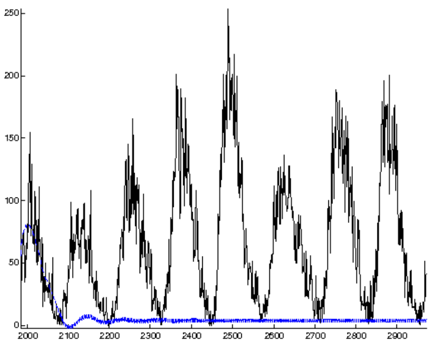
got the best long-term error prediction again, but none of the algorithms could predict reasonably beyond 10 years worth of data, giving constant predictions afterwards, as can be seen in figure A.4. This could be explained by the long-term dependency of this series (as given by its DFA descriptor in table 4.1 and its autocorrelation function in figure 4.5(b)), which is not even handled by the number of time-delays used in the TD versions. Except for the ESIGMN, all TD versions improved on the basic proposed algorithms for one-step predictions, being the TDRecIGMN better than the TDIGMN and RecIGMN, showing the benefit of joining both approaches. ESIGMN was the only algorithm to overcome both the trivial solution (one-step) and mean solution (long-term), as well as all classic algorithms and static IGMN in both tasks.

4.3 Airline Passengers

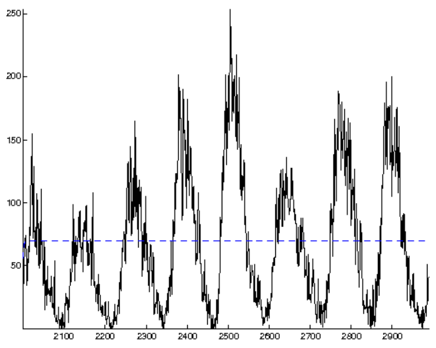
This dataset consists of 143 monthly observations of a stochastic non-stationary time-series, which can be seen in figure 4.9 along with its autocorrelation and partial autocorrelation functions. The first 100 observations were used for training, while the remaining 43 were used for testing. For the TD algorithms, 16 extra time-delays were added to the input (and 16 data points removed from the training set). For this experiment, $NMSE_t$ = 1 corresponds to $NMSE$ = 0.0536. Results are summarized in table 4.4 with median



(a) ESIGMN

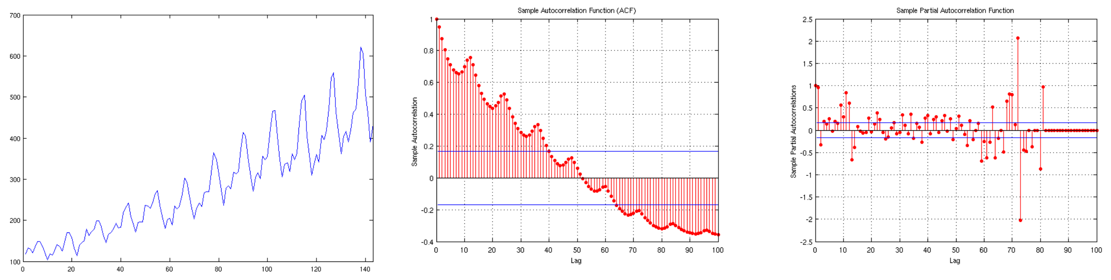


(b) TDRecIGMN



(c) MIGMN

Figure 4.8: Long-term prediction example outputs of best and worst algorithms in the monthly mean sunspot numbers time-series (test set only). ESIGMN is also shown due to its good performance at this task for both 1-step and long-term predictions. Original series in black, estimates in blue.



(a) Original series. (b) Autocorrelation function. (c) Partial autocorrelation function.

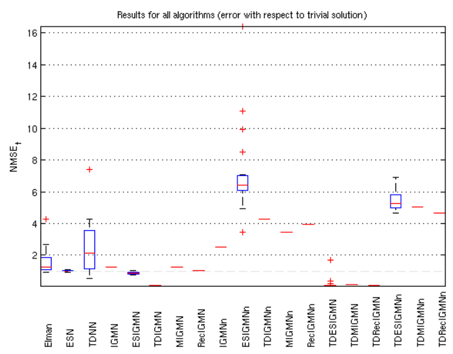
Figure 4.9: The airline passengers time series.

	$NMSE_t$	LT NMSE	Epochs	Runtime	Clusters
Elman	1.25 (0.2)	0.78 (0.3)	9.9 (2.5)	0.26 (0.1)	0.0 (0.0)
ESN	0.99 (0.0)	0.39 (0.1)	5.3 (0.0)	**0.12** (0.0)	0.0 (0.0)
TDNN	2.14 (1.1)	0.53 (0.3)	10.1 (1.0)	0.25 (0.0)	0.0 (0.0)
IGMN	1.26 (0.0)	0.19 (0.0)	1.0 (0.0)	0.17 (0.0)	2.0 (0.0)
ESIGMN	0.90 (0.1)	0.21 (0.1)	1.0 (0.0)	0.21 (0.0)	1.0 (0.0)
TDIGMN	**0.08** (0.0)	**0.02** (0.0)	1.0 (0.0)	**0.14** (0.0)	1.0 (0.0)
MIGMN	1.28 (0.0)	0.25 (0.0)	1.0 (0.0)	0.17 (0.0)	1.0 (0.0)
RecIGMN	1.02 (0.0)	0.56 (0.0)	1.0 (0.0)	0.21 (0.0)	1.0 (0.0)
IGMNn	2.51 (0.0)	0.26 (0.0)	1.0 (0.0)	**0.13** (0.0)	6.0 (0.0)
ESIGMNn	6.43 (0.6)	0.37 (0.1)	1.0 (0.0)	0.17 (0.0)	2.0 (0.0)
TDIGMNn	4.28 (0.0)	0.25 (0.0)	1.0 (0.0)	**0.12** (0.0)	9.0 (0.0)
MIGMNn	3.45 (0.0)	0.43 (0.0)	1.0 (0.0)	0.33 (0.1)	8.0 (0.0)
RecIGMNn	3.96 (0.0)	0.21 (0.0)	1.0 (0.0)	0.19 (0.1)	2.0 (0.0)
TDESIGMN	0.10 (0.0)	0.04 (0.0)	1.0 (0.0)	0.18 (0.0)	1.0 (0.0)
TDMIGMN	0.15 (0.0)	0.09 (0.0)	1.0 (0.0)	0.17 (0.0)	1.0 (0.0)
TDRecIGMN	0.09 (0.0)	0.03 (0.0)	1.0 (0.0)	0.16 (0.0)	1.0 (0.0)
TDESIGMNn	5.27 (0.3)	0.37 (0.0)	1.0 (0.0)	**0.13** (0.0)	7.0 (1.0)
TDMIGMNn	5.03 (0.0)	0.36 (0.0)	1.0 (0.0)	**0.12** (0.0)	9.0 (0.0)
TDRecIGMNn	4.65 (0.0)	0.37 (0.0)	1.0 (0.0)	**0.13** (0.0)	8.0 (0.0)

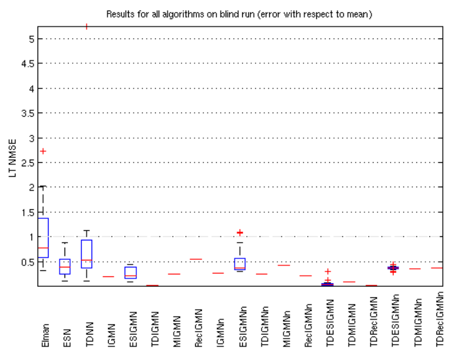
Table 4.4: Results of the airline passengers experiment. Median values outside parenthesis, MADs inside. Best errors and runtimes in bold, accounting for statistical similarities. Each one-epoch algorithm better than the trivial solution (one-step) and the mean solution (long-term) is underlined, as well as the best errors and runtimes among the IGMN-based algorithms, also accounting for statistical similarities.

values and MADs, and also in figure 4.10. Comparison of all algorithms accounting for statistical significance can be seen in tables B.7 for 1-step prediction errors, table B.8 for long-term prediction errors and table B.9 for runtimes (appendix B). Figures A.5 and A.6 show test results for all algorithms on the 1-step and long-term tasks, respectively (appendix A), while figures 4.11 and 4.12 show only best and worst algorithms' results.

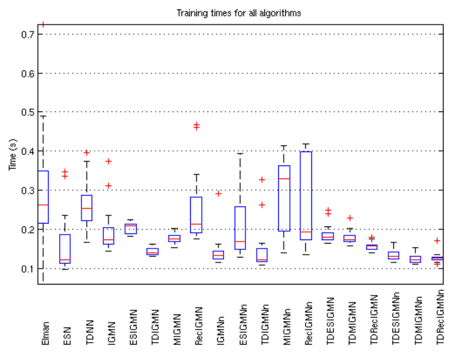
Six algorithms outperformed the trivial (one-step) and mean (long-term) solutions in this experiment: ESN, ESIGMN, TDESIGMN, TDMIGMN, TDRecIGMN and TDIGMN, the later being the best in both tasks. Since this is a relatively small experiment, many algorithms got statistically similar results for the execution time. For this experiment, it is clear that time-delays played a major role in the prediction performance of IGMN-based algorithms, maybe because the delays are long enough to capture the trend, growing variance and seasonality of this time-series (but the ESN's and ESIGMN's reservoirs are good replacements for the time-delays). The naïve algorithms could not generalize well and simply did not learn the series trend, as can be seen in figures A.5 and A.6.



(a) $NMSE_t$ boxplot.

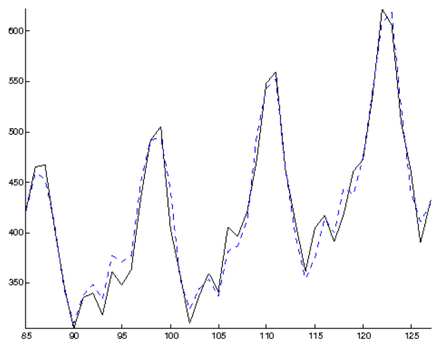


(b) Long-term NMSE boxplot.

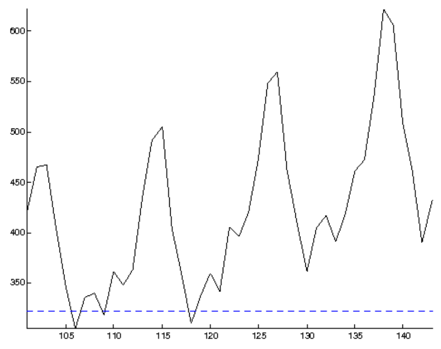


(c) Training time boxplot.

Figure 4.10: Boxplots for the results from all algorithms on the airline passengers experiment.

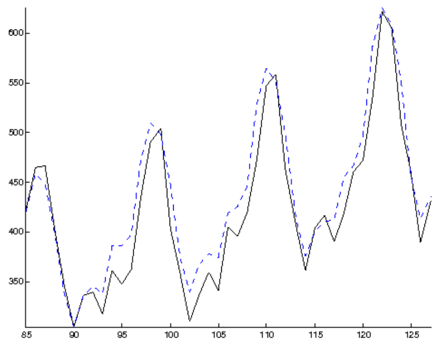
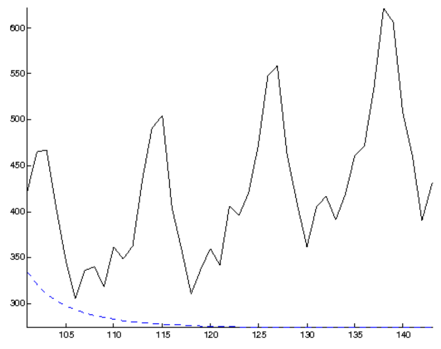


(a) TDIGMN



(b) ESIGMNn

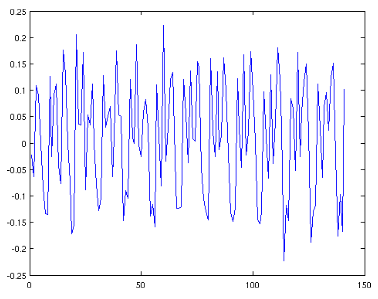
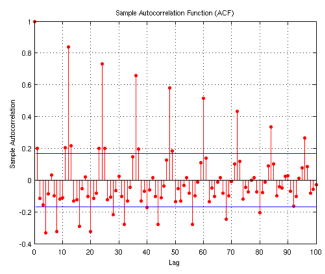
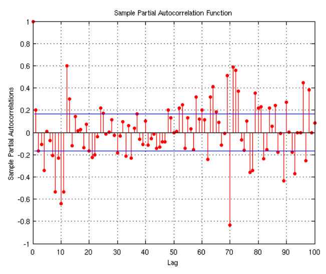
Figure 4.11: One-step prediction example outputs of best and worst algorithms in the airline passengers time-series (test set only). Original series in black, estimates in blue.

(a) TDIGMN

(b) Elman

Figure 4.12: Long-term prediction example outputs of best and worst algorithms in the airline passengers time-series. Original series in black, estimates in blue.

(a) Original series.

(b) Autocorrelation function.

(c) Partial autocorrelation function.

Figure 4.13: The log-differentiated airline passengers time series.

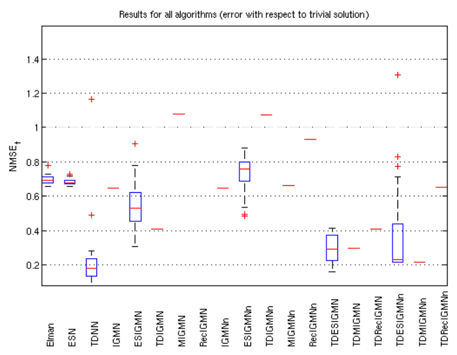
4.4 Log-Differentiated Airline Passengers

In order to assess the importance of the non-stationarity of the airline passengers time-series to the performance of the evaluated algorithms, a new experiment was done using the same time-series but now log-differentiated. The resulting time-series can be seen in figure 4.13 along with its new autocorrelation and partial autocorrelation functions. For this experiment, $NMSE_t = 1$ corresponds to $NMSE = 1.3851$. Results are summarized in table 4.5 with median values and MADs, and also in figure 4.14. Comparison of all algorithms accounting for statistical significance can be seen in tables B.10 for 1-step prediction errors, table B.11 for long-term prediction errors and table B.12 for runtimes (appendix B). Figures A.7 and A.8 show test results for all algorithms on the 1-step and long-term tasks, respectively (appendix A), while figures 4.15 and 4.16 show only best and worst algorithms' results.

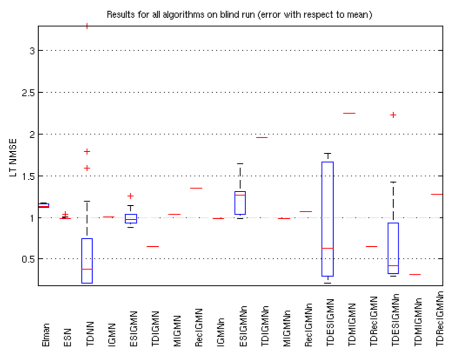
It is clear from those results that removing the trend and the growing variance of the airline passengers time-series allowed many other algorithms to succeed. Actually, the best predictions were obtained by a simple TDNN. Also, many naïve version algorithms managed to solve the problem. Based on those two experiments, we conclude that ESN, ESIGMN, TDESIGMN, TDMIGMN, TDRecIGMN and TDIGMN were more robust to the trend and the growing variance in the airline passengers time-series, showing good results in both experiments.

	$NMSE_t$	LT NMSE	Epochs	Runtime	Clusters
Elman	0.69 (0.0)	1.13 (0.0)	17.1 (4.0)	0.58 (0.2)	0.0 (0.0)
ESN	0.68 (0.0)	0.99 (0.0)	5.1 (0.0)	**0.11** (0.0)	0.0 (0.0)
TDNN	**0.18** (0.1)	**0.38** (0.2)	7.5 (0.5)	**0.12** (0.0)	0.0 (0.0)
IGMN	0.65 (0.0)	1.01 (0.0)	1.0 (0.0)	0.20 (0.0)	2.0 (0.0)
ESIGMN	0.53 (0.1)	0.97 (0.1)	1.0 (0.0)	0.25 (0.0)	2.0 (0.0)
TDIGMN	0.41 (0.0)	0.65 (0.0)	1.0 (0.0)	0.30 (0.0)	5.0 (0.0)
MIGMN	1.08 (0.0)	1.04 (0.0)	1.0 (0.0)	0.26 (0.0)	4.0 (0.0)
RecIGMN	1.59 (0.0)	1.35 (0.0)	1.0 (0.0)	0.27 (0.0)	3.0 (0.0)
IGMNn	0.65 (0.0)	0.99 (0.0)	1.0 (0.0)	0.13 (0.0)	2.0 (0.0)
ESIGMNn	0.76 (0.1)	1.27 (0.1)	1.0 (0.0)	0.15 (0.0)	2.0 (0.0)
TDIGMNn	1.07 (0.0)	1.96 (0.0)	1.0 (0.0)	0.15 (0.0)	3.0 (0.0)
MIGMNn	0.66 (0.0)	0.98 (0.0)	1.0 (0.0)	0.14 (0.0)	2.0 (0.0)
RecIGMNn	0.93 (0.0)	1.07 (0.0)	1.0 (0.0)	0.15 (0.0)	2.0 (0.0)
TDESIGMN	0.29 (0.1)	0.63 (0.4)	1.0 (0.0)	0.37 (0.0)	5.0 (0.0)
TDMIGMN	0.30 (0.0)	2.25 (0.0)	1.0 (0.0)	0.45 (0.0)	6.0 (0.0)
TDRecIGMN	0.41 (0.0)	0.65 (0.0)	1.0 (0.0)	0.36 (0.0)	5.0 (0.0)
TDESIGMNn	0.23 (0.0)	0.42 (0.1)	1.0 (0.0)	0.16 (0.0)	3.0 (0.0)
TDMIGMNn	**0.22** (0.0)	**0.32** (0.0)	1.0 (0.0)	0.16 (0.0)	3.0 (0.0)
TDRecIGMNn	0.65 (0.0)	1.27 (0.0)	1.0 (0.0)	0.13 (0.0)	2.0 (0.0)

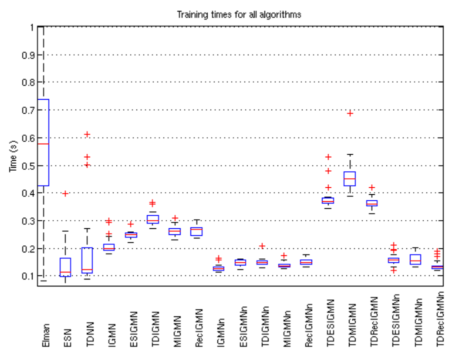
Table 4.5: Results of the log-differentiated airline passengers experiment. Median values outside parenthesis, MADs inside. Best errors and runtimes in bold, accounting for statistical similarities. Each one-epoch algorithm better than the trivial solution (one-step) and the mean solution (long-term) is underlined, as well as the best errors and runtimes among the IGMN-based algorithms, also accounting for statistical similarities.



(a) One-step prediction $NMSE_t$ boxplot.

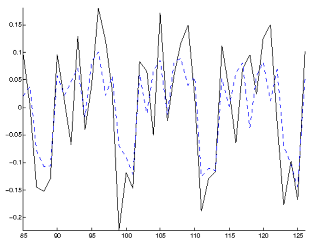


(b) Long-term prediction NMSE boxplot.

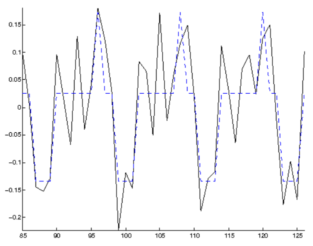


(c) Training time boxplot.

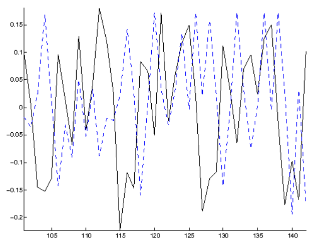
Figure 4.14: Boxplots for the results from all algorithms on the log-differentiated airline passengers experiment.



(a) TDNN

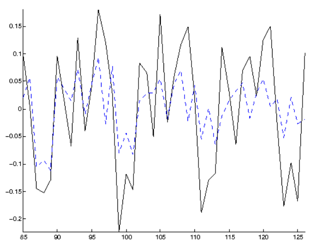


(b) TDMIGMNn

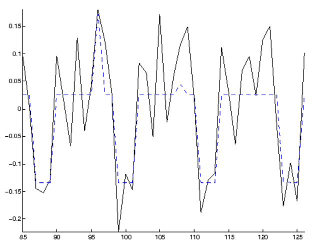


(c) RecIGMN

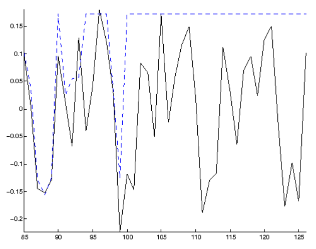
Figure 4.15: One-step prediction example outputs of best and worst algorithms in the log-differentiated airline passengers time-series (test set only). Original series in black, estimates in blue.



(a) TDNN

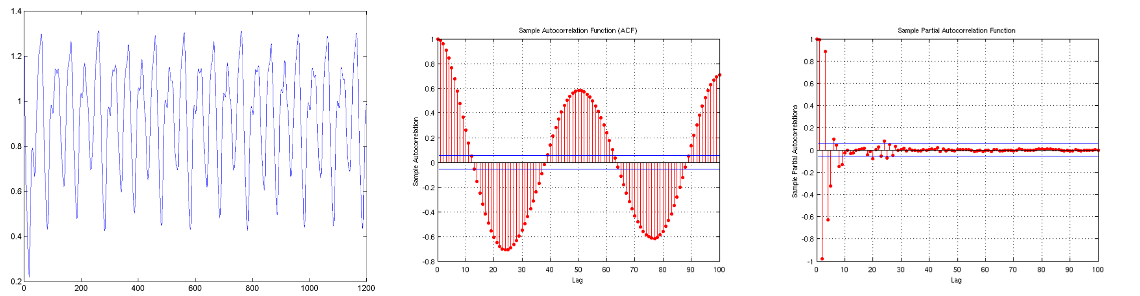


(b) TDMIGMNn



(c) TDMIGMN

Figure 4.16: Long-term prediction example outputs of best and worst algorithms in the log-differentiated airline passengers time-series. Original series in black, estimates in blue.



(a) Original series. (b) Autocorrelation function. (c) Partial autocorrelation function.

Figure 4.17: The Mackey-Glass (τ=17) time-series.

4.5 Mackey-Glass (τ=17)

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

$$\frac{dx}{dt} = 0.2\frac{x_{t-\tau}}{1+x^n_{t-\tau}} - 0.1x_t \tag{4.5}$$

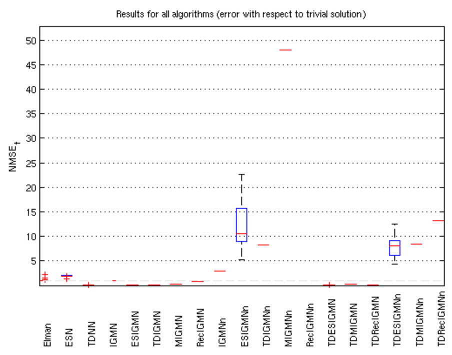
with $\tau = 17$, which can be seen in figure 4.17 along with its autocorrelation and partial autocorrelation functions. The first 1000 observations were used for training, while the remaining 201 were used for testing. For the TD algorithms, 3 extra time-delays were added to the input (at lags 6, 12 and 18), and 17 data points were removed from their training sets. For this experiment, $NMSE_t$ = 1 corresponds to $NMSE$ = 0.0196. Results are summarized in table 4.6 with median values and MADs, and also in figure 4.18. Comparison of all algorithms accounting for statistical significance can be seen in tables B.13 for 1-step prediction errors, table B.14 for long-term prediction errors and table B.15 for runtimes (appendix B). Figures A.9 and A.10 show test results for all algorithms on the 1-step and long-term tasks, respectively (appendix A), while figures 4.19 and 4.20 show only best and worst algorithms' results.

The TDNN got the best one-step prediction error in this experiment, closely followed by the TDIGMN. Only the TDMIGMN could overcome the mean solution for the long-term prediction task, but only by a small amount. It was also the only algorithm with errors smaller than 1 for both tasks. The ESN was by far the fastest algorithm in this experiment. Except for the MIGMN, the time-delays improved all algorithms. This, together with the great one-step prediction performance of the TDNN, could be explained by the carefully picked lags already consolidated in chaos-theoretic studies (SALMERÓN et al., 2002) (WEIGEND; GERSHENFELD, 1994).

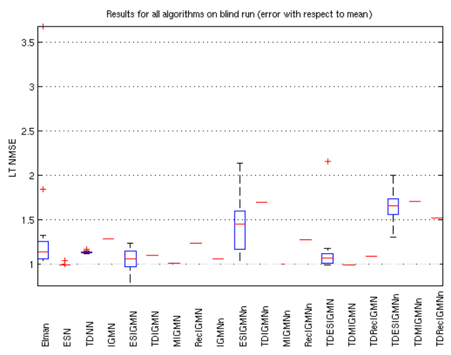
It is also interesting to note that the ESIGMN could achieve better results (by a large margin) than the ESN in the one-step prediction task with only a single Gaussian component in various runs, meaning that it has found a linear solution over input and reservoir state space. An ESIGMN with just one component is almost equivalent to an ESN trained in batch mode, but the ESIGMN linear regression done inside the component is more akin to a Total Least Squares (TLS), since it is done in both input and output variables, while the ESN linear regression done in the output layer is closer to a Ordinary Least Squares (OLS).

	$NMSE_t$	LT NMSE	Epochs	Runtime	Clusters
Elman	1.02 (0.0)	1.14 (0.1)	42.2 (15.0)	3.33 (0.9)	0.0 (0.0)
ESN	1.92 (0.0)	1.00 (0.0)	7.2 (0.0)	**0.24** (0.0)	0.0 (0.0)
TDNN	**0.01** (0.0)	1.14 (0.0)	13.2 (3.5)	0.54 (0.2)	0.0 (0.0)
IGMN	1.00 (0.0)	1.29 (0.0)	1.0 (0.0)	2.13 (0.1)	3.0 (0.0)
ESIGMN	0.10 (0.0)	1.06 (0.1)	1.0 (0.0)	1.88 (0.2)	1.0 (0.0)
TDIGMN	0.03 (0.0)	1.10 (0.0)	1.0 (0.0)	2.19 (0.1)	3.0 (0.0)
MIGMN	0.25 (0.0)	1.02 (0.0)	1.0 (0.0)	1.99 (0.0)	2.0 (0.0)
RecIGMN	0.84 (0.0)	1.24 (0.0)	1.0 (0.0)	2.38 (0.1)	3.0 (0.0)
IGMNn	2.87 (0.0)	1.06 (0.0)	1.0 (0.0)	1.28 (0.1)	4.0 (0.0)
ESIGMNn	10.55 (3.5)	1.45 (0.2)	1.0 (0.0)	1.56 (0.1)	4.0 (1.0)
TDIGMNn	8.16 (0.0)	1.70 (0.0)	1.0 (0.0)	1.38 (0.1)	5.0 (0.0)
MIGMNn	47.94 (0.0)	1.00 (0.0)	1.0 (0.0)	1.54 (0.1)	7.0 (0.0)
RecIGMNn	52.86 (0.0)	1.28 (0.0)	1.0 (0.0)	1.36 (0.0)	2.0 (0.0)
TDESIGMN	0.05 (0.0)	1.07 (0.1)	1.0 (0.0)	2.08 (0.3)	1.0 (0.0)
TDMIGMN	0.29 (0.0)	**0.99** (0.0)	1.0 (0.0)	2.41 (0.1)	3.0 (0.0)
TDRecIGMN	0.05 (0.0)	1.09 (0.0)	1.0 (0.0)	2.70 (0.1)	4.0 (0.0)
TDESIGMNn	8.13 (1.4)	1.66 (0.1)	1.0 (0.0)	1.80 (0.1)	8.0 (1.0)
TDMIGMNn	8.43 (0.0)	1.71 (0.0)	1.0 (0.0)	1.62 (0.1)	13.0 (0.0)
TDRecIGMNn	13.13 (0.0)	1.52 (0.0)	1.0 (0.0)	1.42 (0.1)	5.0 (0.0)

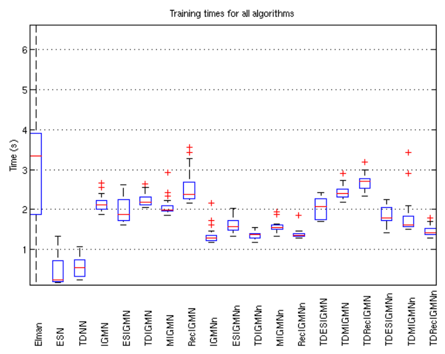
Table 4.6: Results of the Mackey-Glass (τ = 17) experiment. Median values outside parenthesis, MADs inside. Best errors and runtimes in bold, accounting for statistical similarities. Each one-epoch algorithm better than the trivial solution (one-step) and the mean solution (long-term) is underlined, as well as the best errors and runtimes among the IGMN-based algorithms, also accounting for statistical similarities.



(a) $NMSE_t$ boxplot.

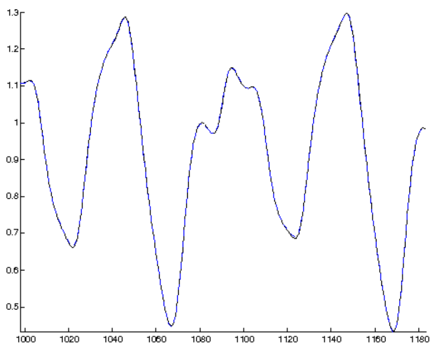


(b) Long-term NMSE boxplot.

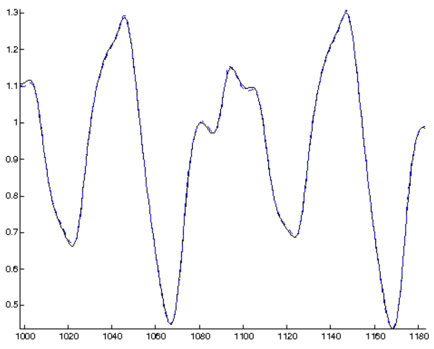


(c) Training time boxplot.

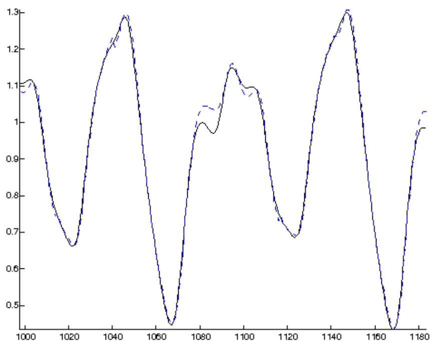
Figure 4.18: Boxplot of the results from all algorithms on the Mackey-Glass (τ = 17) experiment.



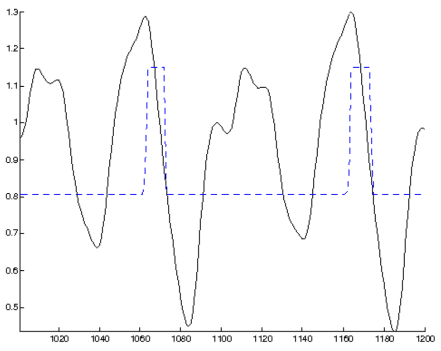
(a) TDNN



(b) TDIGMN

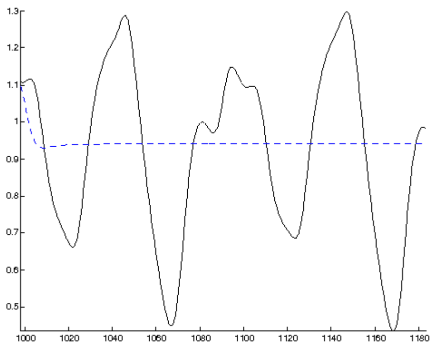


(c) TDMIGMN

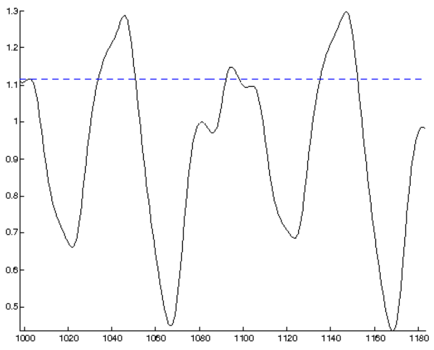


(d) RecIGMNn

Figure 4.19: One-step prediction example outputs of best and worst algorithms in the Mackey-Glass (τ = 17)time-series (test set only). The TDMIGMN is also shown due to its good performance at both 1-step and long-term prediction tasks. Original series in black, estimates in blue.

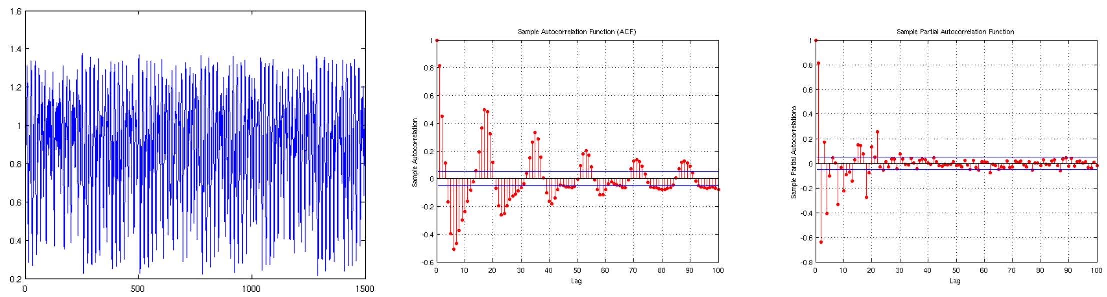


(a) TDMIGMN



(b) TDMIGMNn

Figure 4.20: Long-term prediction example outputs of best and worst algorithms in the Mackey-Glass (τ = 17) time-series. Original series in black, estimates in blue.



(a) Original series.

(b) Autocorrelation function.

(c) Partial autocorrelation function.

Figure 4.21: The Mackey-Glass (τ=30) time-series.

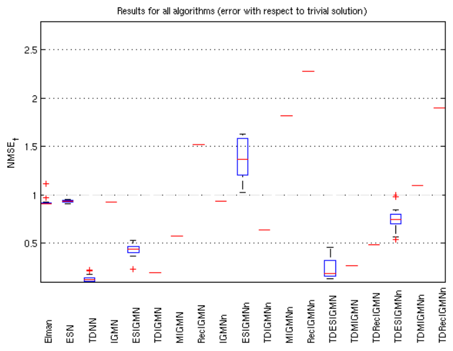
	$NMSE_t$	LT NMSE	Epochs	Runtime	Clusters
Elman	0.91 (0.0)	1.03 (0.0)	25.1 (8.5)	1.81 (0.6)	0.0 (0.0)
ESN	0.93 (0.0)	**1.00** (0.0)	8.4 (3.5)	**0.51** (0.3)	0.0 (0.0)
TDNN	**0.13** (0.0)	2.38 (0.4)	10.2 (2.5)	**0.29** (0.1)	0.0 (0.0)
IGMN	0.93 (0.0)	1.04 (0.0)	1.0 (0.0)	2.03 (0.1)	3.0 (0.0)
ESIGMN	0.44 (0.0)	**1.00** (0.0)	1.0 (0.0)	2.66 (0.1)	3.0 (0.0)
TDIGMN	0.20 (0.0)	1.22 (0.0)	1.0 (0.0)	3.34 (0.1)	9.0 (0.0)
MIGMN	0.58 (0.0)	1.04 (0.0)	1.0 (0.0)	2.26 (0.1)	3.0 (0.0)
RecIGMN	1.52 (0.0)	1.41 (0.0)	1.0 (0.0)	2.48 (0.1)	3.0 (0.0)
IGMNn	0.94 (0.0)	1.07 (0.0)	1.0 (0.0)	1.28 (0.1)	4.0 (0.0)
ESIGMNn	1.37 (0.2)	1.29 (0.2)	1.0 (0.0)	1.57 (0.1)	4.0 (1.0)
TDIGMNn	0.64 (0.0)	1.14 (0.0)	1.0 (0.0)	1.41 (0.1)	6.0 (0.0)
MIGMNn	1.82 (0.0)	1.22 (0.0)	1.0 (0.0)	1.39 (0.0)	4.0 (0.0)
RecIGMNn	2.28 (0.0)	1.05 (0.0)	1.0 (0.0)	1.48 (0.1)	3.0 (0.0)
TDESIGMN	0.19 (0.0)	1.78 (0.8)	1.0 (0.0)	3.37 (0.4)	5.0 (1.0)
TDMIGMN	0.26 (0.0)	1.72 (0.0)	1.0 (0.0)	5.88 (0.2)	17.0 (0.0)
TDRecIGMN	0.49 (0.0)	1.27 (0.0)	1.0 (0.0)	4.04 (0.3)	7.0 (0.0)
TDESIGMNn	0.74 (0.1)	1.16 (0.0)	1.0 (0.0)	1.86 (0.1)	8.0 (1.0)
TDMIGMNn	1.10 (0.0)	1.14 (0.0)	1.0 (0.0)	1.61 (0.0)	7.0 (0.0)
TDRecIGMNn	1.90 (0.0)	1.05 (0.0)	1.0 (0.0)	1.65 (0.0)	3.0 (0.0)

Table 4.7: Results of the Mackey-Glass (τ = 30) experiment. Median values outside parenthesis, MADs inside. Best errors and runtimes in bold, accounting for statistical similarities. Each one-epoch algorithm better than the trivial solution (one-step) and the mean solution (long-term) is underlined, as well as the best errors and runtimes among the IGMN-based algorithms, also accounting for statistical similarities.

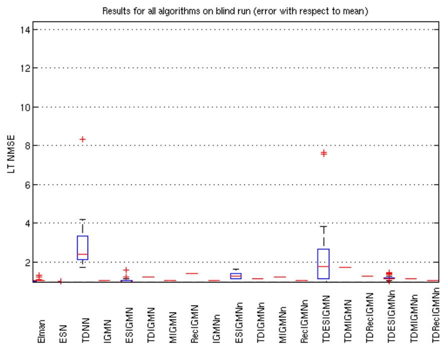
4.6 Mackey-Glass (τ=30)

This dataset with 1500 observations is generated by the same equation 4.5 but with $\tau = 30$, and can be seen in figure 4.21. The first 1000 observations were used for training, while the remaining 500 were used for testing. For the TD algorithms, 3 extra time-delays were added to the input (at lags 6, 12 and 18), and 17 data point were removed from the training set. For this experiment, $NMSE_t$ = 1 corresponds to $NMSE$ = 0.359. Results are summarized in table 4.7 with median values and MADs, and also in figure 4.22. Comparison of all algorithms accounting for statistical significance can be seen in tables B.16 for 1-step prediction errors, table B.17 for long-term prediction errors and table B.18 for runtimes (appendix B). Figures A.11 and A.12 show test results for all algorithms on the 1-step and long-term tasks, respectively (appendix A), while figures 4.23 and 4.24 show only best and worst algorithms' results.

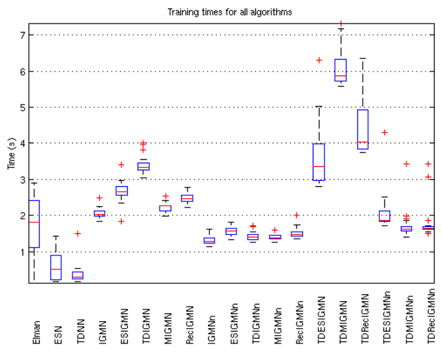
Again, the TDNN got the best one-step prediction error, closely followed by the TDIGMN and TDESIGMN. There was no algorithm capable of overcoming the mean solution in the long-term prediction task in this experiment, but the ESN and ESIGMN



(a) $NMSE_t$ boxplot.

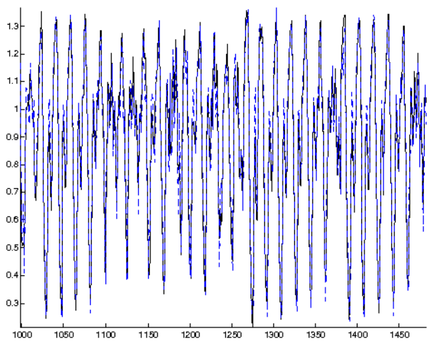


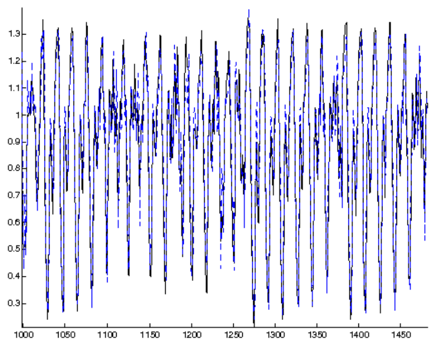
(b) Long-term NMSE boxplot.

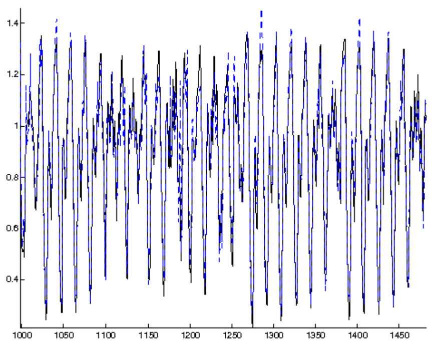


(c) Training time boxplot.

Figure 4.22: Boxplot of the results from all algorithms on the Mackey-Glass (τ = 30) experiment.


(a) TDNN


(b) TDIGMN


(c) TDESIGMN

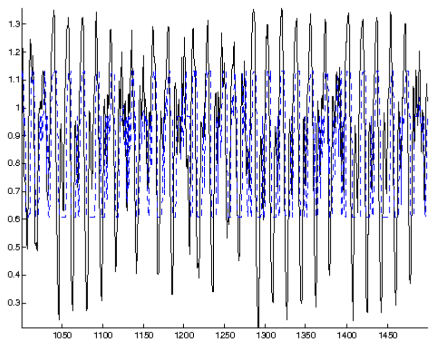
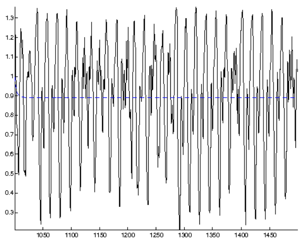
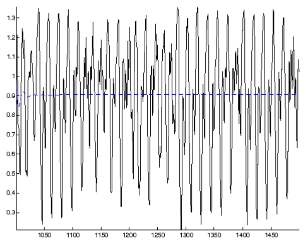

(d) RecIGMNn

Figure 4.23: One-step prediction example outputs of best and worst algorithms in the Mackey-Glass (τ = 30) time-series (test set only). Original series in black, estimates in blue.


(a) ESN


(b) ESIGMN

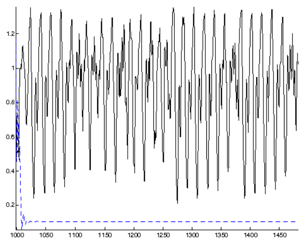
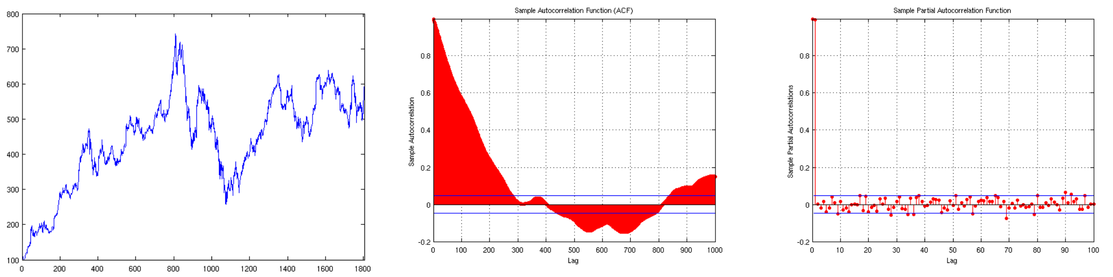

(c) TDNN

Figure 4.24: Long-term prediction example outputs of best and worst algorithms in the Mackey-Glass (τ = 30) time-series. Original series in black, estimates in blue.



(a) Original series. (b) Autocorrelation function. (c) Partial autocorrelation function.

Figure 4.25: The Google stock time-series.

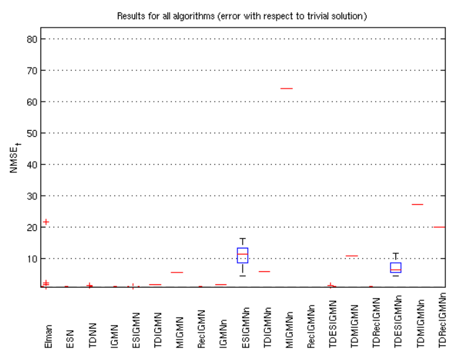
	$NMSE_t$	LT NMSE	Epochs	Runtime	Clusters
Elman	**1.00** (0.0)	0.15 (0.0)	24.0 (12.5)	1.78 (1.1)	0.0 (0.0)
ESN	**1.00** (0.0)	0.31 (0.1)	6.2 (0.0)	**0.49** (0.1)	0.0 (0.0)
TDNN	**1.00** (0.0)	0.24 (0.1)	16.6 (1.5)	0.73 (0.2)	0.0 (0.0)
IGMN	**1.00** (0.0)	0.25 (0.0)	1.0 (0.0)	3.78 (0.8)	2.0 (0.0)
ESIGMN	**1.00** (0.0)	0.25 (0.0)	1.0 (0.0)	3.03 (0.2)	1.0 (0.0)
TDIGMN	1.72 (0.0)	0.17 (0.0)	1.0 (0.0)	3.03 (0.0)	5.0 (0.0)
MIGMN	5.43 (0.0)	0.14 (0.0)	1.0 (0.0)	2.80 (0.1)	3.0 (0.0)
RecIGMN	**1.00** (0.0)	0.14 (0.0)	1.0 (0.0)	2.96 (0.0)	3.0 (0.0)
IGMNn	1.57 (0.0)	0.40 (0.0)	1.0 (0.0)	2.18 (0.1)	9.0 (0.0)
ESIGMNn	11.46 (2.5)	0.40 (0.1)	1.0 (0.0)	2.43 (0.1)	5.0 (1.5)
TDIGMNn	5.86 (0.0)	0.48 (0.0)	1.0 (0.0)	2.31 (0.1)	12.0 (0.0)
MIGMNn	64.20 (0.0)	0.23 (0.0)	1.0 (0.0)	2.39 (0.1)	12.0 (0.0)
RecIGMNn	83.70 (0.0)	0.41 (0.0)	1.0 (0.0)	2.40 (0.1)	5.0 (0.0)
TDESIGMN	1.05 (0.0)	0.23 (0.1)	1.0 (0.0)	3.49 (0.2)	3.0 (0.0)
TDMIGMN	10.81 (0.0)	0.26 (0.0)	1.0 (0.0)	3.25 (0.2)	3.0 (0.0)
TDRecIGMN	1.02 (0.0)	0.19 (0.0)	1.0 (0.0)	3.42 (0.1)	3.0 (0.0)
TDESIGMNn	6.49 (1.6)	0.44 (0.0)	1.0 (0.0)	2.81 (0.2)	10.0 (1.0)
TDMIGMNn	27.26 (0.0)	0.45 (0.0)	1.0 (0.0)	2.60 (0.2)	14.0 (0.0)
TDRecIGMNn	20.10 (0.0)	**0.12** (0.0)	1.0 (0.0)	2.77 (0.2)	9.0 (0.0)

Table 4.8: Results of the Google stock experiment. Median values outside parenthesis, MADs inside. Best errors and runtimes in bold, accounting for statistical similarities. Each one-epoch algorithm better than the trivial solution (one-step) and the mean solution (long-term) is underlined, as well as the best errors and runtimes among the IGMN-based algorithms, also accounting for statistical similarities.

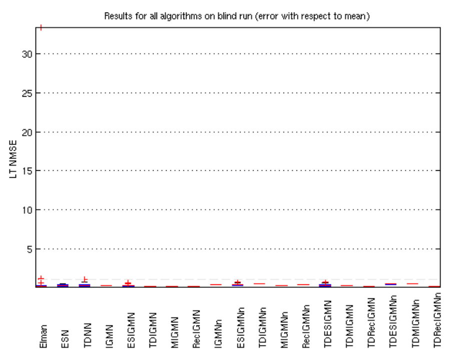
were able to, at least, match it. Many naïve version algorithms did well in this experiment. The ESN and TDNN algorithms were the fastest ones in this experiment.

4.7 Google Stock Prices

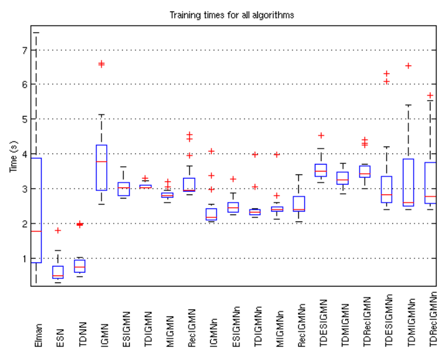
This financial dataset consists of 1803 observations from the Google stock prices (closing values), and can be seen in figure 4.25. The first 1500 observations were used for training, while the remaining 303 were used for testing. For the TD algorithms, 4 extra time-delays were added to the input (and 4 data points were removed from the training set). For this experiment, $NMSE_t = 1$ corresponds to $NMSE = 0.005$. Results are summarized in table 4.8 with median values and MADs, and also in figure 4.26. Comparison of all algorithms accounting for statistical significance can be seen in tables B.19 for 1-step prediction errors, table B.20 for long-term prediction errors and table B.21 for runtimes (appendix B). Figures A.13 and A.14 show test results for all algorithms on the 1-step and long-term tasks, respectively (appendix A), while figures 4.27 and 4.28 show only best and worst algorithms' results.



(a) $NMSE_t$ boxplot.

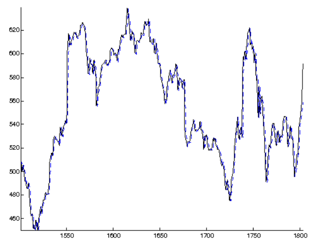


(b) Long-term NMSE boxplot.

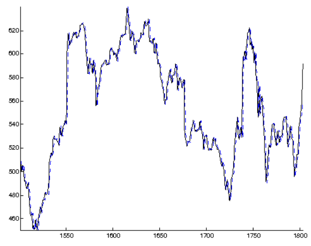


(c) Training time boxplot.

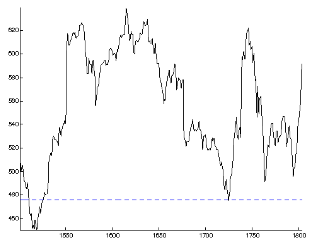
Figure 4.26: Boxplot of the results from all algorithms on the Google stock experiment.



(a) ESIGMN

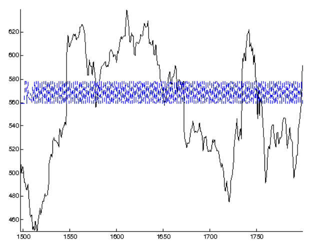


(b) RecIGMN

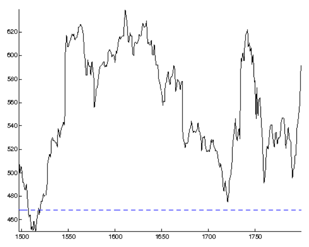


(c) RecIGMNn

Figure 4.27: One-step prediction example outputs of best and worst algorithms in the Google stock time-series (test set only). Original series in black, estimates in blue.

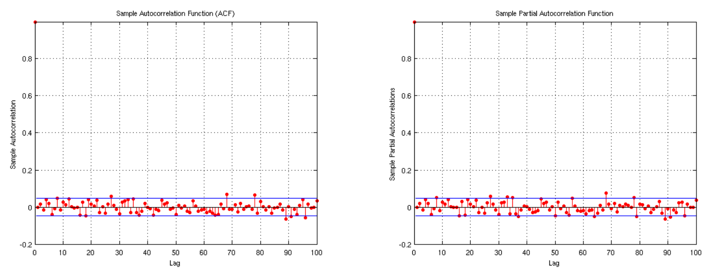


(a) TDRecIGMNn



(b) TDIGMNn

Figure 4.28: Long-term prediction example outputs of best and worst algorithms in the Google stock time-series. Original series in black, estimates in blue.



(a) Autocorrelation Function (b) Partial Autocorrelation Function

Figure 4.29: Autocorrelation and partial autocorrelation functions of the differentiated google stock prices time-series.

No algorithm was capable of overcoming the trivial solution in the one-step prediction task of this experiment. This is due to the random walk nature of the stock market (MALKIEL; MCCUE, 1985) (SITTE; SITTE, 2002), which can also be observed in this time-series' partial autocorrelation function (figure 4.25(c)): except for an almost maximum correlation for the first lag (which can be eliminated by differentiating the time-series), almost all remaining lags are not statistically significant, resulting in a random walk (figures 4.29(a) and 4.29(b)).

Hence, we can only point out the algorithms that at least matched the trivial solution in the one-step prediction task: Elman, ESN, TDNN, IGMN, ESIGMN and RecIGMN. For the long-term prediction task, TDRecIGMNn got the best result, with all algorithms being better than the mean solution. ESN was the fastest algorithm in this experiment.

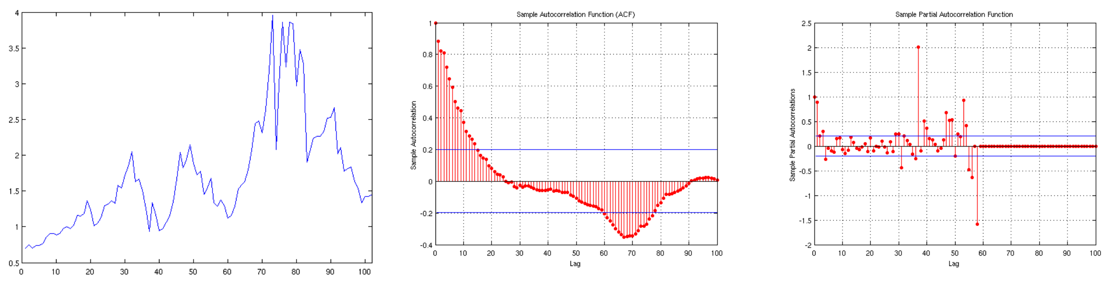
4.8 U.S. Interest Rate

This financial dataset consists of 102 observations from the U.S. interest rates, and can be seen in figure 4.30. The first 80 observations were used for training, while the remaining 22 were used for testing. For the TD algorithms, 1 extra time-delay was added to the input (and 1 data point was removed from the training set). For this experiment, $NMSE_t$ = 1 corresponds to $NMSE$ = 0.3421. Results are summarized in table 4.9 with median values and MADs, and also in figure 4.31. Comparison of all algorithms accounting for statistical significance can be seen in tables B.22 for 1-step prediction errors, table B.23 for long-term prediction errors and table B.24 for runtimes (appendix B). Figures A.15 and A.16 show test results for all algorithms on the 1-step and long-term tasks, respectively (appendix A), while figures 4.32 and 4.33 show only best and worst algorithms' results.

For this experiment, ESN achieved the best result for one-step predictions, followed by the RecIGMN, being both the only algorithms better than the trivial solution. The ESN was also the best algorithm for long-term predictions in this time-series, followed by the IGMNn and the TDRecIGMNn, those three also being the only ones to overcome the mean solution. The ESN and IGMNn were the fastest algorithms in this experiment.

4.9 Parameter Tuning

The ESIGMN and MIGMN introduce extra parameters to the IGMN algorithm: the reservoir size for ESIGMN and the merging parameter for MIGMN. This section aims to

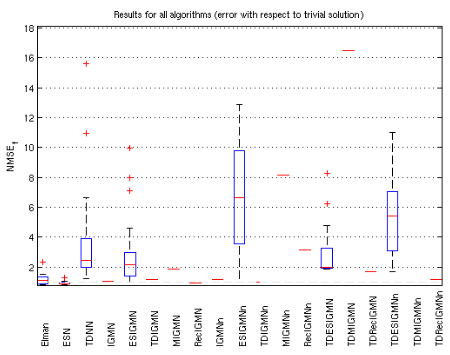


(a) Original series. (b) Autocorrelation function. (c) Partial autocorrelation function.

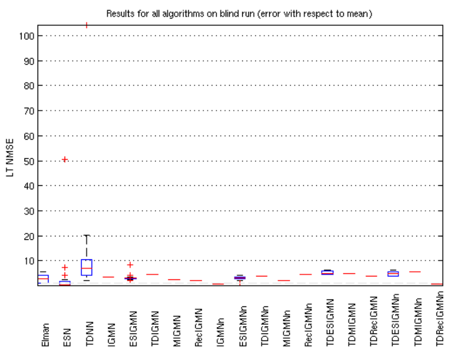
Figure 4.30: The U.S. interest rate time-series.

	$NMSE_t$	LT NMSE	Epochs	Runtime	Clusters
Elman	1.10 (0.2)	2.86 (1.3)	17.1 (5.0)	0.46 (0.2)	0.0 (0.0)
ESN	**0.89** (0.0)	**0.37** (0.1)	6.8 (0.0)	**0.10** (0.0)	0.0 (0.0)
TDNN	2.45 (0.6)	6.86 (3.0)	8.4 (1.0)	0.14 (0.0)	0.0 (0.0)
IGMN	1.06 (0.0)	3.53 (0.0)	1.0 (0.0)	0.15 (0.0)	4.0 (0.0)
ESIGMN	2.18 (0.8)	3.12 (0.2)	1.0 (0.0)	0.18 (0.0)	3.0 (0.0)
TDIGMN	1.17 (0.0)	4.61 (0.0)	1.0 (0.0)	0.16 (0.0)	4.0 (0.0)
MIGMN	1.88 (0.0)	2.52 (0.0)	1.0 (0.0)	0.18 (0.0)	6.0 (0.0)
RecIGMN	**0.92** (0.0)	2.28 (0.0)	1.0 (0.0)	0.17 (0.0)	2.0 (0.0)
IGMNn	1.19 (0.0)	**0.82** (0.0)	1.0 (0.0)	**0.10** (0.0)	6.0 (0.0)
ESIGMNn	6.63 (3.2)	3.06 (0.3)	1.0 (0.0)	0.13 (0.0)	4.0 (0.0)
TDIGMNn	1.00 (0.0)	4.00 (0.0)	1.0 (0.0)	0.11 (0.0)	5.0 (0.0)
MIGMNn	8.19 (0.0)	2.02 (0.0)	1.0 (0.0)	0.12 (0.0)	7.0 (0.0)
RecIGMNn	3.15 (0.0)	4.68 (0.0)	1.0 (0.0)	0.12 (0.0)	4.0 (0.0)
TDESIGMN	1.98 (0.1)	4.78 (0.2)	1.0 (0.0)	0.18 (0.0)	2.0 (0.0)
TDMIGMN	16.49 (0.0)	4.91 (0.0)	1.0 (0.0)	0.18 (0.0)	5.0 (0.0)
TDRecIGMN	1.70 (0.0)	3.91 (0.0)	1.0 (0.0)	0.18 (0.0)	4.0 (0.0)
TDESIGMNn	5.43 (1.9)	5.06 (0.9)	1.0 (0.0)	0.12 (0.0)	4.0 (0.5)
TDMIGMNn	18.15 (0.0)	5.64 (0.0)	1.0 (0.0)	0.12 (0.0)	7.0 (0.0)
TDRecIGMNn	1.15 (0.0)	**0.65** (0.0)	1.0 (0.0)	0.12 (0.0)	5.0 (0.0)

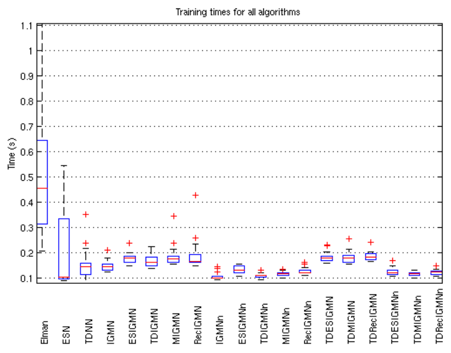
Table 4.9: Results of the U.S. interest rate experiment. Median values outside parenthesis, MADs inside. Best errors and runtimes in bold, accounting for statistical similarities. Each one-epoch algorithm better than the trivial solution (one-step) and the mean solution (long-term) is underlined, as well as the best errors and runtimes among the IGMN-based algorithms, also accounting for statistical similarities.



(a) $NMSE_t$ boxplot.

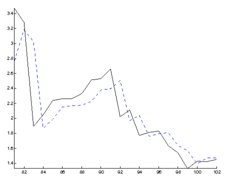


(b) Long-term NMSE boxplot.

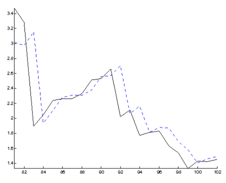


(c) Training time boxplot.

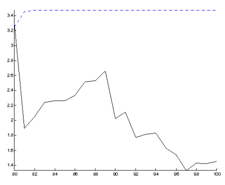
Figure 4.31: Boxplot of the results from all algorithms on the U.S. interest rate experiment.



(a) ESN

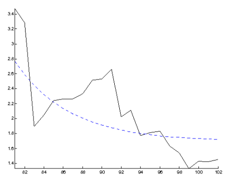


(b) RecIGMN

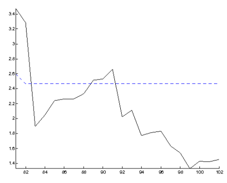


(c) TDMIGMNn

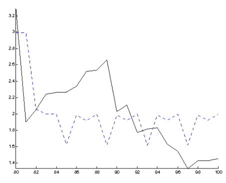
Figure 4.32: One-step prediction example outputs of best and worst algorithms in the U.S. interest rate time-series (test set only). Original series in black, estimates in blue.



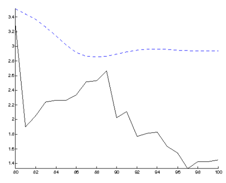
(a) ESN



(b) IGMNn



(c) TDRecIGMNn



(d) TDNN

Figure 4.33: Long-term prediction example outputs of best and worst algorithms in the U.S. interest rate time-series. Original series in black, estimates in blue.

evaluate those algorithms (and their variants) with different values for those extra parameters (the reservoir spectral radius will not be analyzed).

4.9.1 Reservoir Size

In this section, all reservoir-based algorithms (ESN, ESIGMN, ESIGMNn, TDESIGMN and TDESIGMNn) are evaluated with different reservoir sizes in the same 8 time-series used in previous experiments. Figure 4.34 shows how the number of created Gaussian components is affected by the reservoir size (ESN is not included here since it does not have Gaussian components). Figure 4.35 shows how the one-step prediction error is affected by the reservoir size, and figure 4.36 does the same in relation to the long-term prediction error.

As can be seen in figure 4.34, there is a slight pattern of decreasing the number of created Gaussian components in the ESIGMN as the reservoir size increases. This can be attributed to the fact that the reservoir layer is doing, besides temporal processing, a linearization of the input space, such as what happens in the hidden layer of multilayer perceptrons in general. Since each Gaussian component does a local linear regression, it is expected that the more linearized the input space is, more of it can be modeled by a single Gaussian component. The TDESIGMN has a similar pattern, although an opposite behavior can be observed in the airline passengers experiment. The large number of time-delays in this experiment could be an explanation, since the reservoir spectral radius and scaling parameters were not adjusted to handle such a large input dimension. The naïve versions, on the other hand, had a somewhat evident increasing number of Gaussian components as the reservoir size increases, which could be attributed to their inability to handle covariances inside reservoir activations (which are huge, since they are all produced from the same sources). Thus, we conclude that the full versions can exploit the reservoir much better, creating more economical representations.

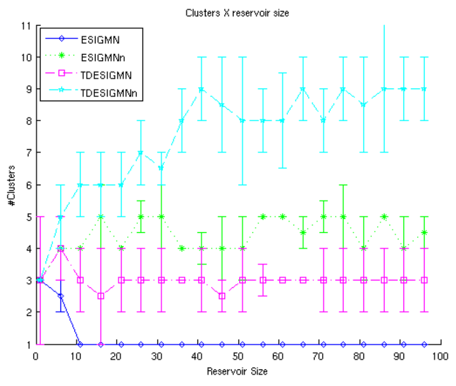
In figure 4.35 it can be seen that there is a exponential decayment in one-step prediction error for the chaotic time-series as the reservoir size increases, except for the naïve versions. Since those time-series are deterministc, it is expected that we can not observe any overfitting effect. On the other hand, this effect can be observed in the stochastic time-series, showing that the IGMN-based algorithms are not immune to overfitting, needing some careful choice of reservoir size. This is not so apparent in long-term predictions (figure 4.36), as the ESIGMN did not increase significantly in almost all experiments. There is not a clear pattern for the other algorithms in long-term predictions.

4.9.2 Merging Parameter

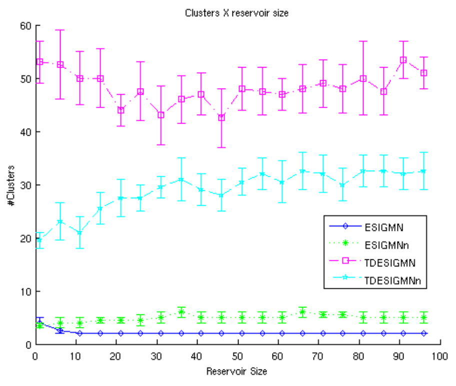
In this section, all MIGMN-based algorithms (MIGMN, MIGMNn, TDMIGMN and TDMIGMNn) are evaluated with different alpha values (the merging parameter) in the same 8 time-series used in previous experiments. Figure 4.37 shows how the number of created Gaussian components is affected by the alpha value. Figure 4.38 shows how the one-step prediction error is affected by the alpha value, and figure 4.39 does the same in relation to the long-term prediction error.

Apparently, there is no significant pattern for the number of created Gaussian components in relation to the alpha parameter, except when $\alpha = 1$. On this extreme, the MIGMN acts almost like a simple IGMN, since the context stays constant for all the algorithm's lifetime. Thus, there is less complexity to be encoded in Gaussian components and less of them are created.

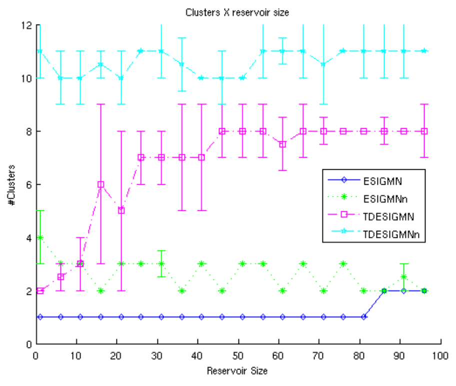
There is no consistent visible pattern for the one-step and long-term prediction errors



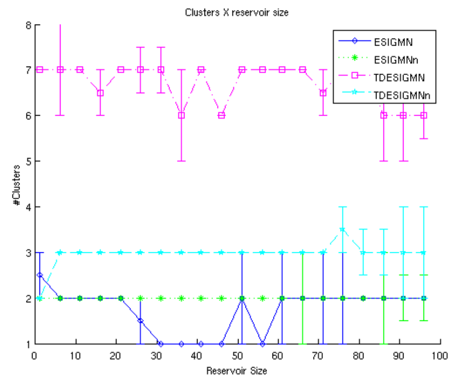
(a) Yearly Sunspot Numbers.



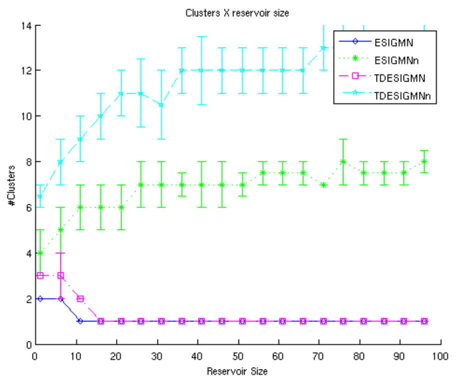
(b) Monthly Sunspot Numbers.



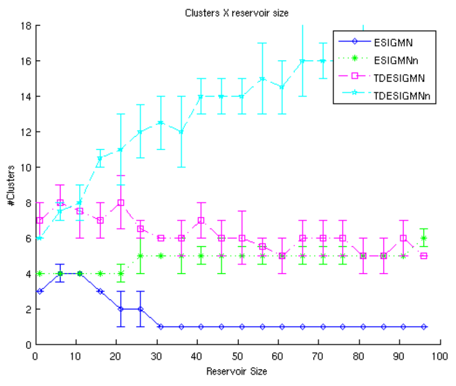
(c) Airline Passengers.



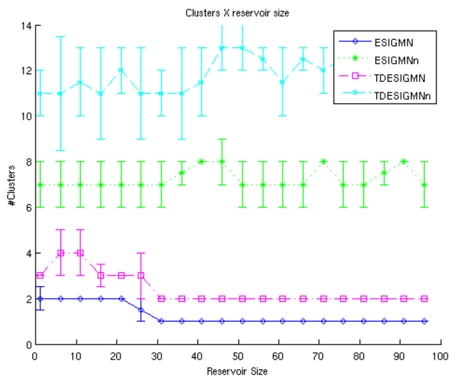
(d) Log-Differentiated Airline Passengers.



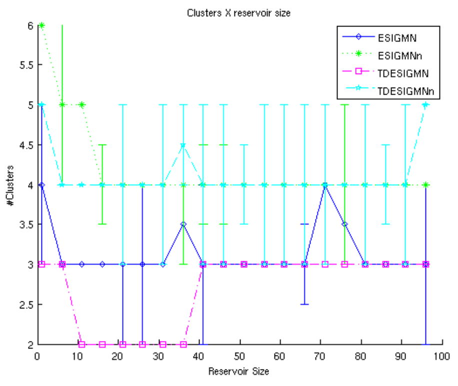
(e) Mackey-Glass ($\tau = 17$).



(f) Mackey-Glass ($\tau = 30$).

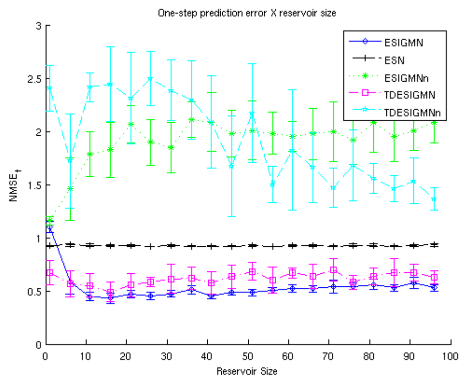


(g) Google Stock Prices.

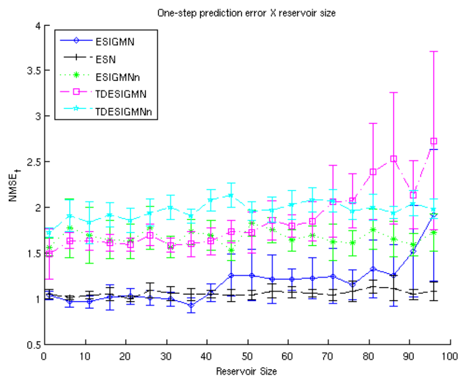


(h) U.S. Interest Rates.

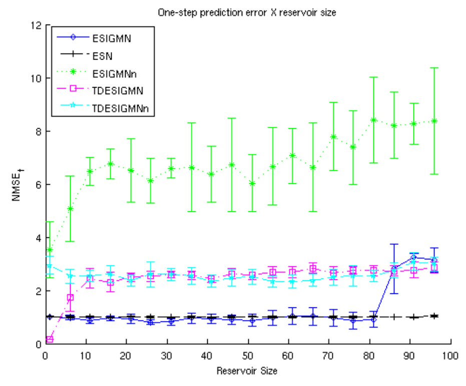
Figure 4.34: Number of created Gaussian components in relation to reservoir size.



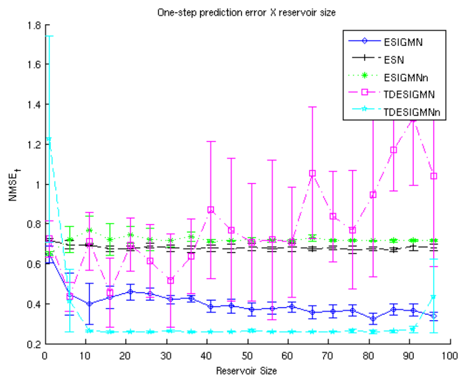
(a) Yearly Sunspot Numbers.



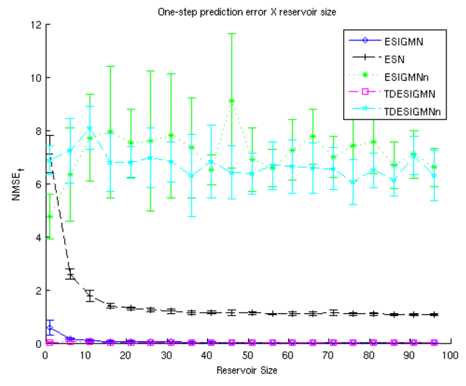
(b) Monthly Sunspot Numbers.



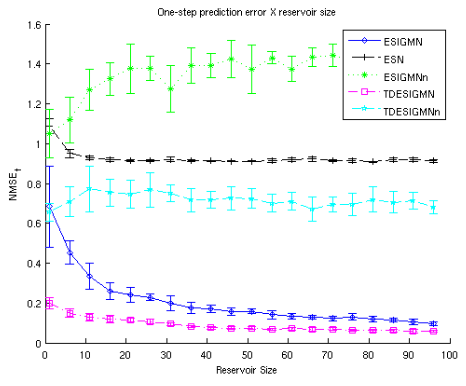
(c) Airline Passengers.



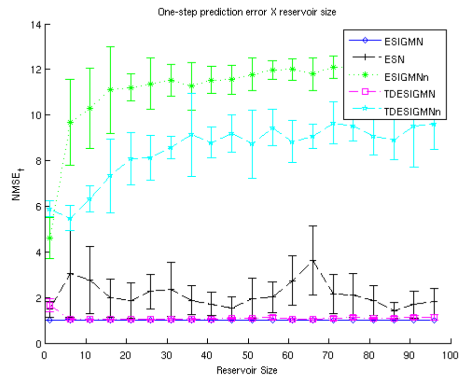
(d) Log-Differentiated Airline Passengers.



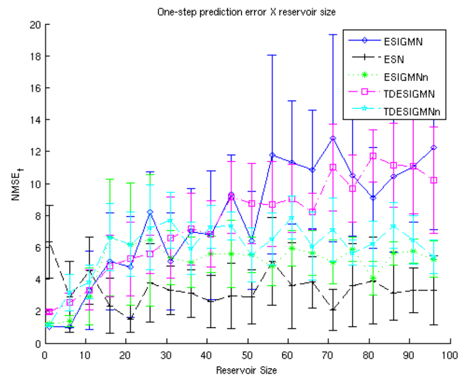
(e) Mackey-Glass ($\tau = 17$).



(f) Mackey-Glass ($\tau = 30$).

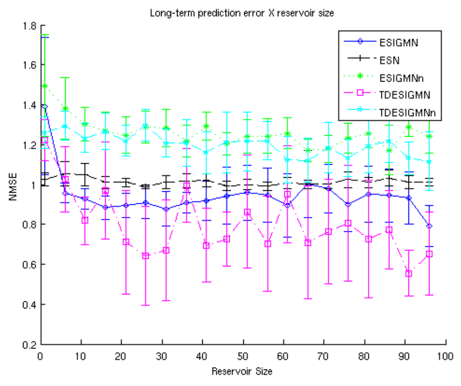


(g) Google Stock Prices.

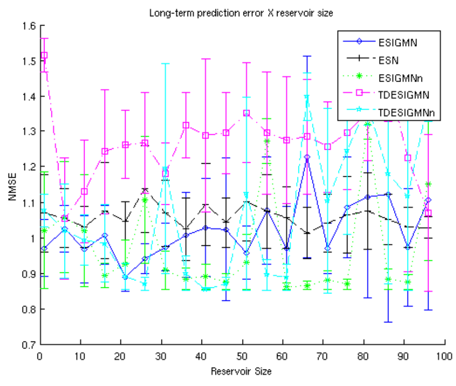


(h) U.S. Interest Rates.

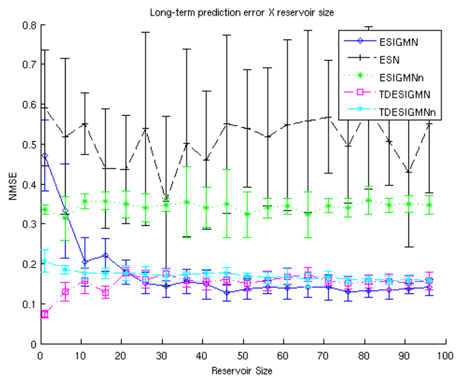
Figure 4.35: One-step prediction error in relation to reservoir size.



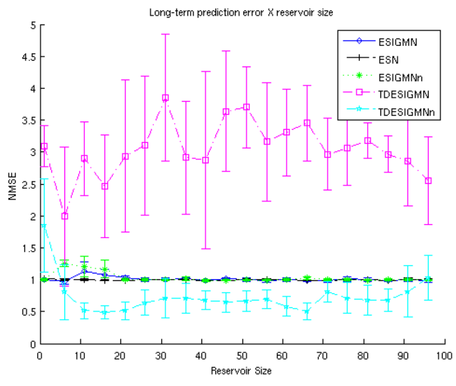
(a) Yearly Sunspot Numbers.



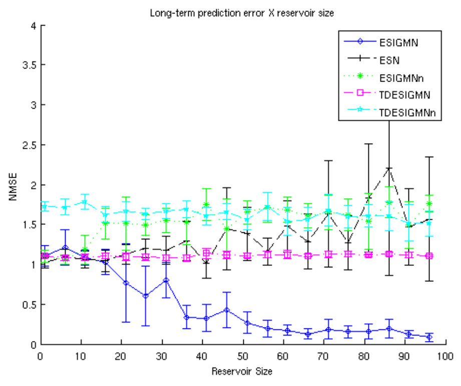
(b) Monthly Sunspot Numbers.



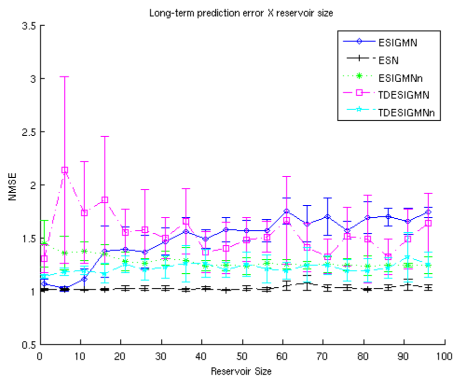
(c) Airline Passengers.



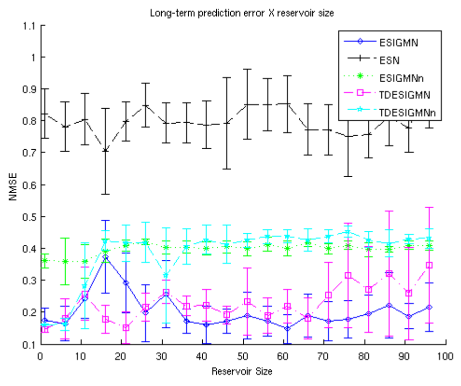
(d) Log-Differentiated Airline Passengers.



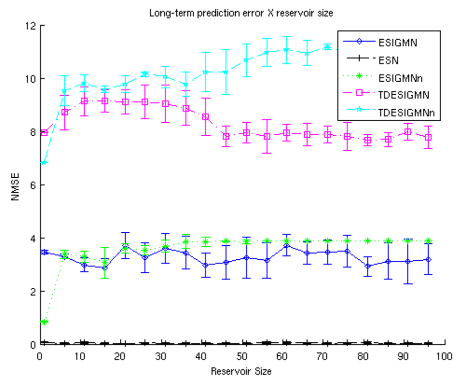
(e) Mackey-Glass ($\tau = 17$).



(f) Mackey-Glass ($\tau = 30$).

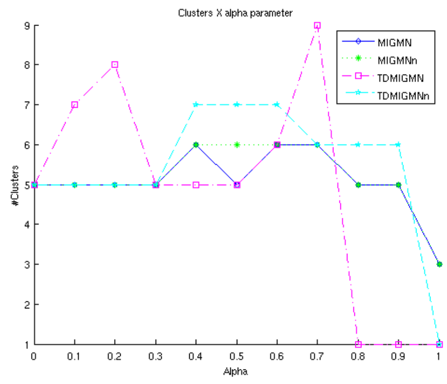


(g) Google Stock Prices.

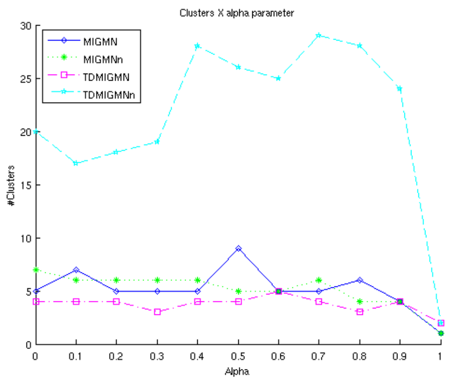


(h) U.S. Interest Rates.

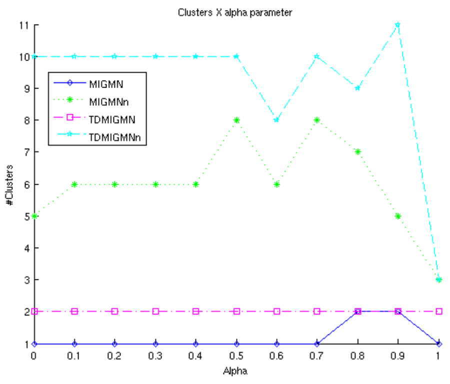
Figure 4.36: Long-term prediction error in relation to reservoir size.



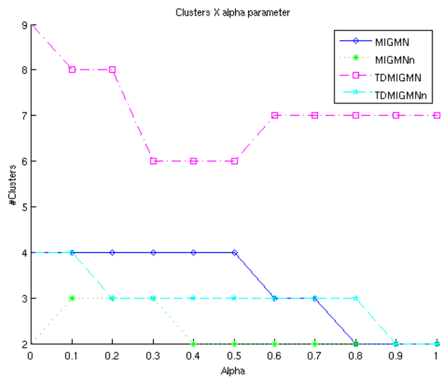
(a) Yearly Sunspot Numbers.



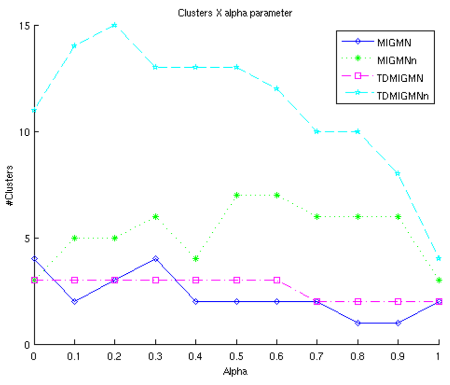
(b) Monthly Sunspot Numbers.



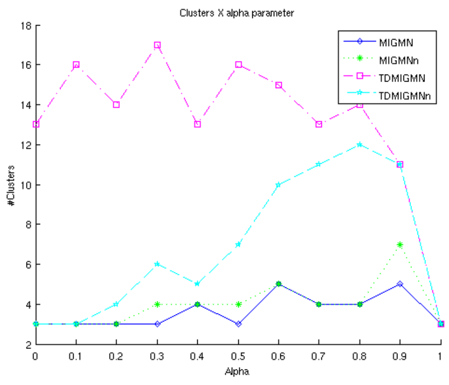
(c) Airline Passengers.



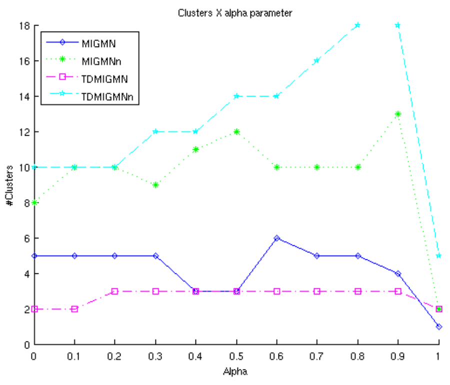
(d) Log-Differentiated Airline Passengers.



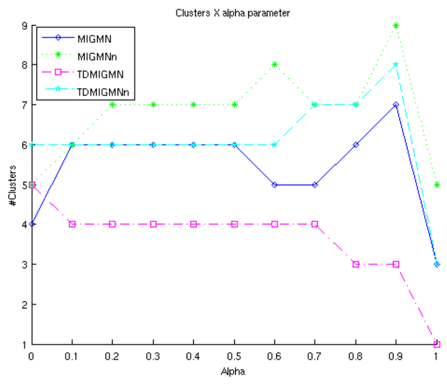
(e) Mackey-Glass ($\tau = 17$).



(f) Mackey-Glass ($\tau = 30$).

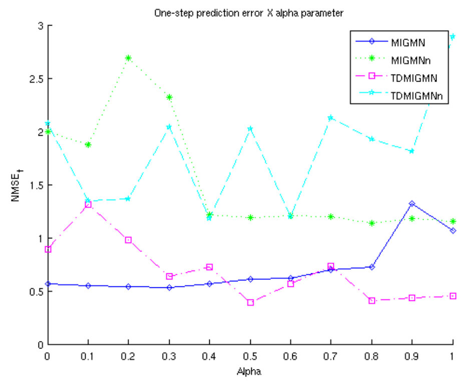


(g) Google Stock Prices.

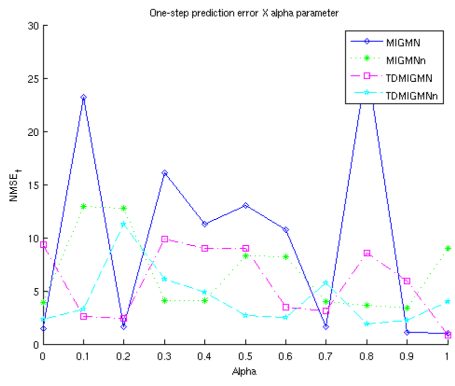


(h) U.S. Interest Rates.

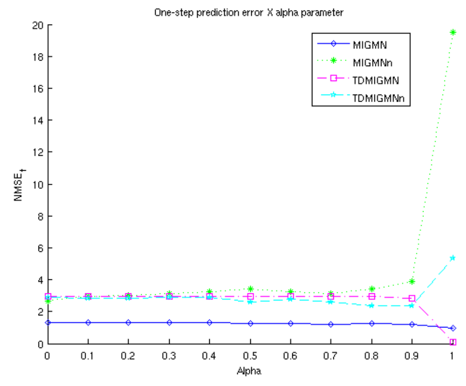
Figure 4.37: Number of created Gaussian components in relation to alpha parameter.



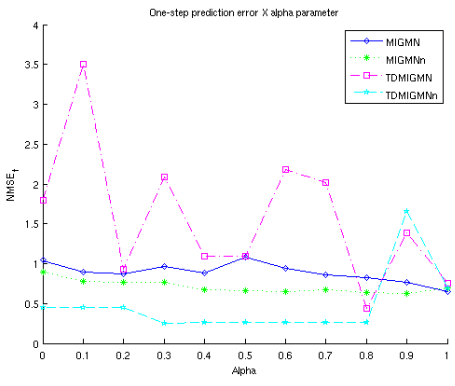
(a) Yearly Sunspot Numbers.



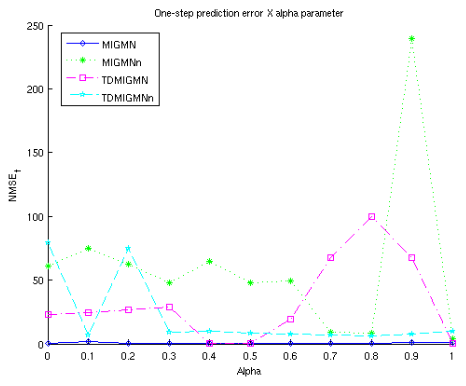
(b) Monthly Sunspot Numbers.



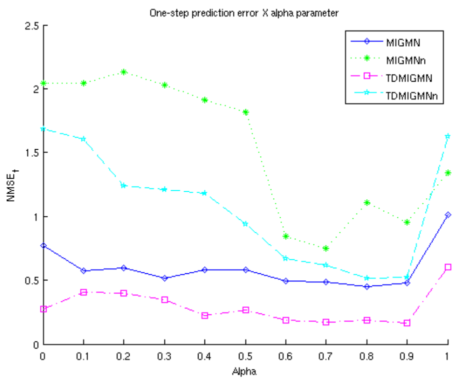
(c) Airline Passengers.



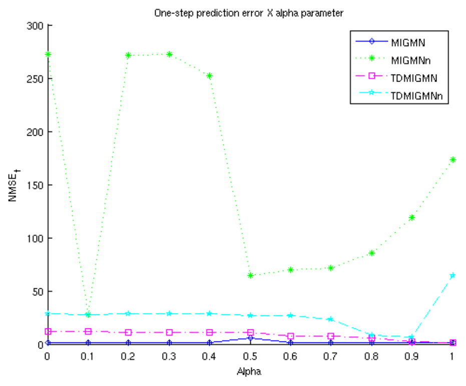
(d) Log-Differentiated Airline Passengers.



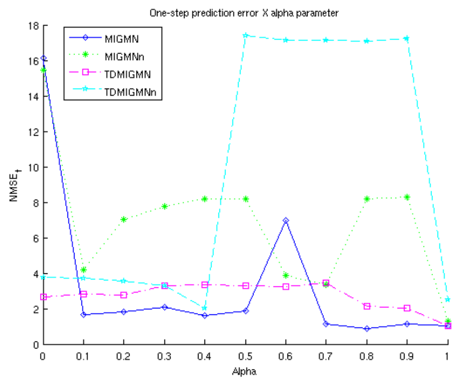
(e) Mackey-Glass ($\tau = 17$).



(f) Mackey-Glass ($\tau = 30$).



(g) Google Stock Prices.



(h) U.S. Interest Rates.

Figure 4.38: One-step prediction error in relation to alpha parameter.

	Corr.Dim.	Alg.Compl.	Mean	Median	Variance	Std.Dev.	IQR	Skewness	Kurtosis	Hjorth Mob.	Hjorth Compl.	Detr.Fluct. Analysis
Elman	0.012	0.080	0.185	0.195	0.227	0.067	0.085	**-0.848**	**-0.942**	0.097	-0.499	-0.331
ESN	-0.281	0.204	0.349	0.334	0.440	0.325	0.380	-0.533	**-0.772**	0.040	-0.302	-0.025
TDNN	-0.090	-0.442	-0.009	-0.012	-0.040	0.032	0.017	0.195	0.282	-0.370	-0.165	0.175
IGMN	0.017	-0.004	0.324	0.340	0.352	0.173	0.178	**-0.872**	**-0.929**	0.083	-0.380	-0.196
ESIGMN	-0.233	-0.121	-0.250	-0.258	-0.270	-0.176	-0.176	-0.023	-0.008	-0.107	**-0.723**	-0.406
TDIGMN	0.303	-0.338	-0.118	-0.133	-0.132	0.001	-0.002	0.658	**0.865**	-0.490	**0.741**	0.472
MIGMN	0.475	0.213	-0.327	-0.319	-0.333	-0.356	-0.356	0.543	0.662	-0.150	0.616	0.336
RecIGMN	0.277	0.213	-0.127	-0.107	-0.125	-0.247	-0.251	-0.689	-0.626	0.183	-0.400	-0.550
IGMNn	-0.175	-0.004	0.313	0.288	0.419	0.335	0.405	-0.245	-0.474	-0.276	-0.027	0.304
ESIGMNn	-0.365	-0.180	0.060	0.011	0.116	0.305	0.356	0.558	0.375	-0.206	0.277	0.291
TDIGMNn	-0.157	-0.272	0.361	0.323	0.445	0.499	0.556	0.119	0.027	-0.464	0.401	0.495
MIGMNn	-0.182	-0.095	0.050	0.009	0.101	0.261	0.304	0.598	0.466	-0.151	0.560	0.367
RecIGMNn	-0.094	-0.134	0.042	0.006	0.084	0.233	0.268	0.547	0.452	-0.146	0.608	0.350
TDESIGMN	-0.165	-0.041	-0.088	-0.101	-0.105	0.003	0.002	**0.724**	**0.794**	-0.297	0.172	0.402
TDMIGMN	-0.053	0.075	-0.328	-0.342	-0.356	-0.195	-0.200	**0.869**	**0.915**	-0.065	0.301	0.195
TDRecIGMN	-0.591	-0.156	0.167	0.158	0.123	0.249	0.224	0.078	0.076	0.099	-0.517	-0.172
TDESIGMNn	-0.528	-0.280	0.293	0.245	0.351	0.514	0.562	0.515	0.338	-0.330	0.219	0.494
TDMIGMNn	-0.226	-0.229	-0.087	-0.118	-0.095	0.127	0.134	0.660	0.628	-0.081	0.323	0.158
TDRecIGMNn	-0.067	-0.228	0.141	0.110	0.176	0.304	0.331	0.435	0.365	-0.162	0.643	0.390

Table 4.10: Correlation between each algorithm's one-step prediction error and each time-series descriptor. Significant values in bold.

	Corr.Dim.	Alg.Compl.	Mean	Median	Variance	Std.Dev.	IQR	Skewness	Kurtosis	Hjorth Mob.	Hjorth Compl.	Detr.Fluct. Analysis
Elman	0.602	-0.343	-0.109	-0.084	-0.121	-0.247	-0.260	-0.266	-0.045	-0.514	0.088	0.184
ESN	0.675	-0.394	-0.252	-0.246	-0.241	-0.274	-0.267	-0.289	-0.156	-0.346	0.256	-0.092
TDNN	-0.219	0.589	-0.328	-0.319	-0.330	-0.363	-0.360	-0.032	-0.395	**0.917**	-0.369	-0.544
IGMN	-0.418	-0.480	**0.893**	**0.877**	**0.906**	**0.925**	**0.920**	-0.082	0.092	-0.593	0.261	**0.717**
ESIGMN	0.014	-0.165	-0.340	-0.370	-0.306	-0.164	-0.125	0.129	0.113	-0.284	-0.232	-0.282
TDIGMN	-0.018	0.396	-0.031	-0.023	-0.047	-0.068	-0.080	0.547	0.555	0.161	0.411	0.360
MIGMN	0.065	0.077	-0.431	-0.443	-0.389	-0.377	-0.335	-0.394	-0.516	0.015	**-0.711**	-0.680
RecIGMN	0.287	-0.108	0.258	0.297	0.235	0.020	-0.015	**-0.831**	**-0.751**	0.107	-0.176	-0.119
IGMNn	0.507	-0.223	-0.211	-0.216	-0.193	-0.165	-0.152	0.009	0.064	-0.154	0.508	0.000
ESIGMNn	-0.075	0.388	-0.139	-0.173	-0.024	-0.030	0.057	0.116	-0.252	0.147	0.159	0.004
TDIGMNn	-0.173	0.316	-0.163	-0.208	-0.056	0.013	0.101	0.255	0.038	-0.166	-0.014	-0.003
MIGMNn	**0.763**	-0.122	-0.173	-0.128	-0.223	-0.383	-0.430	-0.264	-0.046	-0.035	0.371	0.058
RecIGMNn	-0.317	-0.517	0.526	0.499	0.517	0.673	0.665	0.379	0.557	-0.520	0.465	0.603
TDESIGMN	-0.398	0.687	-0.253	-0.255	-0.247	-0.234	-0.225	0.071	-0.314	**0.983**	-0.344	-0.548
TDMIGMN	0.005	0.585	-0.385	-0.370	-0.400	-0.436	-0.444	-0.282	-0.420	**0.750**	-0.460	**-0.782**
TDRecIGMN	0.173	0.397	-0.143	-0.132	-0.152	-0.197	-0.204	0.511	0.532	0.114	0.482	0.338
TDESIGMNn	-0.687	-0.086	0.341	0.292	0.421	0.547	0.608	0.383	0.062	-0.106	0.056	0.390
TDMIGMNn	-0.422	0.027	0.282	0.247	0.343	0.423	0.469	**0.708**	0.520	-0.243	0.478	**0.745**
TDRecIGMNn	-0.064	-0.062	0.355	0.340	0.448	0.337	0.394	-0.464	-0.563	-0.378	-0.115	0.212

Table 4.11: Correlation between each algorithm's long-term prediction error and each time-series descriptor. Significant values in bold.

in relation to the alpha parameter, seeming that it needs specific tuning for each problem.

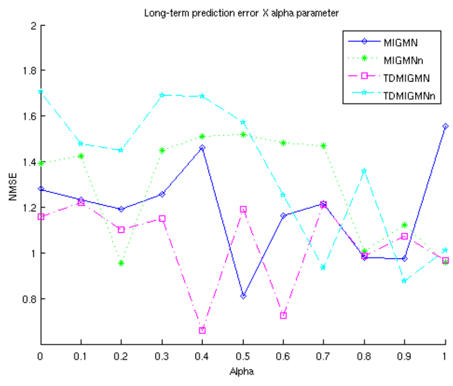
4.10 Summary

Global statistics for all algorithms were gathered in order to provide more general comparisons.

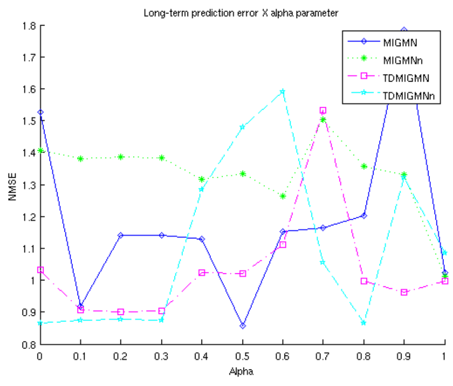
Table 4.10 shows correlations of each algorithm's one-step prediction error with each time-series descriptor (table 4.1). It shows that many tested algorithm were not significantly sensitive to any of the descriptors for one-step predictions, except for the Elman Network, ESN, IGMN, TDIGMN, TDESIGMN, TDMIGMN, which are specially sensitive to the skewness and kurtosis of the time-series, and ESIGMN, which is sensitive to the Hjorth Complexity parameter (its performance increases with higher values of this descriptor). The TDMIGMN is also sensitive to the Hjorth Complexity (its performance degrades with higher values of this descriptor).

Table 4.11 shows correlations of each algorithm's long-term prediction error with each time-series descriptor. The following algorithms were significantly sensitive to any of the descriptors for long-term predictions: TDNN, IGMN, MIGMN, RecIGMN, MIGMNn, TDESIGMN and TDMIGMN.

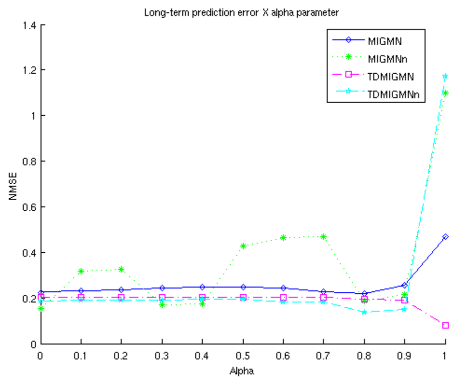
Table 4.12 shows robust estimates of general performance of each algorithm by using MAD-scores (JAIN; NANDAKUMAR; ROSS, 2005), which are robust alternatives to the



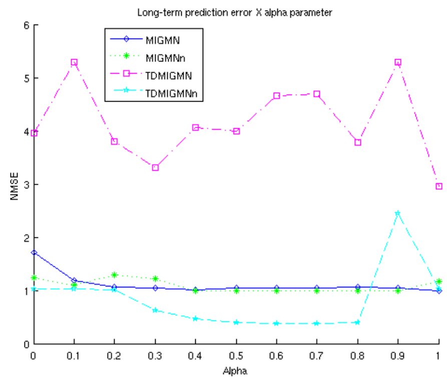
(a) Yearly Sunspot Numbers.



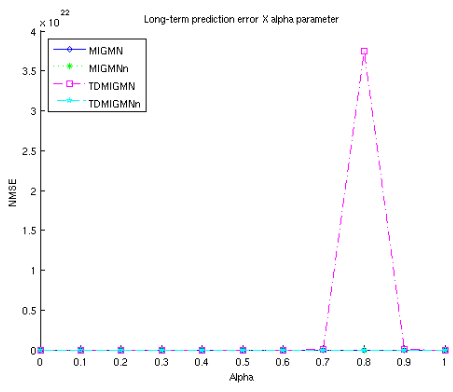
(b) Monthly Sunspot Numbers.



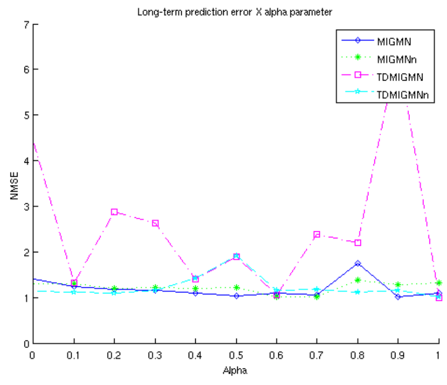
(c) Airline Passengers.



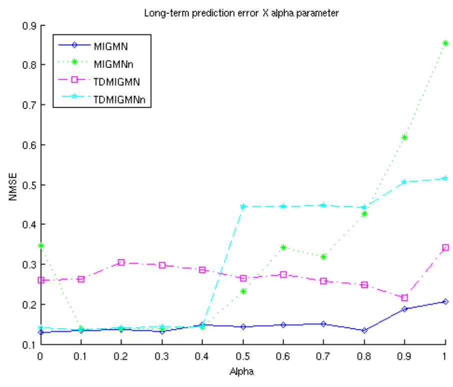
(d) Log-Differentiated Airline Passengers.



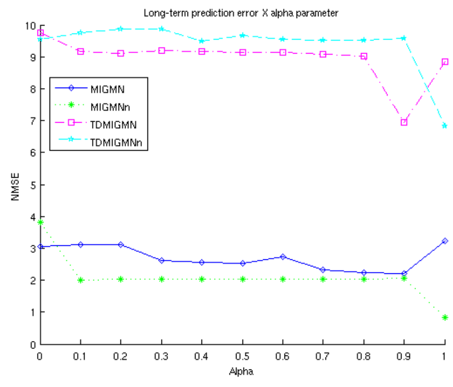
(e) Mackey-Glass ($\tau = 17$).



(f) Mackey-Glass ($\tau = 30$).



(g) Google Stock Prices.



(h) U.S. Interest Rates.

Figure 4.39: Long-term prediction error in relation to alpha parameter.

	One-Step	Long-Term	Runtime
Elman	-0.14 (0.1)	0.21 (0.8)	1.06 (1.6)
ESN	-0.20 (0.1)	-0.15 (0.0)	**-0.80** (0.2)
TDNN	-0.34 (1.1)	0.30 (1.8)	-0.60 (0.4)
IGMN	-0.10 (0.1)	-0.01 (1.0)	1.91 (2.7)
ESIGMN	-0.59 (0.4)	-0.23 (0.2)	1.62 (2.4)
TDIGMN	-0.98 (0.6)	0.26 (1.2)	2.06 (3.0)
MIGMN	0.12 (1.0)	**-0.28** (0.3)	1.95 (2.7)
RecIGMN	-0.13 (0.2)	0.59 (1.0)	2.31 (3.0)
IGMNn	0.35 (0.5)	-0.11 (0.1)	0.66 (1.6)
ESIGMNn	4.61 (3.9)	0.52 (0.3)	1.10 (2.0)
TDIGMNn	1.34 (1.8)	0.25 (1.5)	0.83 (1.8)
MIGMNn	6.95 (6.4)	0.13 (0.8)	1.31 (2.0)
RecIGMNn	3.61 (2.9)	0.09 (0.8)	0.90 (1.8)
TDESIGMN	-1.08 (0.4)	-0.19 (1.3)	1.93 (2.8)
TDMIGMN	**-1.11** (0.2)	0.17 (1.8)	2.59 (3.4)
TDRecIGMN	-0.81 (0.6)	0.31 (1.6)	2.82 (3.7)
TDESIGMNn	3.89 (3.4)	0.08 (1.5)	1.44 (2.4)
TDMIGMNn	4.03 (4.7)	0.65 (1.5)	1.16 (2.1)
TDRecIGMNn	3.47 (2.8)	-0.06 (0.8)	0.91 (1.9)

Table 4.12: Median MAD-scores for all algorithms in all experiments (less is better). MADs for these values inside parenthesis. Best scores in bold, best IGMN-based algorithms' scores underlined.

z-score, used to do scale-invariant comparisons of values from different distributions. The MAD-score formula is as follows:

$$MAD - score(x) = \frac{x - median(X)}{MAD} \tag{4.6}$$

Those scores were computed for each one-step prediction errors, long-term prediction errors and runtimes in each experiment, and the median was taken. As can be seen in the table, TDMIGMN was the best algorithm for one-step predictions in general, closely followed by TDESIGMN and TDIGMN, while the MIGMN was the best one for long-term predictions, followed by the ESIGMN. Averaging these two columns, TDESIGMN and ESIGMN show up as the best algorithms overall. The ESN was the fastest algorithm, while the IGMNn was the fastest IGMN-based algorithm, which was expected since it has not any context and is a naïve implementation. Accounting only for the non-TD, non-naïve proposed approaches, ESIGMN was the fastest one.

As for the inclusion of time-delays, all algorithms, except for the IGMNn, benefited from it in general for one-step predictions, probably because it augments them with an exact short-term memory. This may be an indication that the ESIGMN, MIGMN and RecIGMN memory contexts are complementary with time-delay lines (like human short-term memory and working memory, respectively). For long-term predictions, time-delays inclusion was beneficial only for RecIGMN, IGMNn, and RecIGMNn. The naïve versions were worse for all algorithms in the one-step prediction task, but the naïve versions of IGMN, TDIGMN, RecIGMN and TDRecIGMN were better than their full counterparts for long-term predictions.

From all those experiments and results, we conclude that the ESIGMN and the MIGMN (and their TD versions) improve upon classic temporal neural networks and the simple time-delayed IGMN (TDIGMN) in general, although with not very much confidence due to large MADs and small time-series sample size. They have also a reasonable speed, which along with the IGMN one-shot learning capabilities, turn them into good options for real-time applications. The RecIGMN algorithm would need to be improved in order to become competitive in general. Since its temporal context is inspired by the RecSOM, which is one of the best temporal SOM variants, we expect it (as a general idea) to be much better, and attribute its present failure to our particular implementation, which may not be a reasonable adaptation of the RecSOM for the IGMN algorithm. One possible solution is to weight the contributions of the current input and temporal context differently, by calculating the likelihoods in two parts. This approach was tried in (PINTO; ENGEL; HEINEN, 2011b) with success, but the extra weighting α parameter was removed from this dissertation due to implementation issues, and also to verify the RecIGMN capability of self-adjusting the importance of the temporal context in relation to current input without additional parameters.

5 CONCLUSIONS

This work presented 3 novel temporal extensions for the IGMN algorithm, as well as a new component creation rule.

Chapter 2 presented all algorithms used as the base for the new extensions, covering most aspects needed to implement them.

Chapter 3 described all contributions of this work, starting with the new component creation rule. This rule has enabled the IGMN to successfully learn without the need for the ϵ_{max} parameter, and gave better (more intuitive) solutions in a simple experiment. This is very important in order to achieve truly autonomous and general learning, since manually tuning some parameter involves human knowledge of each problem. Eliminating the remaining training parameters from the IGMN algorithms is something to explore in future works. After that, the TDIGMN, which uses a sliding window with past inputs, was revised (it is not a novel algorithm, but was not described before explicitly as a temporal algorithm) as a temporal enhancement for the IGMN. Then, the 3 novel extensions were described by reusing the previously shown techniques from other temporal neural networks: ESIGMN with its dynamic reservoir layer inspired by the ESN; MIGMN, which uses an exponential moving average of the reconstructed inputs and outputs as its temporal context, similar to the MSOM; and RecIGMN, inspired by the RecSOM and Elman Network, which adds feedback connections from its Gaussian components' activations to its inputs.

In chapter 4, those new algorithms were compared to classic temporal neural networks and static IGMN, in order to assess their performances on the time-series prediction task, using 8 different time-series. The new algorithms were tested with their naïve and multivariate versions, as well as with and without additional time-delays (such as the ones used by TDIGMN). A total of 19 algorithms were tested and results analyzed. We found the ESIGMN and MIGMN to be useful as online incremental one-shot temporal algorithms, even being better than classic temporal neural networks in many experiments. We think that the RecIGMN, although not being a good addition to this set, has potential if properly implemented, with a present-past weighting parameter and maybe more information from the Gaussian component's activations as feedback, such as error terms. The MIGMN was also presented without a present-past weighting parameter, and this possibility should be explored. The ESIGMN has external connection to its reservoir coming only from its inputs, and a version with feedback connections from outputs is yet to be explored. We also found that time-delays are generally beneficial to the new algorithms, complementing their unbounded inexact memories with a bounded exact one (this could be seen as the complimentary roles of short-term memory and working memory in our brains, respectively). All proposed algorithms achieved very good performance in terms of execution time too, making them good candidates for real-time applications. It is yet to be explored how they handle higher dimensionality inputs, such as videos. The matrix inversion is still the bottleneck of all multivariate IGMN algorithms, and the naïve versions proved to be inefficient for temporal tasks, due to their inability to capture the covariances between the highly correlated values of the temporally augmented inputs of the new algorithms.

Therefore, we conclude that the initial goals of this work were reached, which were to create online incremental one-shot temporal algorithms using the IGMN as a base. Summarizing the contributions and conclusions in this work:

- New component creation rule for the IGMN algorithm and its extensions, resulting in better performance and eliminating one manually tunable parameter;
- ESIGMN algorithm, which achieved very good results in general. It was also published in the proceedings of a national conference (PINTO; ENGEL; HEINEN, 2011a);
- MIGMN algorithm, which also achieved very good results;
- RecIGMN algorithm, which need some improvement in order to become competitive. A more complete version of the algorithm was also published in the proceedings of a national conference (PINTO; ENGEL; HEINEN, 2011b);
- Matlab implementations of all proposed algorithms and the IGMN itself, which will be reused in future works and also by our research group and external researchers (in fact, they are already being used by another artificial intelligence research groups at Universidade Federal do Rio Grande do Sul (UFRGS) and Universidade Federal de Santa Catarina (UFSC)).

As future works, we can enumerate the following:

- Verifying the benefits of including output feedback in the ESIGMN, as well as using better reservoir initialization techniques;
- Verifying the benefits of including past-present weighting parameters into the MIGMN and RecIGMN algorithms (and trying to automatically tune them online);
- Analyzing different temporal contexts for the RecIGMN, by using more information from its Gaussian components;
- Application of the proposed algorithms in reinforcement learning tasks in partially observable Markov decision processes (POMDP);
- Research for longer-term memories;
- Verifying the viability of the proposed algorithms in higher dimensional problems and also more real-world problems such as robotics.

REFERENCES

BAUM, L.; PETRIE, T. Statistical inference for probabilistic functions of finite state Markov chains. **The Annals of Mathematical Statistics**, [S.l.], v.37, n.6, p.1554–1563, 1966.

BAUM, L.; PETRIE, T.; SOULES, G.; WEISS, N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. **The annals of mathematical statistics**, [S.l.], v.41, n.1, p.164–171, 1970.

BERTHOLD, M. A time delay radial basis function network for phoneme recognition. In: CONFERENCE ON NEURAL NETWORKS, 1994. IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE., 1994 IEEE INTERNATIONAL, 1994. **Proceedings. . .** [S.l.: s.n.], 1994. v.7, p.4470–4472.

BOX, G.; JENKINS, G. **Time series analysis**: forecasting and control. [S.l.]: Prentice Hall PTR, 1994.

BUEHNER, M.; YOUNG, P. A tighter bound for the echo state property. **Neural Networks, IEEE Transactions on**, [S.l.], v.17, n.3, p.820–824, 2006.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals, and Systems (MCSS)**, [S.l.], v.2, n.4, p.303–314, 1989.

DEMPSTER, A.; LAIRD, N.; RUBIN, D. et al. Maximum likelihood from incomplete data via the EM algorithm. **Journal of the Royal Statistical Society. Series B (Methodological)**, [S.l.], v.39, n.1, p.1–38, 1977.

DOMINEY, P. F. Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. **Biological Cybernetics**, [S.l.], v.73, n.3, p.265–74, 08/1995 1995.

ELMAN, J. Finding structure in time. **Connectionist Psychology: A Text with Readings**, [S.l.], 1999.

ENGEL, P.; HEINEN, M. Concept Formation Using Incremental Gaussian Mixture Models. **Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications**, [S.l.], p.128–135, 2010.

ENGEL, P.; HEINEN, M. Incremental learning of multivariate gaussian mixture models. **Advances in Artificial Intelligence–SBIA 2010**, [S.l.], p.82–91, 2011.

FAHLMAN, S. **The cascade-correlation learning architecture**. [S.l.]: DTIC Document, 1990.

GIBBONS, J.; CHAKRABORTI, S. **Nonparametric statistical inference**. [S.l.]: CRC Press, 2003. v.168.

GRASSBERGER, P.; PROCACCIA, I. Measuring the strangeness of strange attractors. **Physica D: Nonlinear Phenomena**, [S.l.], v.9, n.1-2, p.189–208, 1983.

HAGAN, M.; MENHAJ, M. Training feedforward networks with the Marquardt algorithm. **Neural Networks, IEEE Transactions on**, [S.l.], v.5, n.6, p.989–993, 1994.

HEINEN, M. **A connectionist approach for incremental function approximation and on-line tasks**. 2011. Tese (Doutorado em Ciência da Computação) — PhD thesis, Universidade Federal do Rio Grande do Sul - Instituto de Informática.

HEINEN, M.; ENGEL, P. IPNN: an incremental probabilistic neural network for function approximation and regression tasks. **Proceedings of SBRN 2010**, [S.l.], p.25–30, 2010.

HEINEN, M.; ENGEL, P. Feature-Based Mapping using Incremental Gaussian Mixture Models. **Proceedings of LARS 2010**, [S.l.], p.67–72, 2010.

HEINEN, M.; ENGEL, P. An Incremental Probabilistic Neural Network for Regression and Reinforcement Learning Tasks. **Artificial Neural Networks–ICANN 2010**, [S.l.], p.170–179, 2010.

HEINEN, M.; ENGEL, P. IGMN: an incremental connectionist approach for concept formation, reinforcement learning and robotics. **Journal of Applied Computing Research**, [S.l.], v.1, n.1, p.2–19, 2011.

HEINEN, M.; ENGEL, P. Incremental Feature-Based Mapping from Sonar Data using Gaussian Mixture Models. **Proceedings of ACM SAC 2011**, [S.l.], 2011.

HEINEN, M.; ENGEL, P.; PINTO, R. IGMN: an incremental gaussian mixture network that learns instantaneously from data flows. In: IX ENIA - BRAZILIAN MEETING ON ARTIFICIAL INTELLIGENCE, 2011, Natal (RN). **Proceedings...** [S.l.: s.n.], 2011.

HJORTH, B. EEG analysis based on time domain properties. **Electroencephalography and Clinical Neurophysiology**, [S.l.], v.29, n.3, p.306–310, 1970.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, [S.l.], v.9, n.8, p.1735–1780, 1997.

HORNIK MAXWELL, K.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural networks**, [S.l.], v.2, n.5, p.359–366, 1989.

HUBER, P.; RONCHETTI, E.; MYILIBRARY. **Robust statistics**. [S.l.]: Wiley Online Library, 1981. v.1.

HURST, H.; BLACK, R.; SIMAIKA, Y. **Long-term storage**: an experimental study. [S.l.]: Constable, 1965.

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

JAEGER, H.; HAAS, H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. **Science**, [S.l.], v.304, n.5667, p.78, 2004.

JAEGER, H.; LUKOSEVICIUS, M.; POPOVICI, D.; SIEWERT, U. Optimization and applications of echo state networks with leaky-integrator neurons. **Neural Networks**, [S.l.], v.20, n.3, p.335–352, 2007.

JAIN, A.; NANDAKUMAR, K.; ROSS, A. Score normalization in multimodal biometric systems. **Pattern recognition**, [S.l.], v.38, n.12, p.2270–2285, 2005.

KALMAN, R. A new approach to linear filtering and prediction problems. **Journal of basic Engineering**, [S.l.], v.82, n.Series D, p.35–45, 1960.

KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological cybernetics**, [S.l.], v.43, n.1, p.59–69, 1982.

KOLMOGOROV, A. Three approaches to the quantitative definition of information. **Problems of information transmission**, [S.l.], v.1, n.1, p.1–7, 1965.

Kugiumtzis, D.; Tsimpiris, A. Measures of Analysis of Time Series (MATS): a matlab toolkit for computation of multiple measures on time series data bases. **ArXiv e-prints**, [S.l.], Feb. 2010.

LECUN, Y.; BOTTOU, L.; ORR, G.; MÜLLER, K. Efficient backprop. **Neural networks: Tricks of the trade**, [S.l.], p.546–546, 1998.

LEMPEL, A.; ZIV, J. On the complexity of finite sequences. **Information Theory, IEEE Transactions on**, [S.l.], v.22, n.1, p.75–81, 1976.

LIN, T.; HORNE, B.; TINO, P.; GILES, C. Learning long-term dependencies in NARX recurrent neural networks. **Neural Networks, IEEE Transactions on**, [S.l.], v.7, n.6, p.1329–1338, 1996.

LUKOŠEVIČIUS, M.; JAEGER, H. Reservoir computing approaches to recurrent neural network training. **Computer Science Review**, [S.l.], v.3, n.3, p.127–149, August 2009.

MALKIEL, B.; MCCUE, K. **A random walk down Wall Street**. [S.l.]: Norton New York, 1985.

NATSCHLÄGER, T.; MAASS, W.; MARKRAM, H. The "Liquid Computer": a novel strategy for real-time computing on time series. **Special Issue on Foundations of Information Processing of TELEMATIK**, [S.l.], v.8, n.1, p.39–43, 2002.

PENG, C.; BULDYREV, S.; HAVLIN, S.; SIMONS, M.; STANLEY, H.; GOLDBERGER, A. Mosaic organization of DNA nucleotides. **Physical Review E**, [S.l.], v.49, n.2, p.1685, 1994.

PINTO, R.; ENGEL, P.; HEINEN, M. Echo State Incremental Gaussian Mixture Network for Spatio-Temporal Pattern Processing. In: IX ENIA - BRAZILIAN MEETING ON ARTIFICIAL INTELLIGENCE, 2011, Natal (RN). **Proceedings...** [S.l.: s.n.], 2011.

PINTO, R.; ENGEL, P.; HEINEN, M. Recursive Incremental Gaussian Mixture Network for Spatio-Temporal Pattern Processing. In: X CBIC - BRAZILIAN CONFERENCE ON COMPUTATIONAL INTELLIGENCE, 2011, Fortaleza (CE). **Proceedings...** [S.l.: s.n.], 2011.

RABINER, L.; JUANG, B. An introduction to hidden Markov models. **ASSP Magazine, IEEE**, [S.l.], v.3, n.1, p.4–16, 1986.

RUMELHART, D.; MCCLELLAND, J. **Parallel distributed processing**. [S.l.]: MIT Pr., 1986.

SALMERÓN, M.; ORTEGA, J.; PUNTONET, C.; PRIETO, A.; ROJAS, I. SSA, SVD, QR-cp, and RBF model reduction. **Artificial Neural Networks-ICANN 2002**, [S.l.], p.796–796, 2002.

SCALES, L. **Introduction to non-linear optimization**. [S.l.]: Springer-Verlag New York, Inc., 1985.

SITTE, R.; SITTE, J. Neural networks approach to the random walk dilemma of financial time series. **Applied Intelligence**, [S.l.], v.16, n.3, p.163–171, 2002.

STANLEY, K.; MIIKKULAINEN, R. Evolving neural networks through augmenting topologies. **Evolutionary Computation**, [S.l.], v.10, p.2002, 2001.

STEIL, J. J. **Backpropagation-Decorrelation**: online recurrent learning with o(n) complexity,. 2004.

STRICKERT, M.; HAMMER, B. Merge SOM for temporal data. **Neurocomputing**, [S.l.], v.64, p.39–71, 2005.

STRICKERT, M.; HAMMER, B.; BLOHM, S. Unsupervised recursive sequence processing. **Neurocomputing**, [S.l.], v.63, p.69–97, 2005.

VERSTRAETEN, D.; SCHRAUWEN, B.; STROOBANDT, D. **Reservoir-based techniques for speech recognition**. 2006. 1050–1053p.

VOEGTLIN, T. Recursive self-organizing maps. **Neural Networks**, [S.l.], v.15, n.8-9, p.979–991, 2002.

WEIGEND, A.; GERSHENFELD, N. Time series prediction: forecasting the future and understanding the past. In: NATO ADVANCED RESEARCH WORKSHOP ON COMPARATIVE TIME SERIES ANALYSIS, 1994. **Proceedings...** [S.l.: s.n.], 1994. v.1.

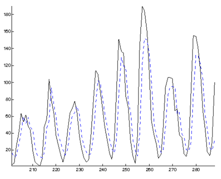
WERBOS, P. **Beyond regression**: new tools for regression and analysis in the behavioral sciences. 1974. Tese (Doutorado em Ciência da Computação) — PhD thesis, Harvard University, Division of Engineering and Applied Physics.

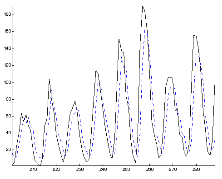
WILCOXON, F. Individual comparisons by ranking methods. **Biometrics Bulletin**, [S.l.], v.1, n.6, p.80–83, 1945.

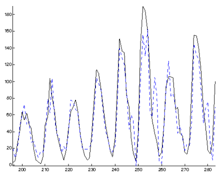
WILLIAMS, R. Training recurrent networks using the extended Kalman filter. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS - IJCNN, 1992. **Proceedings...** [S.l.: s.n.], 1992. v.4, p.241–246.

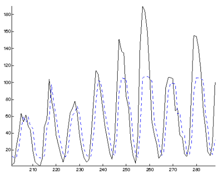
WILLIAMS, R.; ZIPSER, D. A learning algorithm for continually running fully recurrent neural networks. **Neural computation**, [S.l.], v.1, n.2, p.270–280, 1989.

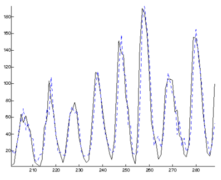
APPENDIX A OTHER FIGURES

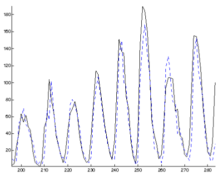

(a) Elman

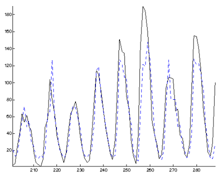

(b) ESN

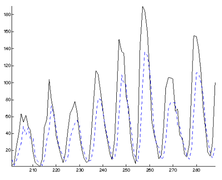

(c) TDNN

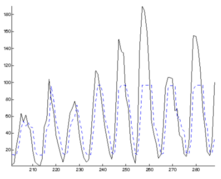

(d) IGMN

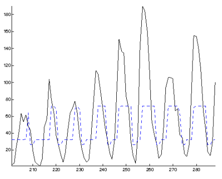

(e) ESIGMN

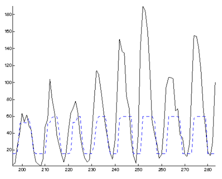

(f) TDIGMN

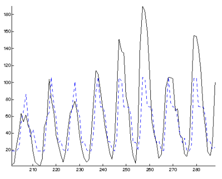

(g) MIGMN

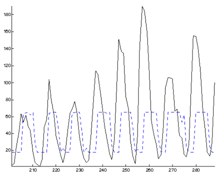

(h) RecIGMN

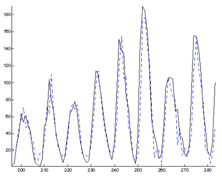

(i) IGMNn

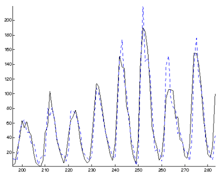

(j) ESIGMNn

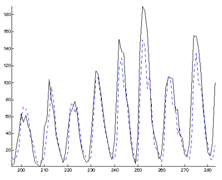

(k) TDIGMNn

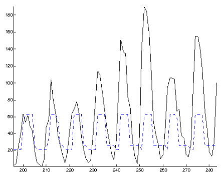

(l) MIGMNn

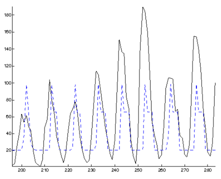

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

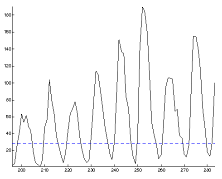
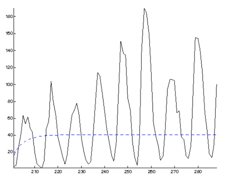
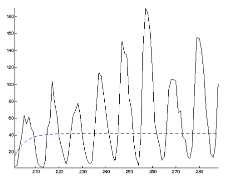
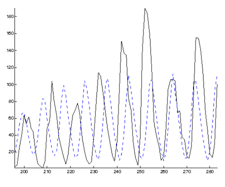
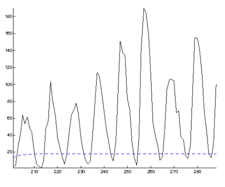

(s) TDRecIGMNn

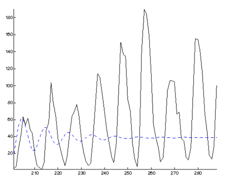
Figure A.1: One-step prediction example outputs of best and worst algorithms on the yearly mean sunspot numbers time-series (test set only). Original series in black, estimates in blue.

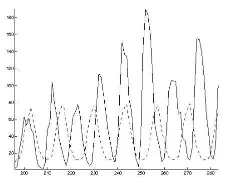

(a) Elman

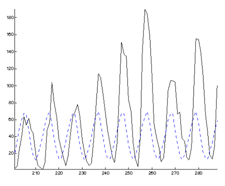

(b) ESN

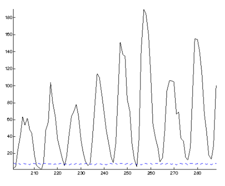

(c) TDNN

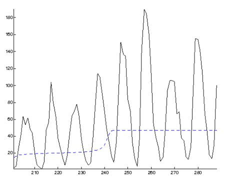

(d) IGMN

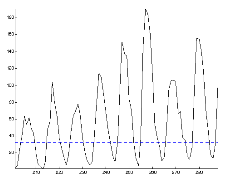

(e) ESIGMN

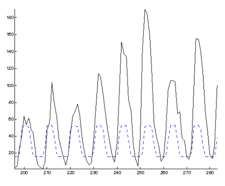

(f) TDIGMN

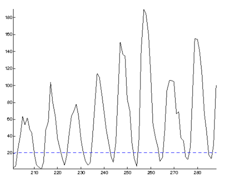

(g) MIGMN

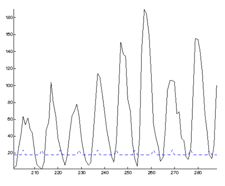

(h) RecIGMN

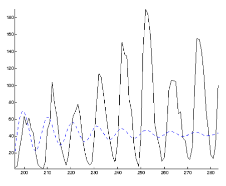

(i) IGMNn

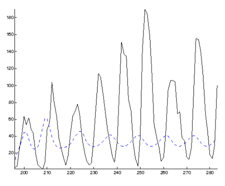

(j) ESIGMNn

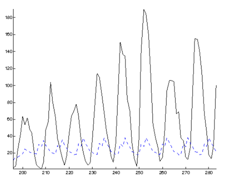

(k) TDIGMNn

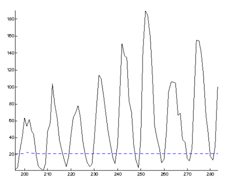

(l) MIGMNn

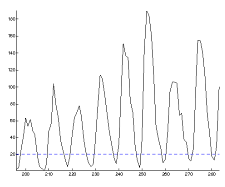

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

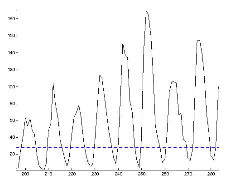

(s) TDRecIGMNn

Figure A.2: Long-term prediction example outputs of best and worst algorithms in the yearly mean sunspot numbers time-series (test set only). Original series in black, estimates in blue.

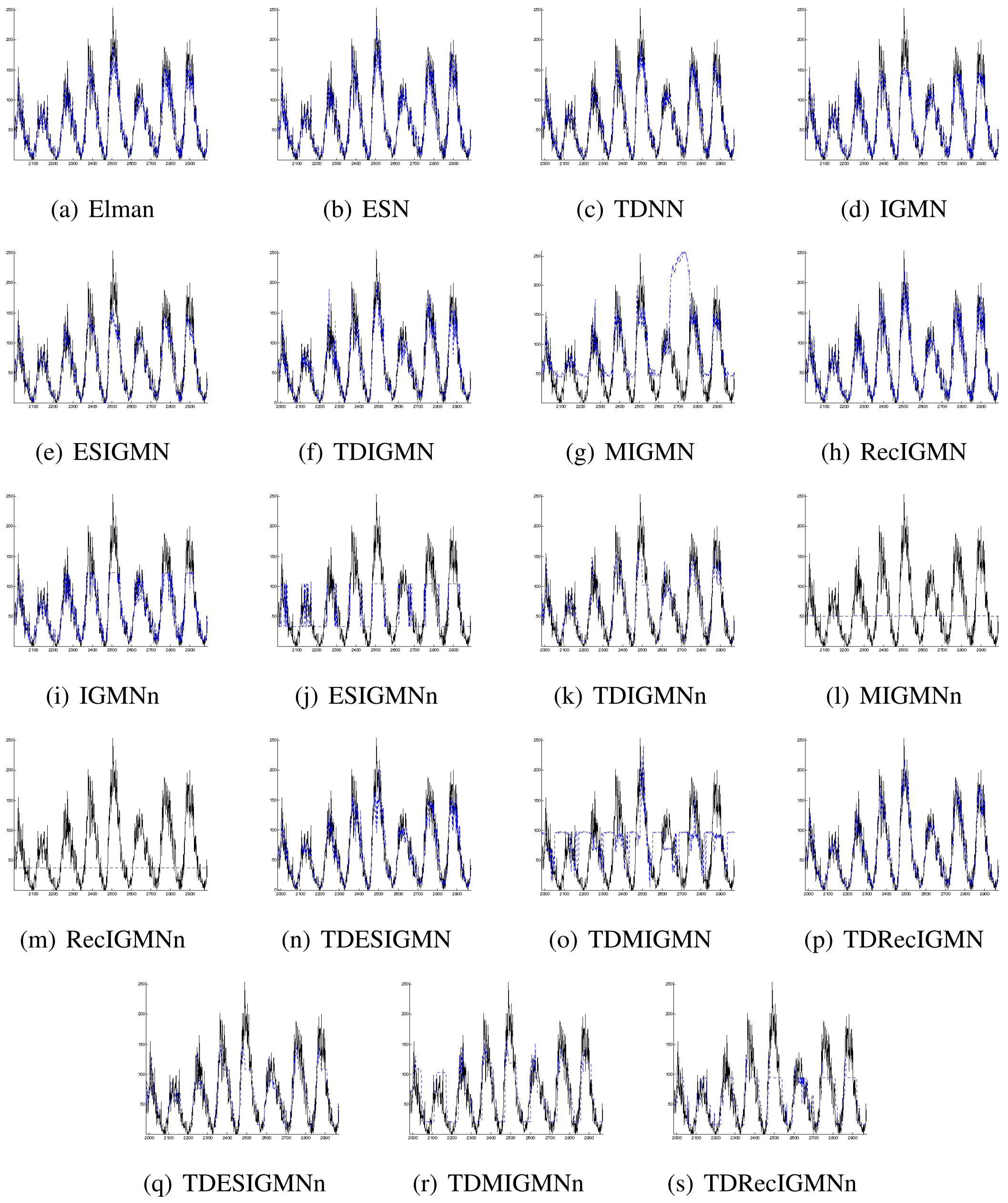
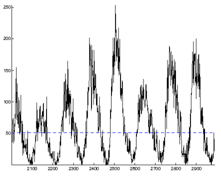
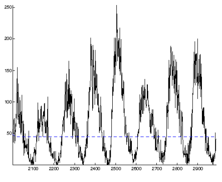
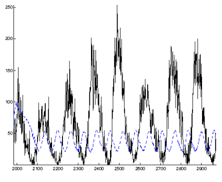
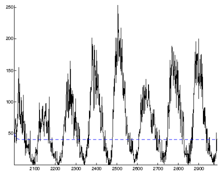


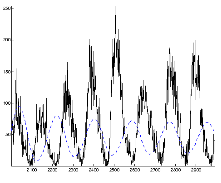
Figure A.3: One-step prediction example outputs of best and worst algorithms in the monthly mean sunspot numbers time-series (test set only). Original series in black, estimates in blue.

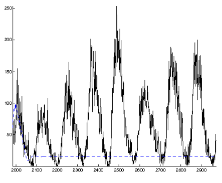

(a) Elman

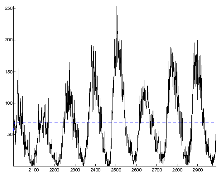

(b) ESN

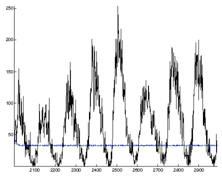

(c) TDNN

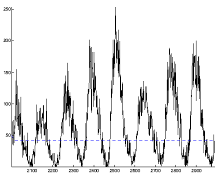

(d) IGMN

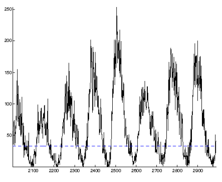

(e) ESIGMN

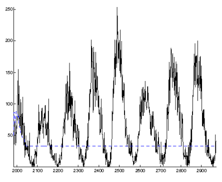

(f) TDIGMN

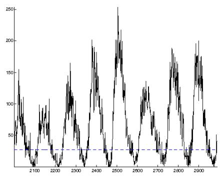

(g) MIGMN

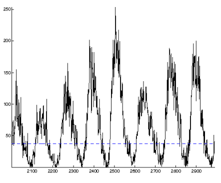

(h) RecIGMN

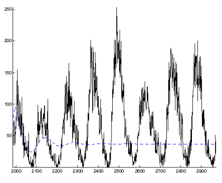

(i) IGMNn

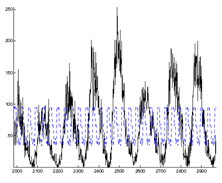

(j) ESIGMNn

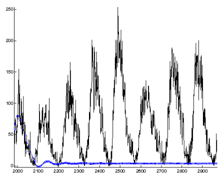

(k) TDIGMNn

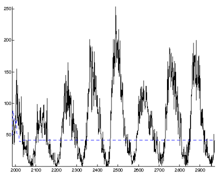

(l) MIGMNn

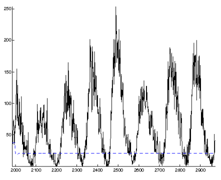

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

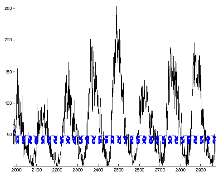
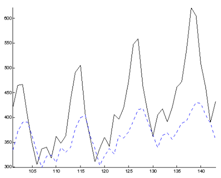
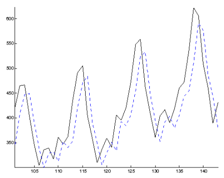
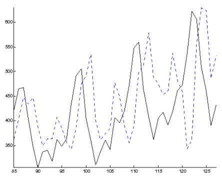
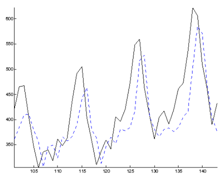

(s) TDRecIGMNn

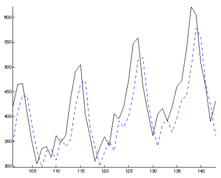
Figure A.4: Long-term prediction example outputs of best and worst algorithms in the monthly mean sunspot numbers time-series (test set only). Original series in black, estimates in blue.

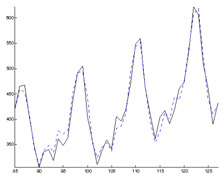

(a) Elman

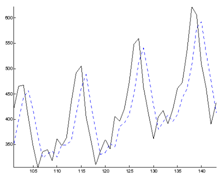

(b) ESN

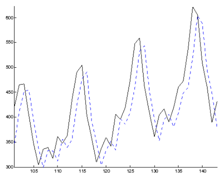

(c) TDNN

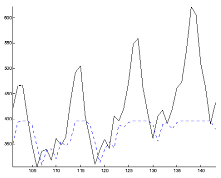

(d) IGMN

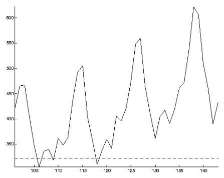

(e) ESIGMN

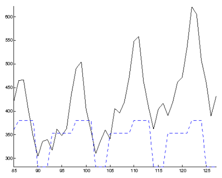

(f) TDIGMN

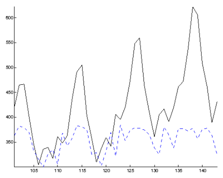

(g) MIGMN

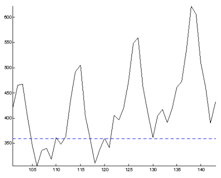

(h) RecIGMN

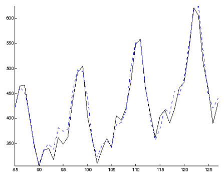

(i) IGMNn

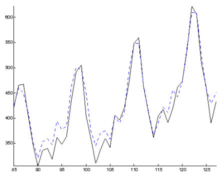

(j) ESIGMNn

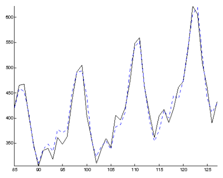

(k) TDIGMNn

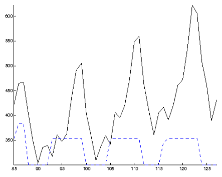

(l) MIGMNn

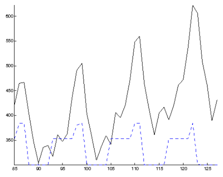

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

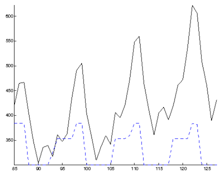
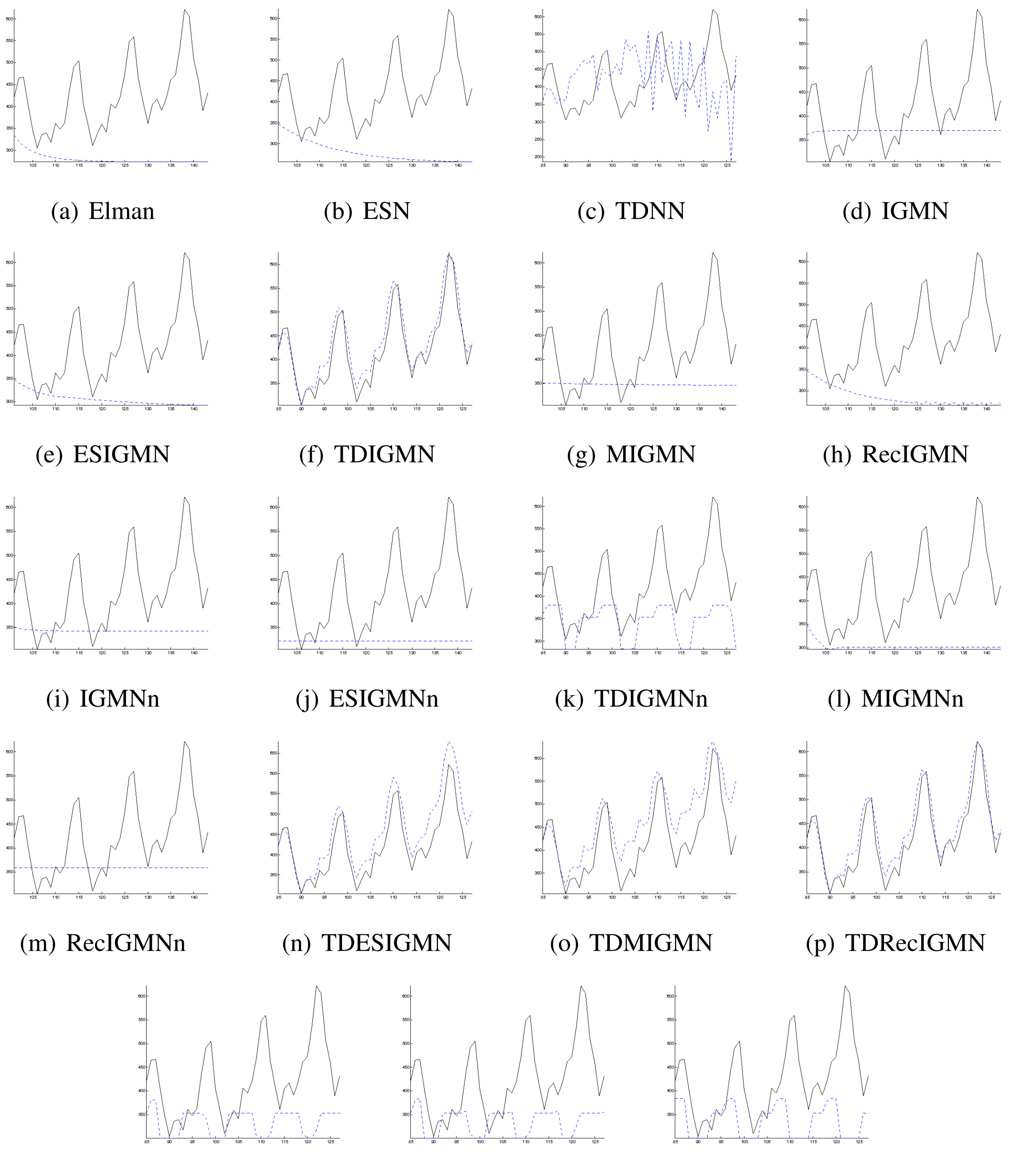

(s) TDRecIGMNn

Figure A.5: One-step prediction example outputs of best and worst algorithms in the airline passengers time-series (test set only). Original series in black, estimates in blue.



(a) Elman (b) ESN (c) TDNN (d) IGMN

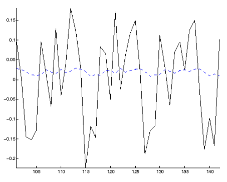
(e) ESIGMN (f) TDIGMN (g) MIGMN (h) RecIGMN

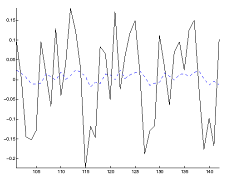
(i) IGMNn (j) ESIGMNn (k) TDIGMNn (l) MIGMNn

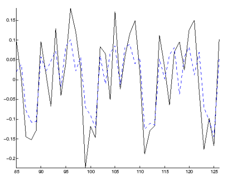
(m) RecIGMNn (n) TDESIGMN (o) TDMIGMN (p) TDRecIGMN

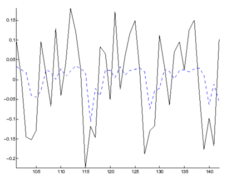
(q) TDESIGMNn (r) TDMIGMNn (s) TDRecIGMNn

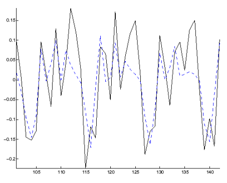
Figure A.6: Long-term prediction example outputs of best and worst algorithms in the airline passengers time-series. Original series in black, estimates in blue.

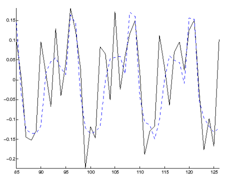

(a) Elman

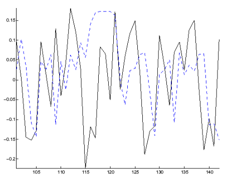

(b) ESN

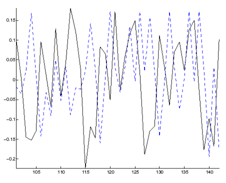

(c) TDNN

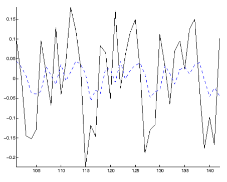

(d) IGMN

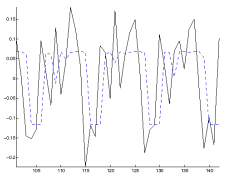

(e) ESIGMN

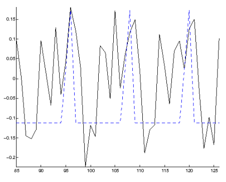

(f) TDIGMN

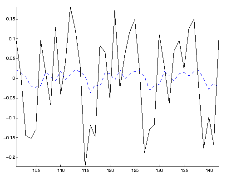

(g) MIGMN

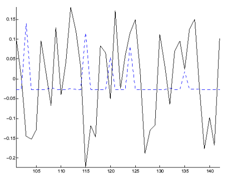

(h) RecIGMN

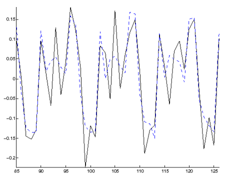

(i) IGMNn

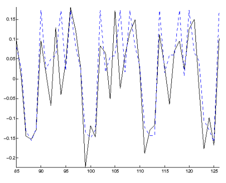

(j) ESIGMNn

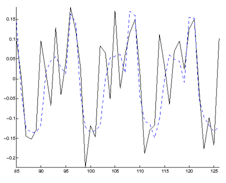

(k) TDIGMNn

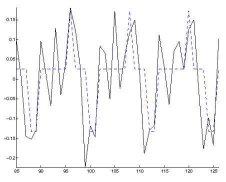

(l) MIGMNn

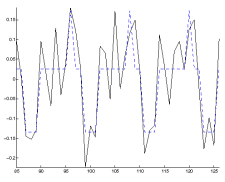

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

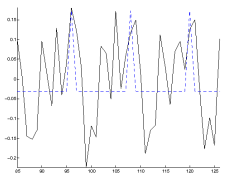
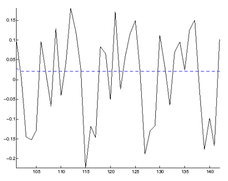
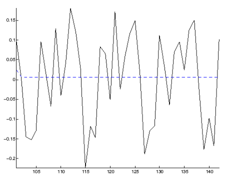
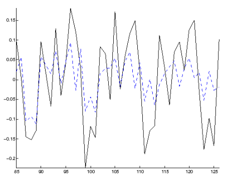
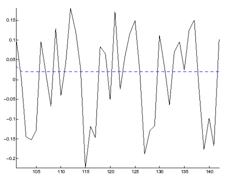

(s) TDRecIGMNn

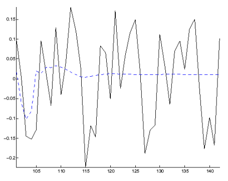
Figure A.7: One-step prediction example outputs of best and worst algorithms in the log-differentiated airline passengers time-series (test set only). Original series in black, estimates in blue.

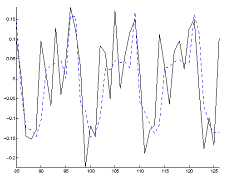

(a) Elman

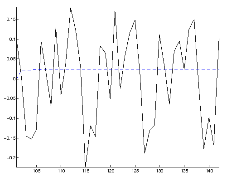

(b) ESN

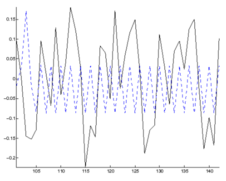

(c) TDNN

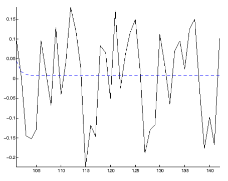

(d) IGMN

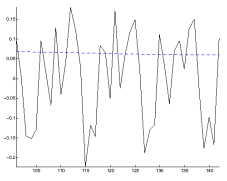

(e) ESIGMN

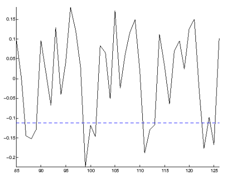

(f) TDIGMN

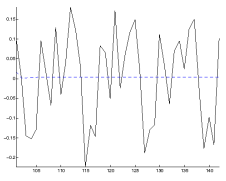

(g) MIGMN

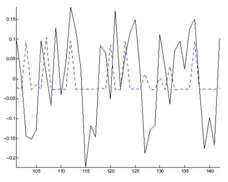

(h) RecIGMN

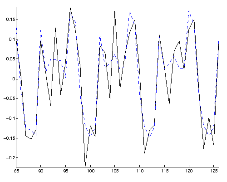

(i) IGMNn

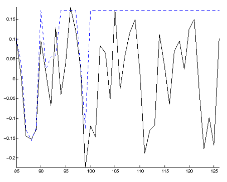

(j) ESIGMNn

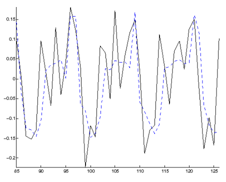

(k) TDIGMNn

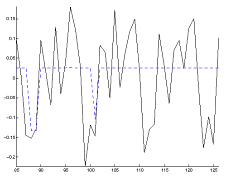

(l) MIGMNn

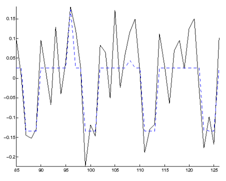

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

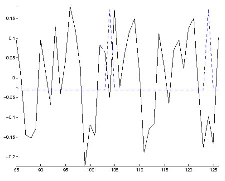
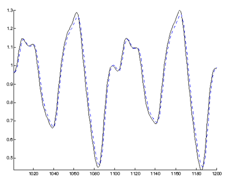
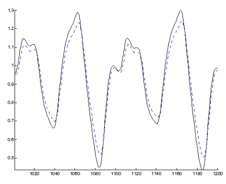
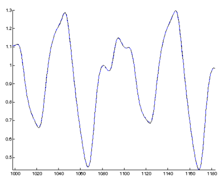
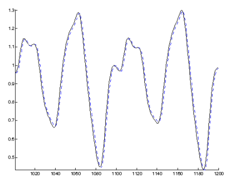

(s) TDRecIGMNn

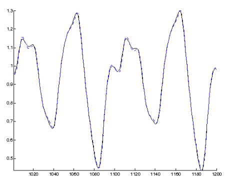
Figure A.8: Long-term prediction example outputs of best and worst algorithms in the log-differentiated airline passengers time-series. Original series in black, estimates in blue.

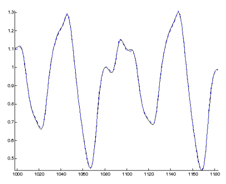

(a) Elman

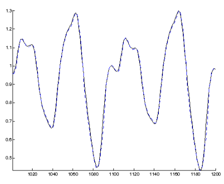

(b) ESN

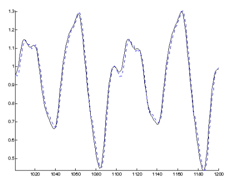

(c) TDNN

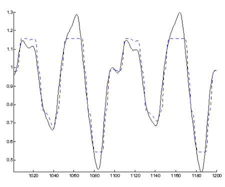

(d) IGMN

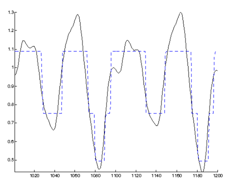

(e) ESIGMN

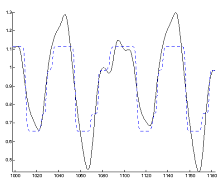

(f) TDIGMN

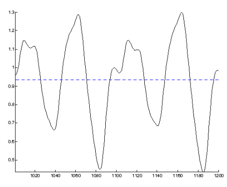

(g) MIGMN

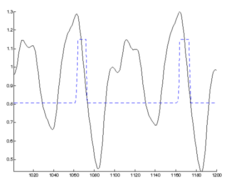

(h) RecIGMN

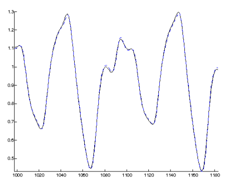

(i) IGMNn

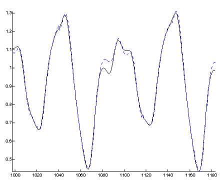

(j) ESIGMNn

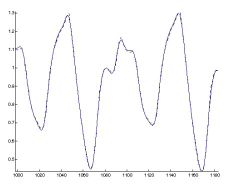

(k) TDIGMNn

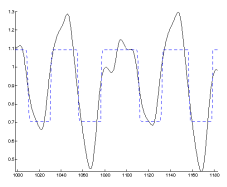

(l) MIGMNn

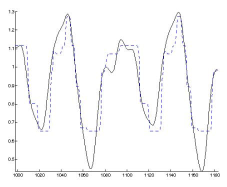

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

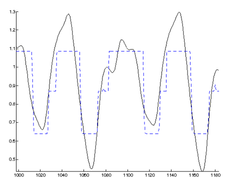
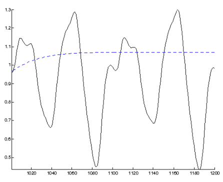
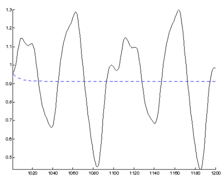
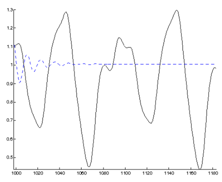
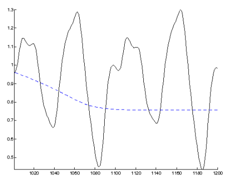

(s) TDRecIGMNn

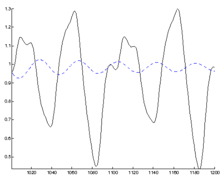
Figure A.9: One-step prediction example outputs of best and worst algorithms in the Mackey-Glass (τ = 17) time-series (test set only). Original series in black, estimates in blue.

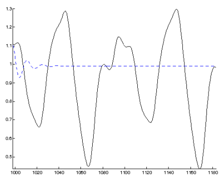

(a) Elman

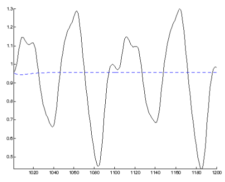

(b) ESN

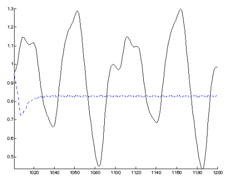

(c) TDNN

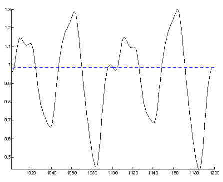

(d) IGMN

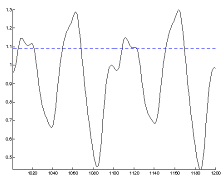

(e) ESIGMN

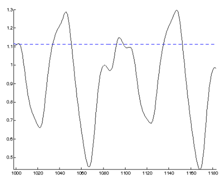

(f) TDIGMN

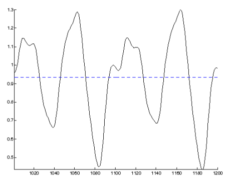

(g) MIGMN

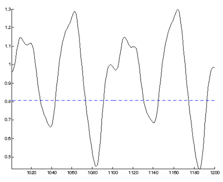

(h) RecIGMN

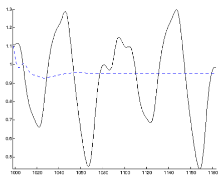

(i) IGMNn

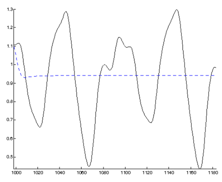

(j) ESIGMNn

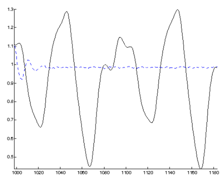

(k) TDIGMNn

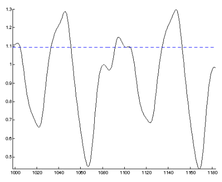

(l) MIGMNn

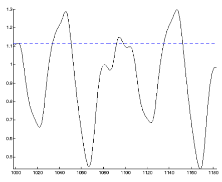

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

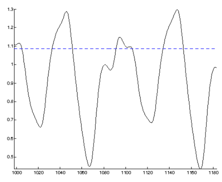
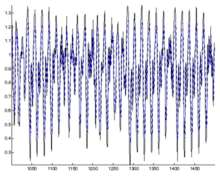
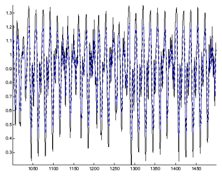
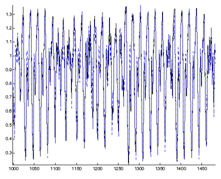
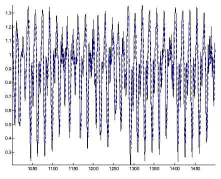

(s) TDRecIGMNn

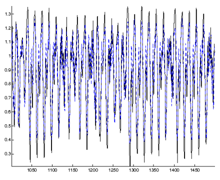
Figure A.10: Long-term prediction example outputs of best and worst algorithms in the Mackey-Glass (τ = 17) time-series. Original series in black, estimates in blue.

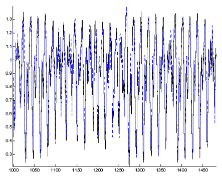

(a) Elman

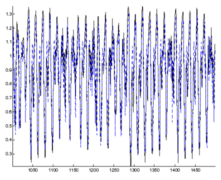

(b) ESN

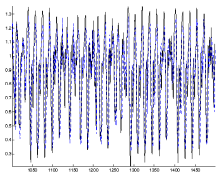

(c) TDNN

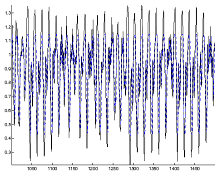

(d) IGMN

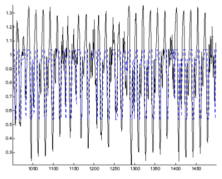

(e) ESIGMN

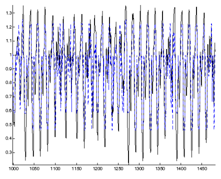

(f) TDIGMN

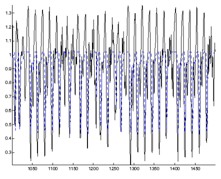

(g) MIGMN

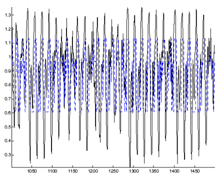

(h) RecIGMN

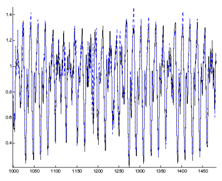

(i) IGMNn

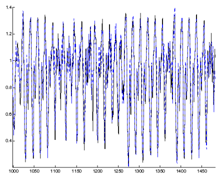

(j) ESIGMNn

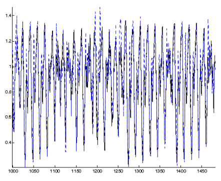

(k) TDIGMNn

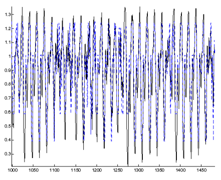

(l) MIGMNn

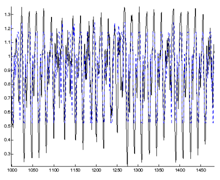

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

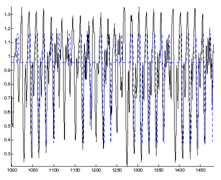
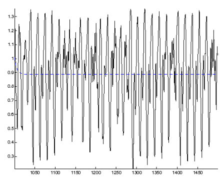
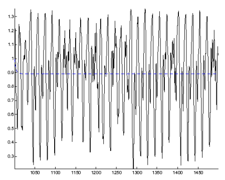
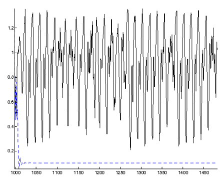
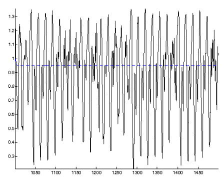

(s) TDRecIGMNn

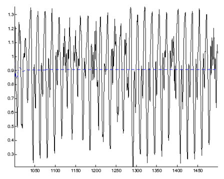
Figure A.11: One-step prediction example outputs of best and worst algorithms in the Mackey-Glass (τ = 30) time-series (test set only). Original series in black, estimates in blue.

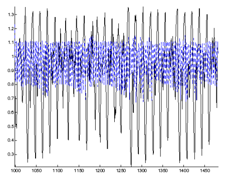

(a) Elman

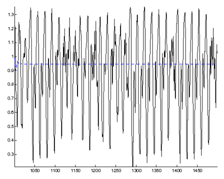

(b) ESN

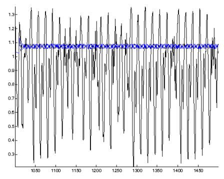

(c) TDNN

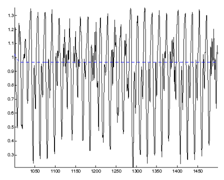

(d) IGMN

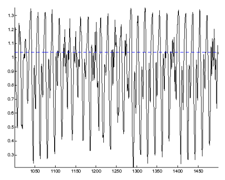

(e) ESIGMN

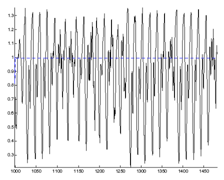

(f) TDIGMN

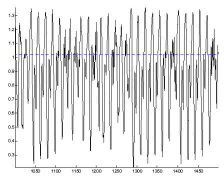

(g) MIGMN

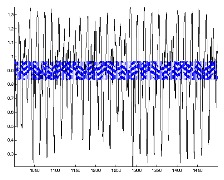

(h) RecIGMN

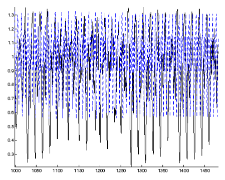

(i) IGMNn

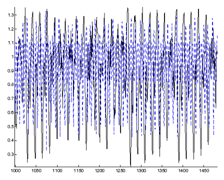

(j) ESIGMNn

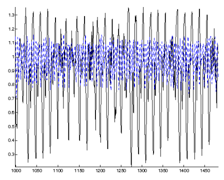

(k) TDIGMNn

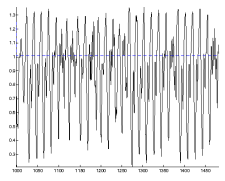

(l) MIGMNn

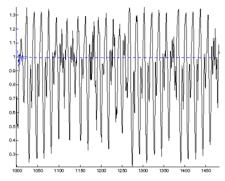

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

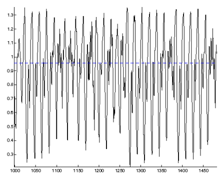
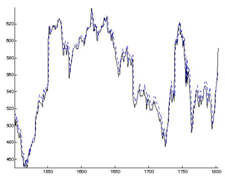
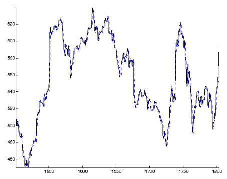
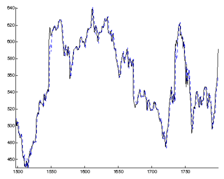
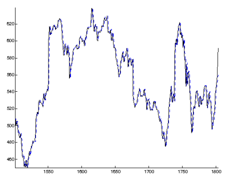

(s) TDRecIGMNn

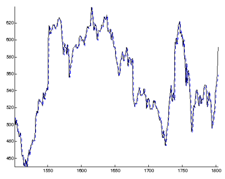
Figure A.12: Long-term prediction example outputs of best and worst algorithms in the Mackey-Glass (τ = 30) time-series. Original series in black, estimates in blue.

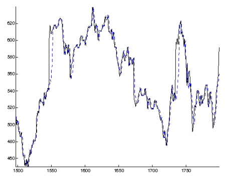

(a) Elman

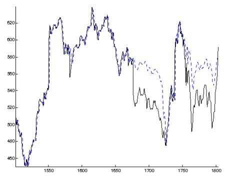

(b) ESN

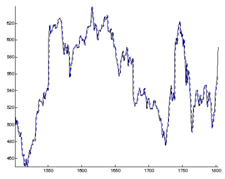

(c) TDNN

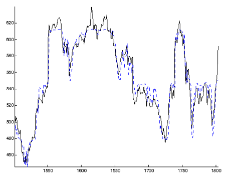

(d) IGMN

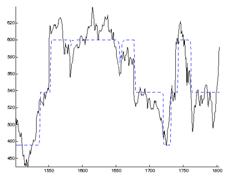

(e) ESIGMN

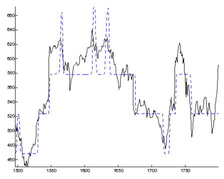

(f) TDIGMN

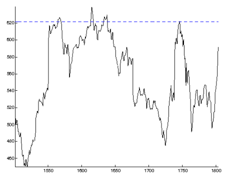

(g) MIGMN

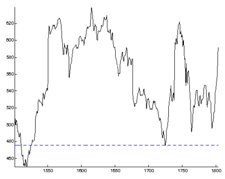

(h) RecIGMN

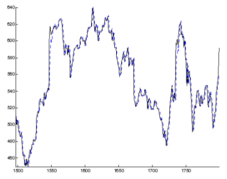

(i) IGMNn

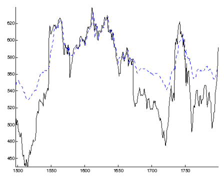

(j) ESIGMNn

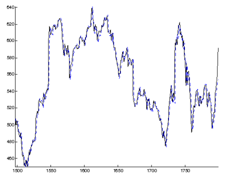

(k) TDIGMNn

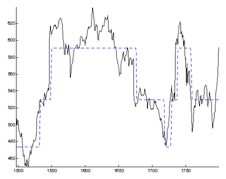

(l) MIGMNn

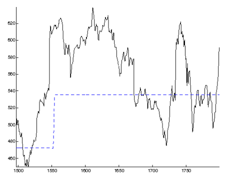

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

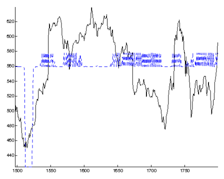
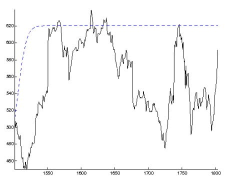
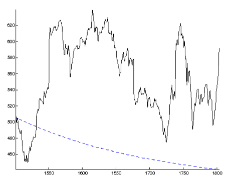
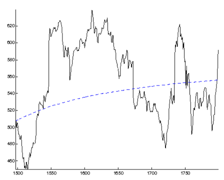
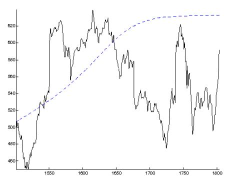

(s) TDRecIGMNn

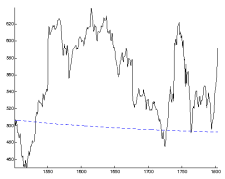
Figure A.13: One-step prediction example outputs of best and worst algorithms in the Google stock time-series (test set only). Original series in black, estimates in blue.

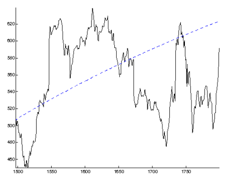

(a) Elman

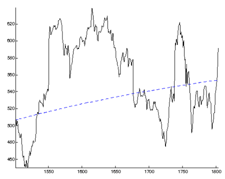

(b) ESN

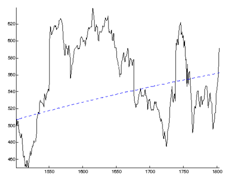

(c) TDNN

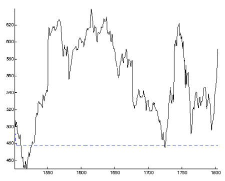

(d) IGMN

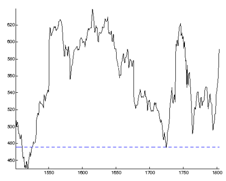

(e) ESIGMN

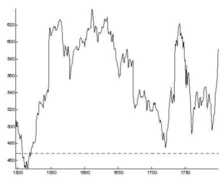

(f) TDIGMN

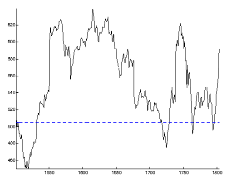

(g) MIGMN

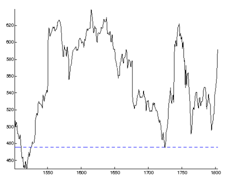

(h) RecIGMN

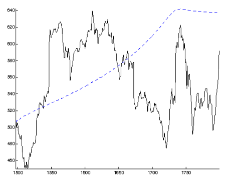

(i) IGMNn

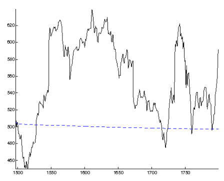

(j) ESIGMNn

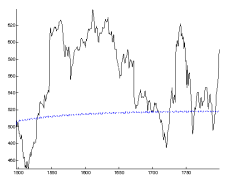

(k) TDIGMNn

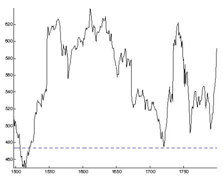

(l) MIGMNn

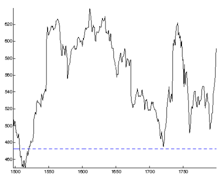

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

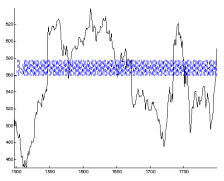
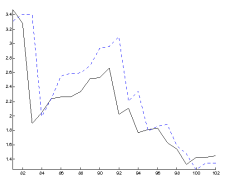
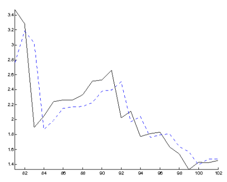
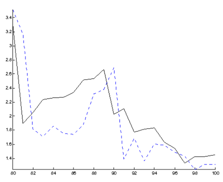
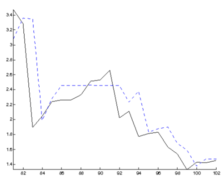

(s) TDRecIGMNn

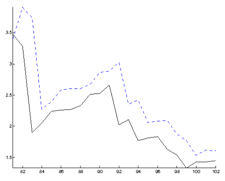
Figure A.14: Long-term prediction example outputs of best and worst algorithms in the Google stock time-series. Original series in black, estimates in blue.

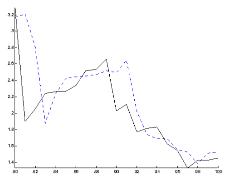

(a) Elman

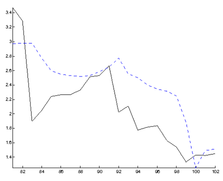

(b) ESN

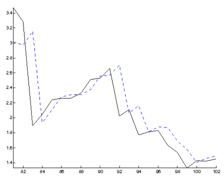

(c) TDNN

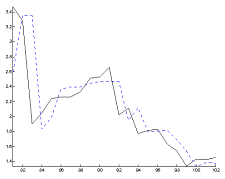

(d) IGMN

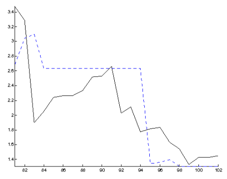

(e) ESIGMN

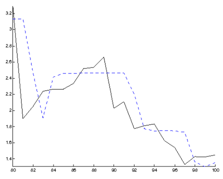

(f) TDIGMN

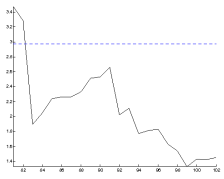

(g) MIGMN

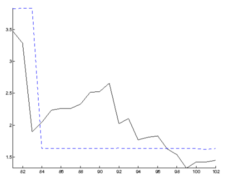

(h) RecIGMN

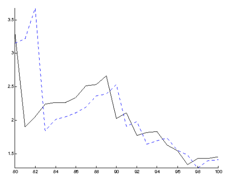

(i) IGMNn

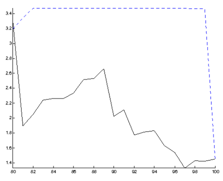

(j) ESIGMNn

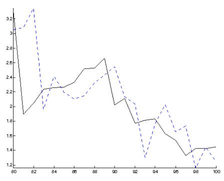

(k) TDIGMNn

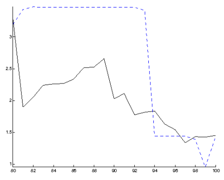

(l) MIGMNn

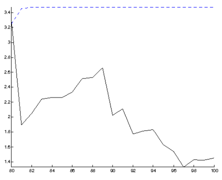

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

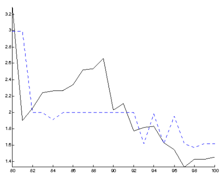
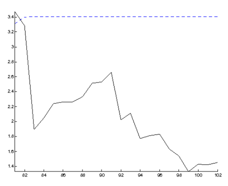
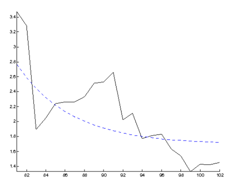
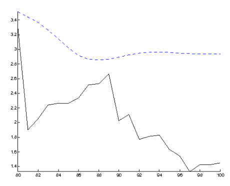
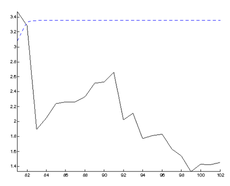

(s) TDRecIGMNn

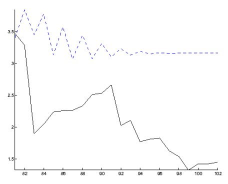
Figure A.15: One-step prediction example outputs of best and worst algorithms in the U.S. interest rate time-series (test set only). Original series in black, estimates in blue.

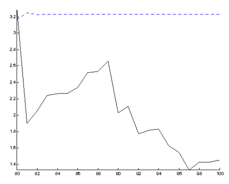

(a) Elman

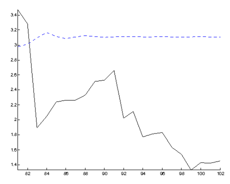

(b) ESN

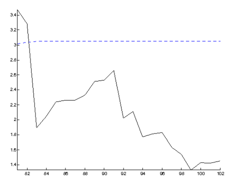

(c) TDNN

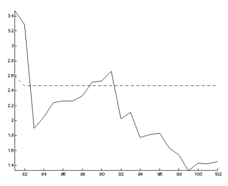

(d) IGMN

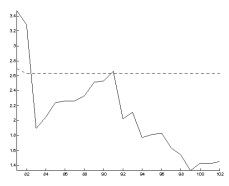

(e) ESIGMN

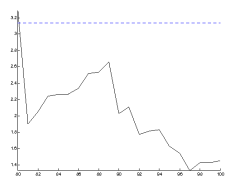

(f) TDIGMN

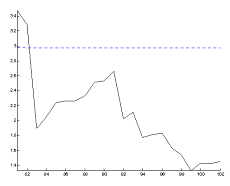

(g) MIGMN

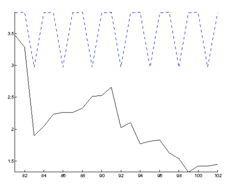

(h) RecIGMN

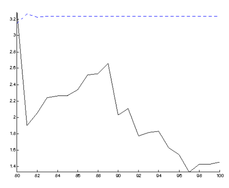

(i) IGMNn

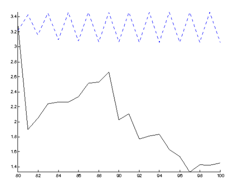

(j) ESIGMNn

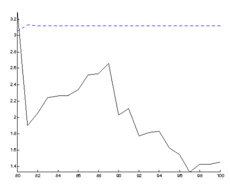

(k) TDIGMNn

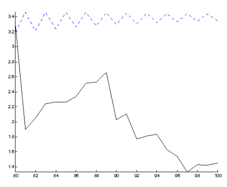

(l) MIGMNn

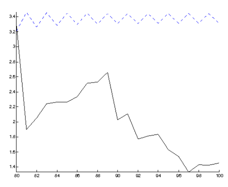

(m) RecIGMNn


(n) TDESIGMN


(o) TDMIGMN


(p) TDRecIGMN


(q) TDESIGMNn


(r) TDMIGMNn

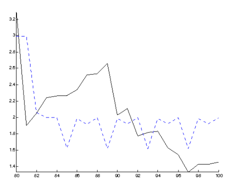

(s) TDRecIGMNn

Figure A.16: Long-term prediction example outputs of best and worst algorithms in the U.S. interest rate time-series. Original series in black, estimates in blue.

APPENDIX B OTHER TABLES

For all the following tables, an arrow pointing upwards means the column algorithm was statistically better than the row algorithm. An arrow pointing leftwards means the row algorithm was statistically better than the column algorithm. A hyphen means both row and column algorithms are statistically similar.

	Trivial	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Trivial	-	↑	↑	↑	←	↑	↑	↑	←	←	←	←	←	←	↑	↑	↑	←	←	←
Elman	←	-	↑	↑	←	↑	↑	↑	←	←	←	←	←	←	↑	↑	↑	←	←	←
ESN	←	←	-	↑	←	↑	↑	↑	←	←	←	←	←	←	↑	↑	↑	←	←	←
TDNN	←	←	←	-	←	↑	↑	-	←	←	←	←	←	←	↑	↑	-	←	←	←
IGMN	↑	↑	↑	↑	-	↑	↑	↑	←	←	←	←	←	←	↑	↑	↑	←	←	←
ESIGMN	←	←	←	←	←	-	-	←	←	←	←	←	←	←	-	↑	←	←	←	←
TDIGMN	←	←	←	←	←	-	-	←	←	←	←	←	←	←	↑	↑	←	←	←	←
MIGMN	←	←	←	-	←	↑	↑	-	←	←	←	←	←	←	↑	↑	←	←	←	←
RecIGMN	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	←	←	↑	←	↑	↑	↑	←	←	←
IGMNn	↑	↑	↑	↑	↑	↑	↑	↑	←	-	←	←	←	←	↑	↑	↑	←	←	←
ESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	←	↑	←	↑	↑	↑	←	←	←
TDIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	←	↑	↑	↑	-	←	←
MIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	←	←	-	←	↑	↑	↑	←	←	←
RecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	←
TDESIGMN	←	←	←	←	←	-	←	←	←	←	←	←	←	←	-	↑	←	←	←	←
TDMIGMN	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	-	←	←	←	←
TDRecIGMN	←	←	←	-	←	↑	↑	↑	←	←	←	←	←	←	↑	↑	-	←	←	←
TDESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	←	↑	↑	↑	-	-	←
TDMIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	-	-	←
TDRecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-

Table B.1: Statistical comparison of obtained one-step prediction errors for all algorithms in the yearly sunspot numbers experiment.

	Mean	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Mean	-	←	-	←	←	↑	←	↑	←	←	←	↑	←	←	-	←	←	←	←	←
Elman	↑	-	↑	↑	-	↑	↑	↑	←	↑	↑	↑	-	-	↑	↑	↑	↑	-	↑
ESN	-	←	-	←	←	↑	←	↑	←	←	←	↑	←	←	-	←	←	←	←	←
TDNN	↑	←	↑	-	←	↑	-	↑	←	↑	-	↑	-	←	↑	-	-	-	-	-
IGMN	↑	-	↑	↑	-	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
ESIGMN	←	←	←	←	←	-	←	↑	←	←	←	↑	←	←	-	←	←	←	←	←
TDIGMN	↑	←	↑	-	←	↑	-	↑	←	↑	↑	↑	←	←	↑	↑	↑	↑	←	↑
MIGMN	←	←	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←
RecIGMN	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
IGMNn	↑	←	↑	←	←	↑	←	↑	←	-	←	↑	←	←	-	←	←	←	←	←
ESIGMNn	↑	←	↑	-	←	↑	←	↑	←	↑	-	↑	←	←	↑	-	←	-	←	-
TDIGMNn	←	←	←	←	←	←	←	↑	←	←	←	-	←	←	←	←	←	←	←	←
MIGMNn	↑	-	↑	-	←	↑	↑	↑	←	↑	↑	↑	-	←	↑	↑	↑	↑	←	↑
RecIGMNn	↑	-	↑	↑	←	↑	↑	↑	←	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑
TDESIGMN	-	←	-	←	←	-	←	↑	←	-	←	↑	←	←	-	←	←	←	←	←
TDMIGMN	↑	←	↑	-	←	↑	←	↑	←	↑	-	↑	←	←	↑	-	←	←	←	←
TDRecIGMN	↑	←	↑	-	←	↑	←	↑	←	↑	↑	↑	←	←	↑	↑	-	↑	←	↑
TDESIGMNn	↑	←	↑	-	←	↑	←	↑	←	↑	-	↑	←	←	↑	↑	←	-	←	-
TDMIGMNn	↑	-	↑	-	←	↑	↑	↑	←	↑	↑	↑	↑	←	↑	↑	↑	↑	-	↑
TDRecIGMNn	↑	←	↑	-	←	↑	←	↑	←	↑	-	↑	←	←	↑	↑	←	-	←	-

Table B.2: Statistical comparison of obtained long-term prediction errors for all algorithms in the yearly sunspot numbers experiment.

	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Elman	-	↑	-	←	←	←	←	←	↑	-	-	←	-	←	←	←	-	-	-
ESN	←	-	-	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←
TDNN	-	-	-	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←
IGMN	↑	↑	↑	-	↑	↑	←	↑	↑	↑	↑	←	↑	-	←	←	↑	↑	↑
ESIGMN	↑	↑	↑	←	-	←	←	←	↑	↑	↑	←	-	←	←	←	↑	↑	↑
TDIGMN	↑	↑	↑	←	↑	-	←	-	↑	↑	↑	←	-	-	←	←	↑	↑	↑
MIGMN	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	-	↑	-	←	↑	↑	↑	↑
RecIGMN	↑	↑	↑	←	↑	-	←	-	↑	↑	↑	←	-	-	←	←	↑	↑	↑
IGMNn	←	-	-	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←
ESIGMNn	-	↑	↑	←	←	←	←	←	↑	-	-	←	-	←	←	←	←	-	-
TDIGMNn	-	↑	↑	←	←	←	←	←	↑	-	-	←	←	←	←	←	←	←	-
MIGMNn	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	-	↑	↑	-	↑	↑	↑	↑
RecIGMNn	-	↑	↑	←	-	-	←	-	↑	-	↑	←	-	←	←	←	-	-	↑
TDESIGMN	↑	↑	↑	-	↑	-	-	-	↑	↑	↑	←	↑	-	←	-	↑	↑	↑
TDMIGMN	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	-	↑	↑	↑	↑
TDRecIGMN	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	←	↑	-	←	-	↑	↑	↑
TDESIGMNn	-	↑	↑	←	←	←	←	←	↑	↑	↑	←	-	←	←	←	-	↑	↑
TDMIGMNn	-	↑	↑	←	←	←	←	←	↑	-	↑	←	-	←	←	←	←	-	-
TDRecIGMNn	-	↑	↑	←	←	←	←	←	↑	-	-	←	←	←	←	←	←	-	-

Table B.3: Statistical comparison of obtained runtimes for all algorithms in the yearly sunspot numbers experiment.

	Trivial	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Trivial	-	←	↑	←	←	↑	←	←	↑	←	←	←	←	←	←	←	↑	←	←	←
Elman	↑	-	↑	←	←	↑	←	←	↑	←	←	←	←	←	←	←	↑	←	←	←
ESN	←	←	-	←	←	↑	←	←	-	←	←	←	←	←	←	←	↑	←	←	←
TDNN	↑	↑	↑	-	-	↑	←	←	↑	←	←	←	←	←	←	←	↑	←	←	←
IGMN	↑	↑	↑	-	-	↑	←	←	↑	←	←	←	←	←	←	←	↑	←	←	←
ESIGMN	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←	↑	←	←	←
TDIGMN	↑	↑	↑	↑	↑	↑	-	←	↑	←	←	←	←	←	←	←	↑	←	←	←
MIGMN	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
RecIGMN	←	←	-	←	←	↑	←	←	-	←	←	←	←	←	←	←	↑	←	←	←
IGMNn	↑	↑	↑	↑	↑	↑	↑	←	↑	-	←	←	←	←	-	←	↑	←	←	←
ESIGMNn	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	-	↑	←	←	↑	←	↑	-	-	←
TDIGMNn	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	←	-	←	←	↑	←	↑	-	←	←
MIGMNn	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	-	←	↑	←	↑	↑	↑	↑
RecIGMNn	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑
TDESIGMN	↑	↑	↑	↑	↑	↑	↑	←	↑	-	←	←	←	←	-	←	↑	←	←	←
TDMIGMN	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	←	↑	-	↑	↑	↑	↑
TDRecIGMN	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	-	←	←	←
TDESIGMNn	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	-	-	←	←	↑	←	↑	-	←	←
TDMIGMNn	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	-	↑	←	←	↑	←	↑	↑	-	←
TDRecIGMNn	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	←	←	↑	←	↑	↑	↑	-

Table B.4: Statistical comparison of obtained one-step prediction errors for all algorithms in the monthly sunspot numbers experiment.

	Mean	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Mean	-	←	-	-	←	↑	←	↑	←	←	-	←	←	←	-	←	←	-	←	↑
Elman	↑	-	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	-	↑
ESN	-	←	-	←	←	↑	←	↑	←	←	-	←	←	←	-	←	←	-	←	-
TDNN	-	←	↑	-	-	↑	←	↑	-	-	-	-	-	-	↑	-	←	↑	←	-
IGMN	↑	←	↑	-	-	↑	←	↑	←	↑	-	←	←	←	↑	↑	←	↑	←	↑
ESIGMN	←	←	←	←	←	-	←	↑	←	←	-	←	←	←	-	←	←	-	←	←
TDIGMN	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑
MIGMN	←	←	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←
RecIGMN	↑	←	↑	-	↑	↑	←	↑	-	↑	-	↑	←	↑	↑	↑	←	↑	←	↑
IGMNn	↑	←	↑	-	←	↑	←	↑	←	-	-	←	←	←	↑	↑	←	-	←	↑
ESIGMNn	-	←	-	-	-	-	←	↑	-	-	-	-	←	-	-	-	←	-	←	-
TDIGMNn	↑	←	↑	-	↑	↑	←	↑	←	↑	-	-	←	↑	↑	↑	←	↑	←	↑
MIGMNn	↑	←	↑	-	↑	↑	←	↑	↑	↑	↑	↑	-	↑	↑	↑	←	↑	←	↑
RecIGMNn	↑	←	↑	-	↑	↑	←	↑	←	↑	-	←	←	-	↑	↑	←	↑	←	↑
TDESIGMN	-	←	-	←	←	-	←	↑	←	←	-	←	←	←	-	←	←	-	←	-
TDMIGMN	↑	←	↑	-	←	↑	←	↑	←	←	-	←	←	←	↑	-	←	-	←	↑
TDRecIGMN	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑
TDESIGMNn	-	←	-	←	←	-	←	↑	←	-	-	←	←	←	-	-	←	-	←	-
TDMIGMNn	↑	-	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	-	↑
TDRecIGMNn	←	←	-	-	←	↑	←	↑	←	←	-	←	←	←	-	←	←	-	←	-

Table B.5: Statistical comparison of obtained long-term prediction errors for all algorithms in the monthly sunspot numbers experiment.

	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Elman	-	↑	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
ESN	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
TDNN	-	↑	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
IGMN	↑	↑	↑	-	↑	←	←	↑	↑	↑	↑	↑	↑	←	←	←	-	-	↑
ESIGMN	↑	↑	↑	←	-	←	←	↑	↑	↑	-	↑	↑	←	←	←	←	←	↑
TDIGMN	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
MIGMN	↑	↑	↑	↑	↑	←	-	↑	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
RecIGMN	↑	↑	↑	←	←	←	←	-	↑	↑	←	↑	↑	←	←	←	←	←	←
IGMNn	↑	↑	↑	←	←	←	←	←	-	←	←	-	↑	←	←	←	←	←	←
ESIGMNn	↑	↑	↑	←	←	←	←	←	↑	-	←	↑	↑	←	←	←	←	←	←
TDIGMNn	↑	↑	↑	←	-	←	←	↑	↑	↑	-	↑	↑	←	←	←	-	←	↑
MIGMNn	↑	↑	↑	←	←	←	←	←	-	←	←	-	↑	←	←	←	←	←	←
RecIGMNn	↑	↑	↑	←	←	←	←	←	←	←	←	←	-	←	←	←	←	←	←
TDESIGMN	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑
TDMIGMN	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	↑	←	-	←	↑	↑	↑
TDRecIGMN	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	↑	←	↑	-	↑	↑	↑
TDESIGMNn	↑	↑	↑	-	↑	←	←	↑	↑	↑	-	↑	↑	←	←	←	-	←	↑
TDMIGMNn	↑	↑	↑	-	↑	←	←	↑	↑	↑	↑	↑	↑	←	←	←	↑	-	↑
TDRecIGMNn	↑	↑	↑	←	←	←	←	↑	↑	↑	←	↑	↑	←	←	←	←	←	-

Table B.6: Statistical comparison of obtained runtimes for all algorithms in the monthly sunspot numbers experiment.

	Trivial	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Trivial	-	←	-	←	←	↑	↑	←	←	←	←	←	←	←	↑	↑	↑	←	←	←
Elman	↑	-	↑	-	-	↑	↑	-	↑	←	←	←	←	←	↑	↑	↑	←	←	←
ESN	-	←	-	←	←	↑	↑	←	←	←	←	←	←	←	↑	↑	↑	←	←	←
TDNN	↑	-	↑	-	↑	↑	↑	↑	↑	-	←	←	←	←	↑	↑	↑	←	←	←
IGMN	↑	-	↑	←	-	↑	↑	←	↑	←	←	←	←	←	↑	↑	↑	←	←	←
ESIGMN	←	←	←	←	←	-	↑	←	←	←	←	←	←	←	↑	↑	↑	←	←	←
TDIGMN	←	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←
MIGMN	↑	-	↑	←	↑	↑	↑	-	↑	←	←	←	←	←	↑	↑	↑	←	←	←
RecIGMN	↑	←	↑	←	←	↑	↑	←	-	←	←	←	←	←	↑	↑	↑	←	←	←
IGMNn	↑	↑	↑	-	↑	↑	↑	↑	↑	-	←	←	←	←	↑	↑	↑	←	←	←
ESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑
TDIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	-	↑	↑	↑	↑	↑	←	←	←
MIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	←	-	←	↑	↑	↑	←	←	←
RecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	←	↑	-	↑	↑	↑	←	←	←
TDESIGMN	←	←	←	←	←	←	↑	←	←	←	←	←	←	←	-	←	↑	←	←	←
TDMIGMN	←	←	←	←	←	←	↑	←	←	←	←	←	←	←	↑	-	↑	←	←	←
TDRecIGMN	←	←	←	←	←	←	↑	←	←	←	←	←	←	←	←	←	-	←	←	←
TDESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	-	↑	↑
TDMIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	←	-	↑
TDRecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	←	←	-

Table B.7: Comparison of obtained one-step prediction errors for all algorithms in the airline passengers experiment.

	Mean	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Mean	-	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
Elman	-	-	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
ESN	←	←	-	-	↑	↑	↑	↑	←	↑	-	↑	-	↑	↑	↑	↑	-	-	-
TDNN	←	-	-	-	↑	↑	↑	↑	-	↑	-	↑	-	↑	↑	↑	↑	↑	↑	↑
IGMN	←	←	←	←	-	←	↑	←	←	←	←	←	←	←	↑	↑	↑	←	←	←
ESIGMN	←	←	←	←	↑	-	↑	-	←	-	←	-	←	-	↑	↑	↑	←	←	←
TDIGMN	←	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←
MIGMN	←	←	←	←	↑	-	↑	-	←	←	←	←	←	↑	↑	↑	↑	←	←	←
RecIGMN	←	←	↑	-	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
IGMNn	←	←	←	←	↑	-	↑	↑	←	-	←	↑	←	↑	↑	↑	↑	←	←	←
ESIGMNn	←	←	-	-	↑	↑	↑	↑	←	↑	-	↑	-	↑	↑	↑	↑	-	-	-
TDIGMNn	←	←	←	←	↑	-	↑	↑	←	←	←	-	←	↑	↑	↑	↑	←	←	←
MIGMNn	←	←	-	-	↑	↑	↑	↑	←	↑	-	↑	-	↑	↑	↑	↑	↑	↑	↑
RecIGMNn	←	←	←	←	↑	-	↑	←	←	←	←	←	←	-	↑	↑	↑	←	←	←
TDESIGMN	←	←	←	←	←	←	↑	←	←	←	←	←	←	←	-	←	↑	←	←	←
TDMIGMN	←	←	←	←	←	←	↑	←	←	←	←	←	←	←	↑	-	↑	←	←	←
TDRecIGMN	←	←	←	←	←	←	↑	←	←	←	←	←	←	←	←	←	-	←	←	←
TDESIGMNn	←	←	-	←	↑	↑	↑	↑	←	↑	-	↑	←	↑	↑	↑	↑	-	↑	-
TDMIGMNn	←	←	-	←	↑	↑	↑	↑	←	↑	-	↑	←	↑	↑	↑	↑	←	-	←
TDRecIGMNn	←	←	-	←	↑	↑	↑	↑	←	↑	-	↑	←	↑	↑	↑	↑	-	↑	-

Table B.8: Comparison of obtained long-term prediction errors for all algorithms in the airline passengers experiment.

	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Elman	-	↑	-	↑	↑	↑	↑	-	↑	↑	↑	-	-	↑	↑	↑	↑	↑	↑
ESN	←	-	←	←	←	-	←	←	-	←	-	←	←	←	←	←	-	-	-
TDNN	-	↑	-	↑	↑	↑	↑	-	↑	↑	↑	-	-	↑	↑	↑	↑	↑	↑
IGMN	←	↑	←	-	←	↑	-	←	↑	-	↑	←	-	-	-	↑	↑	↑	↑
ESIGMN	←	↑	←	↑	-	↑	↑	-	↑	-	↑	-	-	↑	↑	↑	↑	↑	↑
TDIGMN	←	-	←	←	←	-	←	←	-	←	↑	←	←	←	←	←	↑	↑	↑
MIGMN	←	↑	←	-	←	↑	-	←	↑	-	↑	←	←	-	-	↑	↑	↑	↑
RecIGMN	-	↑	-	↑	-	↑	↑	-	↑	↑	↑	-	-	↑	↑	↑	↑	↑	↑
IGMNn	←	-	←	←	←	-	←	←	-	←	-	←	←	←	←	←	-	↑	↑
ESIGMNn	←	↑	←	-	-	↑	-	←	↑	-	↑	←	-	-	-	-	↑	↑	↑
TDIGMNn	←	-	←	←	←	←	←	←	-	←	-	←	←	←	←	←	-	-	-
MIGMNn	-	↑	-	↑	-	↑	↑	-	↑	↑	↑	-	-	↑	↑	↑	↑	↑	↑
RecIGMNn	-	↑	-	-	-	↑	↑	-	↑	-	↑	-	-	-	↑	↑	↑	↑	↑
TDESIGMN	←	↑	←	-	←	↑	-	←	↑	-	↑	←	-	-	-	↑	↑	↑	↑
TDMIGMN	←	↑	←	-	←	↑	-	←	↑	-	↑	←	←	-	-	↑	↑	↑	↑
TDRecIGMN	←	↑	←	←	←	↑	←	←	↑	-	↑	←	←	←	←	-	↑	↑	↑
TDESIGMNn	←	-	←	←	←	←	←	←	-	←	-	←	←	←	←	←	-	↑	-
TDMIGMNn	←	-	←	←	←	←	←	←	←	←	-	←	←	←	←	←	←	-	-
TDRecIGMNn	←	-	←	←	←	←	←	←	←	←	-	←	←	←	←	←	-	-	-

Table B.9: Comparison of obtained runtimes for all algorithms in the airline passengers experiment.

	Trivial	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Trivial	-	↑	↑	↑	↑	↑	↑	←	←	↑	↑	←	↑	↑	↑	↑	↑	↑	↑	↑
Elman	←	-	↑	↑	↑	↑	↑	←	←	↑	←	←	↑	←	↑	↑	↑	↑	↑	↑
ESN	←	←	-	↑	↑	↑	↑	←	←	↑	←	←	↑	←	↑	↑	↑	↑	↑	↑
TDNN	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	-	←
IGMN	←	←	←	↑	-	↑	↑	←	←	←	←	←	←	←	↑	↑	↑	↑	↑	←
ESIGMN	←	←	←	↑	←	-	↑	←	←	←	←	←	←	←	↑	↑	↑	↑	↑	←
TDIGMN	←	←	←	↑	←	←	-	←	←	←	←	←	←	←	↑	↑	←	↑	↑	←
MIGMN	↑	↑	↑	↑	↑	↑	↑	-	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
RecIGMN	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
IGMNn	←	←	←	↑	↑	↑	↑	←	←	-	←	←	←	←	↑	↑	↑	↑	↑	←
ESIGMNn	←	↑	↑	↑	↑	↑	↑	←	←	↑	-	←	↑	←	↑	↑	↑	↑	↑	↑
TDIGMNn	↑	↑	↑	↑	↑	↑	↑	←	←	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑
MIGMNn	←	←	←	↑	↑	↑	↑	←	←	↑	←	←	-	←	↑	↑	↑	↑	↑	↑
RecIGMNn	←	↑	↑	↑	↑	↑	↑	←	←	↑	↑	←	↑	-	↑	↑	↑	↑	↑	↑
TDESIGMN	←	←	←	↑	←	←	←	←	←	←	←	←	←	←	-	-	←	-	↑	←
TDMIGMN	←	←	←	↑	←	←	←	←	←	←	←	←	←	←	-	-	←	↑	↑	←
TDRecIGMN	←	←	←	↑	←	←	↑	←	←	←	←	←	←	←	↑	↑	-	↑	↑	←
TDESIGMNn	←	←	←	↑	←	←	←	←	←	←	←	←	←	←	-	←	←	-	↑	←
TDMIGMNn	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	-	←
TDRecIGMNn	←	←	←	↑	↑	↑	↑	←	←	↑	←	←	←	←	↑	↑	↑	↑	↑	-

Table B.10: Comparison of obtained one-step prediction errors for all algorithms in the log-differentiated airline passengers experiment.

	Mean	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Mean	-	←	↑	↑	←	-	↑	←	←	↑	←	←	↑	←	↑	←	↑	↑	↑	←
Elman	↑	-	↑	↑	↑	↑	↑	↑	←	↑	-	←	↑	↑	↑	←	↑	↑	↑	←
ESN	←	←	-	↑	←	-	↑	←	←	↑	←	←	↑	←	↑	←	↑	↑	↑	←
TDNN	←	←	←	-	←	←	←	←	←	←	←	←	←	←	-	←	←	-	-	←
IGMN	↑	←	↑	↑	-	↑	↑	←	←	↑	←	←	↑	←	↑	←	↑	↑	↑	←
ESIGMN	-	←	-	↑	←	-	↑	←	←	-	←	←	-	←	↑	←	↑	↑	↑	←
TDIGMN	←	←	←	↑	←	←	-	←	←	←	←	←	←	←	-	←	↑	↑	↑	←
MIGMN	↑	←	↑	↑	↑	↑	↑	-	←	↑	←	←	↑	←	↑	←	↑	↑	↑	←
RecIGMN	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	←	↑	↑	↑	←	↑	↑	↑	↑
IGMNn	←	←	←	↑	←	-	↑	←	←	-	←	←	↑	←	↑	←	↑	↑	↑	←
ESIGMNn	↑	-	↑	↑	↑	↑	↑	↑	←	↑	-	←	↑	↑	↑	←	↑	↑	↑	-
TDIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	←	↑	↑	↑	↑
MIGMNn	←	←	←	↑	←	-	↑	←	←	←	←	←	-	←	↑	←	↑	↑	↑	←
RecIGMNn	↑	←	↑	↑	↑	↑	↑	↑	←	↑	←	←	↑	-	↑	←	↑	↑	↑	←
TDESIGMN	←	←	←	-	←	←	-	←	←	←	←	←	←	←	-	←	-	-	↑	←
TDMIGMN	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑
TDRecIGMN	←	←	←	↑	←	←	←	←	←	←	←	←	←	←	-	←	-	↑	↑	←
TDESIGMNn	←	←	←	-	←	←	←	←	←	←	←	←	←	←	-	←	←	-	↑	←
TDMIGMNn	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	-	←
TDRecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	-	←	↑	↑	↑	←	↑	↑	↑	-

Table B.11: Comparison of obtained long-term prediction errors for all algorithms in the log-differentiated airline passengers experiment.

	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Elman	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
ESN	←	-	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
TDNN	←	-	-	←	←	←	←	←	-	-	-	-	-	←	←	←	-	←	-
IGMN	←	↑	↑	-	←	←	←	←	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
ESIGMN	←	↑	↑	↑	-	←	←	←	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
TDIGMN	←	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
MIGMN	←	↑	↑	↑	↑	←	-	-	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
RecIGMN	←	↑	↑	↑	↑	←	-	-	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
IGMNn	←	↑	-	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	-
ESIGMNn	←	↑	-	←	←	←	←	←	↑	-	-	↑	-	←	←	←	-	-	↑
TDIGMNn	←	↑	-	←	←	←	←	←	↑	-	-	↑	-	←	←	←	←	-	↑
MIGMNn	←	↑	-	←	←	←	←	←	↑	←	←	-	←	←	←	←	←	←	-
RecIGMNn	←	↑	-	←	←	←	←	←	↑	-	-	↑	-	←	←	←	-	-	↑
TDESIGMN	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	←	-	↑	↑	↑
TDMIGMN	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑
TDRecIGMN	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	←	-	↑	↑	↑
TDESIGMNn	←	↑	-	←	←	←	←	←	↑	-	↑	↑	-	←	←	←	-	-	↑
TDMIGMNn	←	↑	↑	←	←	←	←	←	↑	-	-	↑	-	←	←	←	-	-	↑
TDRecIGMNn	←	↑	-	←	←	←	←	←	-	←	←	-	←	←	←	←	←	←	-

Table B.12: Comparison of obtained runtimes for all algorithms in the log-differentiated airline passengers experiment.

	Trivial	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Trivial	-	←	←	↑	←	↑	↑	↑	↑	←	←	←	←	←	↑	↑	↑	←	←	←
Elman	↑	-	←	↑	↑	↑	↑	↑	↑	←	←	←	←	←	↑	↑	↑	←	←	←
ESN	↑	↑	-	↑	↑	↑	↑	↑	↑	←	←	←	←	←	↑	↑	↑	←	←	←
TDNN	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
IGMN	↑	←	←	↑	-	↑	↑	↑	↑	←	←	←	←	←	↑	↑	↑	←	←	←
ESIGMN	←	←	←	↑	←	-	↑	←	←	←	←	←	←	←	↑	←	↑	←	←	←
TDIGMN	←	←	←	↑	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←
MIGMN	←	←	←	↑	←	↑	↑	-	←	←	←	←	←	←	↑	←	↑	←	←	←
RecIGMN	←	←	←	↑	←	↑	↑	↑	-	←	←	←	←	←	↑	↑	↑	←	←	←
IGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	←	←	←	←	↑	↑	↑	←	←	←
ESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	←	←	↑	↑	↑	↑	↑	-
TDIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	-	←	←	↑	↑	↑	-	←	←
MIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	←	↑	↑	↑	↑	↑	↑
RecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑
TDESIGMN	←	←	←	↑	←	←	↑	←	←	←	←	←	←	←	-	←	↑	←	←	←
TDMIGMN	←	←	←	↑	←	↑	↑	↑	←	←	←	←	←	←	↑	-	↑	←	←	←
TDRecIGMN	←	←	←	↑	←	←	↑	←	←	←	←	←	←	←	←	←	-	←	←	←
TDESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	-	←	←	↑	↑	↑	-	-	←
TDMIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	←	←	↑	↑	↑	-	-	←
TDRecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	←	←	↑	↑	↑	↑	↑	-

Table B.13: Comparison of obtained one-step prediction errors for all algorithms in the Mackey-Glass (τ = 17) experiment.

	Mean	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Mean	-	←	↑	←	←	-	←	←	←	←	←	←	↑	←	←	↑	←	←	←	←
Elman	↑	-	↑	-	←	↑	↑	↑	←	↑	←	←	↑	←	↑	↑	↑	←	←	←
ESN	←	←	-	←	←	-	←	←	←	←	←	←	←	←	←	↑	←	←	←	←
TDNN	↑	-	↑	-	←	-	↑	↑	←	↑	←	←	↑	←	↑	↑	↑	←	←	←
IGMN	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	←	←	↑	↑	↑	↑	↑	←	←	←
ESIGMN	-	←	-	-	←	-	-	-	←	-	←	←	-	←	-	-	-	←	←	←
TDIGMN	↑	←	↑	←	←	-	-	↑	←	↑	←	←	↑	←	↑	↑	↑	←	←	←
MIGMN	↑	←	↑	←	←	-	←	-	←	←	←	←	↑	←	←	↑	←	←	←	←
RecIGMN	↑	↑	↑	↑	←	↑	↑	↑	-	↑	←	←	↑	←	↑	↑	↑	←	←	←
IGMNn	↑	←	↑	←	←	-	←	↑	←	-	←	←	↑	←	-	↑	←	←	←	←
ESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	←	↑	↑	↑	↑	↑	←	←	←
TDIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	-	←	↑
MIGMNn	←	←	↑	←	←	-	←	←	←	←	←	←	-	←	←	↑	←	←	←	←
RecIGMNn	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	←	←	↑	-	↑	↑	↑	←	←	←
TDESIGMN	↑	←	↑	←	←	-	←	↑	←	-	←	←	↑	←	-	↑	←	←	←	←
TDMIGMN	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	-	←	←	←	←
TDRecIGMN	↑	←	↑	←	←	-	←	↑	←	↑	←	←	↑	←	↑	↑	-	←	←	←
TDESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	-	-	↑
TDMIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	-	↑
TDRecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	←	←	-

Table B.14: Comparison of obtained long-term prediction errors for all algorithms in the Mackey-Glass (τ = 17) experiment.

	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Elman	-	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑
ESN	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
TDNN	←	↑	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
IGMN	←	↑	↑	-	↑	-	↑	←	↑	↑	↑	↑	↑	-	←	←	↑	↑	↑
ESIGMN	←	↑	↑	←	-	←	-	←	↑	↑	↑	↑	↑	-	←	←	-	↑	↑
TDIGMN	←	↑	↑	-	↑	-	↑	←	↑	↑	↑	↑	↑	-	←	←	↑	↑	↑
MIGMN	←	↑	↑	←	-	←	-	←	↑	↑	↑	↑	↑	-	←	←	↑	↑	↑
RecIGMN	-	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	-	←	↑	↑	↑
IGMNn	←	↑	↑	←	←	←	←	←	-	←	-	←	←	←	←	←	←	←	←
ESIGMNn	←	↑	↑	←	←	←	←	←	↑	-	↑	-	↑	←	←	←	←	←	↑
TDIGMNn	←	↑	↑	←	←	←	←	←	-	←	-	←	-	←	←	←	←	←	←
MIGMNn	←	↑	↑	←	←	←	←	←	↑	-	↑	-	↑	←	←	←	←	←	↑
RecIGMNn	←	↑	↑	←	←	←	←	←	↑	←	-	←	-	←	←	←	←	←	←
TDESIGMN	←	↑	↑	-	-	-	-	←	↑	↑	↑	↑	↑	-	←	←	↑	↑	↑
TDMIGMN	←	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	-	←	↑	↑	↑
TDRecIGMN	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑
TDESIGMNn	←	↑	↑	←	-	←	←	←	↑	↑	↑	↑	↑	←	←	←	-	↑	↑
TDMIGMNn	←	↑	↑	←	←	←	←	←	↑	↑	↑	↑	↑	←	←	←	←	-	↑
TDRecIGMNn	←	↑	↑	←	←	←	←	←	↑	←	↑	←	↑	←	←	←	←	←	-

Table B.15: Comparison of obtained runtimes for all algorithms in the Mackey-Glass (τ = 17) experiment.

	Trivial	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Trivial	-	↑	↑	↑	↑	↑	↑	↑	←	↑	←	↑	←	←	↑	↑	↑	↑	←	←
Elman	←	-	←	↑	←	↑	↑	↑	←	←	←	↑	←	←	↑	↑	↑	↑	←	←
ESN	←	↑	-	↑	↑	↑	↑	↑	←	-	←	↑	←	←	↑	↑	↑	↑	←	←
TDNN	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
IGMN	←	↑	←	↑	-	↑	↑	↑	←	←	←	↑	←	←	↑	↑	↑	↑	←	←
ESIGMN	←	←	←	↑	←	-	↑	←	←	←	←	←	←	←	↑	↑	←	←	←	←
TDIGMN	←	←	←	↑	←	←	-	←	←	←	←	←	←	←	-	←	←	←	←	←
MIGMN	←	←	←	↑	←	↑	↑	-	←	←	←	←	←	←	↑	↑	↑	←	←	←
RecIGMN	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	-	↑	←	←	↑	↑	↑	↑	↑	←
IGMNn	←	↑	-	↑	↑	↑	↑	↑	←	-	←	↑	←	←	↑	↑	↑	↑	←	←
ESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	-	↑	←	←	↑	↑	↑	↑	↑	←
TDIGMNn	←	←	←	↑	←	↑	↑	↑	←	←	←	-	←	←	↑	↑	↑	←	←	←
MIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	←	↑	↑	↑	↑	↑	←
RecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑
TDESIGMN	←	←	←	↑	←	←	-	←	←	←	←	←	←	←	-	←	←	←	←	←
TDMIGMN	←	←	←	↑	←	←	↑	←	←	←	←	←	←	←	↑	-	←	←	←	←
TDRecIGMN	←	←	←	↑	←	↑	↑	←	←	←	←	←	←	←	↑	↑	-	←	←	←
TDESIGMNn	←	←	←	↑	←	↑	↑	↑	←	←	←	↑	←	←	↑	↑	↑	-	←	←
TDMIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	←	↑	←	←	↑	↑	↑	↑	-	←
TDRecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	-

Table B.16: Comparison of obtained one-step prediction errors for all algorithms in the Mackey-Glass (τ = 30) experiment.

	Mean	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Mean	-	←	↑	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←
Elman	↑	-	↑	←	-	↑	←	-	←	←	←	←	←	←	←	←	←	←	←	←
ESN	←	←	-	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←
TDNN	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
IGMN	↑	-	↑	←	-	↑	←	↑	←	←	←	←	←	←	←	←	←	←	←	←
ESIGMN	-	←	-	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←
TDIGMN	↑	↑	↑	←	↑	↑	-	↑	←	↑	-	↑	↑	↑	←	←	←	↑	↑	↑
MIGMN	↑	-	↑	←	←	↑	←	-	←	←	←	←	←	←	←	←	←	←	←	←
RecIGMN	↑	↑	↑	←	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	-	←	↑	↑	↑	↑
IGMNn	↑	↑	↑	←	↑	↑	←	↑	←	-	←	←	←	↑	←	←	←	←	←	↑
ESIGMNn	↑	↑	↑	←	↑	↑	-	↑	←	↑	-	↑	-	↑	←	←	-	-	↑	↑
TDIGMNn	↑	↑	↑	←	↑	↑	←	↑	←	↑	←	-	←	↑	←	←	←	-	←	↑
MIGMNn	↑	↑	↑	←	↑	↑	←	↑	←	↑	-	↑	-	↑	←	←	←	↑	↑	↑
RecIGMNn	↑	↑	↑	←	↑	↑	←	↑	←	←	←	←	←	-	←	←	←	←	←	↑
TDESIGMN	↑	↑	↑	←	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	-	-	↑	↑	↑	↑
TDMIGMN	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	-	↑	↑	↑	↑
TDRecIGMN	↑	↑	↑	←	↑	↑	↑	↑	←	↑	-	↑	↑	↑	←	←	-	↑	↑	↑
TDESIGMNn	↑	↑	↑	←	↑	↑	←	↑	←	↑	-	-	←	↑	←	←	←	-	-	↑
TDMIGMNn	↑	↑	↑	←	↑	↑	←	↑	←	↑	←	↑	←	↑	←	←	←	-	-	↑
TDRecIGMNn	↑	↑	↑	←	↑	↑	←	↑	←	←	←	←	←	←	←	←	←	←	←	-

Table B.17: Comparison of obtained long-term prediction errors for all algorithms in the Mackey-Glass (τ = 30) experiment.

	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Elman	-	↑	↑	-	←	←	-	←	↑	↑	↑	↑	-	←	←	←	-	-	-
ESN	←	-	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
TDNN	←	-	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
IGMN	-	↑	↑	-	←	←	←	←	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
ESIGMN	↑	↑	↑	↑	-	←	↑	↑	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
TDIGMN	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	-	←	←	↑	↑	↑
MIGMN	-	↑	↑	↑	←	←	-	←	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
RecIGMN	↑	↑	↑	↑	←	←	↑	-	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
IGMNn	←	↑	↑	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←
ESIGMNn	←	↑	↑	←	←	←	←	←	↑	-	↑	↑	↑	←	←	←	←	-	←
TDIGMNn	←	↑	↑	←	←	←	←	←	↑	←	-	-	←	←	←	←	←	←	←
MIGMNn	←	↑	↑	←	←	←	←	←	↑	←	-	-	←	←	←	←	←	←	←
RecIGMNn	-	↑	↑	←	←	←	←	←	↑	←	↑	↑	-	←	←	←	←	←	←
TDESIGMN	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	-	←	←	↑	↑	↑
TDMIGMN	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑
TDRecIGMN	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	-	↑	↑	↑
TDESIGMNn	-	↑	↑	←	←	←	←	←	↑	↑	↑	↑	↑	←	←	←	-	↑	↑
TDMIGMNn	-	↑	↑	←	←	←	←	←	↑	-	↑	↑	↑	←	←	←	←	-	-
TDRecIGMNn	-	↑	↑	←	←	←	←	←	↑	↑	↑	↑	↑	←	←	←	←	-	-

Table B.18: Comparison of obtained runtimes for all algorithms in the Mackey-Glass (τ = 30) experiment.

	Trivial	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Trivial	-	-	-	←	←	-	←	←	↑	←	←	←	←	←	←	←	←	←	←	←
Elman	-	-	-	-	-	-	←	←	-	←	←	←	←	←	←	←	←	←	←	←
ESN	-	-	-	←	-	-	←	←	-	←	←	←	←	←	←	←	←	←	←	←
TDNN	↑	-	↑	-	↑	↑	←	←	↑	←	←	←	←	←	-	←	↑	←	←	←
IGMN	↑	-	-	←	-	-	←	←	↑	←	←	←	←	←	←	←	←	←	←	←
ESIGMN	-	-	-	←	-	-	←	←	-	←	←	←	←	←	←	←	←	←	←	←
TDIGMN	↑	↑	↑	↑	↑	↑	-	←	↑	↑	←	←	←	←	↑	←	↑	←	←	←
MIGMN	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	←	←	←	←	↑	←	↑	←	←	←
RecIGMN	-	-	-	←	←	-	←	←	-	←	←	←	←	←	←	←	←	←	←	←
IGMNn	↑	↑	↑	↑	↑	↑	←	←	↑	-	←	←	←	←	↑	←	↑	←	←	←
ESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	←	←	↑	-	↑	↑	←	←
TDIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	-	←	←	↑	←	↑	←	←	←
MIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	←	↑	↑	↑	↑	↑	↑
RecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑
TDESIGMN	↑	↑	↑	-	↑	↑	←	←	↑	←	←	←	←	←	-	←	↑	←	←	←
TDMIGMN	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	←	←	↑	-	↑	↑	←	←
TDRecIGMN	↑	↑	↑	←	↑	↑	←	←	↑	←	←	←	←	←	←	←	-	←	←	←
TDESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	←	←	↑	←	↑	-	←	←
TDMIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	←	↑	↑	↑	↑	-	↑
TDRecIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	←	↑	↑	↑	↑	←	-

Table B.19: Comparison of obtained one-step prediction errors for all algorithms in the Google stock experiment.

	Mean	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Mean	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
Elman	←	-	←	-	←	-	-	-	-	←	←	←	←	←	-	←	-	←	←	↑
ESN	←	↑	-	-	-	-	↑	↑	↑	←	-	←	-	←	-	-	↑	←	←	↑
TDNN	←	-	-	-	-	-	↑	↑	↑	←	←	←	-	←	-	-	↑	←	←	↑
IGMN	←	↑	-	-	-	-	↑	↑	↑	←	←	←	↑	←	-	←	↑	←	←	↑
ESIGMN	←	-	-	-	-	-	↑	↑	↑	←	←	←	-	←	-	-	↑	←	←	↑
TDIGMN	←	-	←	←	←	←	-	↑	↑	←	←	←	←	←	←	←	←	←	←	↑
MIGMN	←	-	←	←	←	←	←	-	↑	←	←	←	←	←	←	←	←	←	←	↑
RecIGMN	←	-	←	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←	↑
IGMNn	←	↑	↑	↑	↑	↑	↑	↑	↑	-	-	←	↑	←	↑	↑	↑	←	←	↑
ESIGMNn	←	↑	-	↑	↑	↑	↑	↑	↑	-	-	←	↑	-	↑	↑	↑	←	←	↑
TDIGMNn	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑
MIGMNn	←	↑	-	-	←	-	↑	↑	↑	←	←	←	-	←	-	←	↑	←	←	↑
RecIGMNn	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	←	↑	-	↑	↑	↑	←	←	↑
TDESIGMN	←	-	-	-	-	-	↑	↑	↑	←	←	←	-	←	-	-	-	←	←	↑
TDMIGMN	←	↑	-	-	↑	-	↑	↑	↑	←	←	←	↑	←	-	-	↑	←	←	↑
TDRecIGMN	←	-	←	←	←	←	↑	↑	↑	←	←	←	←	←	-	←	-	←	←	↑
TDESIGMNn	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	-	-	↑
TDMIGMNn	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	↑	↑	↑	↑	↑	-	-	↑
TDRecIGMNn	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	-

Table B.20: Comparison of obtained long-term prediction errors for all algorithms in the Google stock experiment.

	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Elman	-	↑	↑	←	-	←	-	←	-	-	-	-	-	←	←	←	←	←	←
ESN	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
TDNN	←	↑	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←
IGMN	↑	↑	↑	-	↑	-	↑	-	↑	↑	↑	↑	↑	-	-	-	↑	↑	-
ESIGMN	-	↑	↑	←	-	-	↑	-	↑	↑	↑	↑	↑	←	←	←	-	-	-
TDIGMN	↑	↑	↑	-	-	-	↑	-	↑	↑	↑	↑	↑	←	←	←	-	-	-
MIGMN	-	↑	↑	←	←	←	-	←	↑	↑	↑	↑	↑	←	←	←	-	-	-
RecIGMN	↑	↑	↑	-	-	-	↑	-	↑	↑	↑	↑	↑	←	←	←	-	↑	-
IGMNn	-	↑	↑	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←
ESIGMNn	-	↑	↑	←	←	←	←	←	↑	-	↑	-	-	←	←	←	←	←	←
TDIGMNn	-	↑	↑	←	←	←	←	←	↑	←	-	←	-	←	←	←	←	←	←
MIGMNn	-	↑	↑	←	←	←	←	←	↑	-	↑	-	-	←	←	←	←	←	←
RecIGMNn	-	↑	↑	←	←	←	←	←	↑	-	-	-	-	←	←	←	←	←	←
TDESIGMN	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	-	↑	↑	↑
TDMIGMN	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	←	-	←	↑	↑	-
TDRecIGMN	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	-	↑	↑	↑
TDESIGMNn	↑	↑	↑	←	-	-	-	-	↑	↑	↑	↑	↑	←	←	←	-	-	-
TDMIGMNn	↑	↑	↑	←	-	-	-	←	↑	↑	↑	↑	↑	←	←	←	-	-	-
TDRecIGMNn	↑	↑	↑	-	-	-	-	-	↑	↑	↑	↑	↑	←	-	←	-	-	-

Table B.21: Comparison of obtained runtimes for all algorithms in the Google stock experiment.

	Trivial	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Trivial	-	-	↑	←	←	←	←	←	↑	←	←	←	←	←	←	←	←	←	←	←
Elman	-	-	↑	←	-	←	-	←	↑	-	←	-	←	←	←	←	←	←	←	-
ESN	←	←	-	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←
TDNN	↑	↑	↑	-	↑	-	↑	↑	↑	↑	←	↑	←	-	-	←	↑	←	←	↑
IGMN	↑	-	↑	←	-	←	←	←	↑	←	←	↑	←	←	←	←	←	←	←	←
ESIGMN	↑	↑	↑	-	↑	-	↑	-	↑	↑	←	↑	←	←	-	←	↑	←	←	↑
TDIGMN	↑	-	↑	←	↑	←	-	←	↑	←	←	↑	←	←	←	←	←	←	←	↑
MIGMN	↑	↑	↑	←	↑	-	↑	-	↑	↑	←	↑	←	←	←	←	↑	←	←	↑
RecIGMN	←	←	-	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←	←
IGMNn	↑	-	↑	←	↑	←	↑	←	↑	-	←	↑	←	←	←	←	←	←	←	↑
ESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	←	↑	↑	←	↑	-	←	↑
TDIGMNn	↑	-	↑	←	←	←	←	←	↑	←	←	-	←	←	←	←	←	←	←	←
MIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	←	↑	↑	←	↑
RecIGMNn	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	←	↑	←	-	↑	←	↑	←	←	↑
TDESIGMN	↑	↑	↑	-	↑	-	↑	↑	↑	↑	←	↑	←	←	-	←	↑	←	←	↑
TDMIGMN	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	↑	←	↑
TDRecIGMN	↑	↑	↑	←	↑	←	↑	←	↑	↑	←	↑	←	←	←	←	-	←	←	↑
TDESIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑	←	↑	↑	←	↑	-	←	↑
TDMIGMNn	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑
TDRecIGMNn	↑	-	↑	←	↑	←	←	←	↑	←	←	↑	←	←	←	←	←	←	←	-

Table B.22: Comparison of obtained one-step prediction errors for all algorithms in the U.S. interest rate experiment.

	Mean	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Mean	-	←	-	←	←	←	←	←	←	↑	←	←	←	←	←	←	←	←	←	↑
Elman	↑	-	↑	←	←	-	←	-	-	↑	-	←	↑	←	←	←	←	←	←	↑
ESN	-	←	-	←	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	-
TDNN	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	-	↑	↑	-	↑
IGMN	↑	↑	↑	←	-	↑	←	↑	↑	↑	↑	←	↑	←	←	←	←	←	←	↑
ESIGMN	↑	-	↑	←	←	-	←	↑	↑	↑	-	←	↑	←	←	←	←	←	←	↑
TDIGMN	↑	↑	↑	←	↑	↑	-	↑	↑	↑	↑	↑	↑	←	←	←	↑	-	←	↑
MIGMN	↑	-	↑	←	←	←	←	-	↑	↑	←	←	↑	←	←	←	←	←	←	↑
RecIGMN	↑	-	↑	←	←	←	←	←	-	↑	←	←	↑	←	←	←	←	←	←	↑
IGMNn	←	←	-	←	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	↑
ESIGMNn	↑	-	↑	←	←	-	←	↑	↑	↑	-	←	↑	←	←	←	←	←	←	↑
TDIGMNn	↑	↑	↑	←	↑	↑	←	↑	↑	↑	↑	-	↑	←	←	←	↑	←	←	↑
MIGMNn	↑	←	↑	←	←	←	←	←	←	↑	←	←	-	←	←	←	←	←	←	↑
RecIGMNn	↑	↑	↑	←	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	-	←	↑	-	←	↑
TDESIGMN	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	-	-	↑	-	←	↑
TDMIGMN	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	-	↑	-	←	↑
TDRecIGMN	↑	↑	↑	←	↑	↑	←	↑	↑	↑	↑	←	↑	←	←	←	-	←	←	↑
TDESIGMNn	↑	↑	↑	←	↑	↑	-	↑	↑	↑	↑	↑	↑	-	-	-	↑	-	←	↑
TDMIGMNn	↑	↑	↑	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	-	↑
TDRecIGMNn	←	←	-	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	←	-

Table B.23: Comparison of obtained long-term prediction errors for all algorithms in the U.S. interest rate experiment.

	Elman	ESN	TDNN	IGMN	ESIGMN	TDIGMN	MIGMN	RecIGMN	IGMNn	ESIGMNn	TDIGMNn	MIGMNn	RecIGMNn	TDESIGMN	TDMIGMN	TDRecIGMN	TDESIGMNn	TDMIGMNn	TDRecIGMNn
Elman	-	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
ESN	←	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TDNN	←	-	-	-	←	←	←	←	↑	-	↑	↑	-	←	←	←	-	↑	-
IGMN	←	-	-	-	←	←	←	←	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
ESIGMN	←	-	↑	↑	-	↑	-	-	↑	↑	↑	↑	↑	-	-	-	↑	↑	↑
TDIGMN	←	-	↑	↑	←	-	←	-	↑	↑	↑	↑	↑	←	←	←	↑	↑	↑
MIGMN	←	-	↑	↑	-	↑	-	-	↑	↑	↑	↑	↑	-	-	-	↑	↑	↑
RecIGMN	←	-	↑	↑	-	-	-	-	↑	↑	↑	↑	↑	-	-	-	↑	↑	↑
IGMNn	←	-	←	←	←	←	←	←	-	←	←	←	←	←	←	←	←	←	←
ESIGMNn	←	-	-	←	←	←	←	←	↑	-	↑	↑	-	←	←	←	↑	↑	↑
TDIGMNn	←	-	←	←	←	←	←	←	↑	←	-	←	←	←	←	←	←	←	←
MIGMNn	←	-	←	←	←	←	←	←	↑	←	↑	-	←	←	←	←	-	-	-
RecIGMNn	←	-	-	←	←	←	←	←	↑	-	↑	↑	-	←	←	←	-	↑	-
TDESIGMN	←	-	↑	↑	-	↑	-	-	↑	↑	↑	↑	↑	-	-	-	↑	↑	↑
TDMIGMN	←	-	↑	↑	-	↑	-	-	↑	↑	↑	↑	↑	-	-	-	↑	↑	↑
TDRecIGMN	←	-	↑	↑	-	↑	-	-	↑	↑	↑	↑	↑	-	-	-	↑	↑	↑
TDESIGMNn	←	-	-	←	←	←	←	←	↑	←	↑	-	-	←	←	←	-	-	-
TDMIGMNn	←	-	←	←	←	←	←	←	↑	←	↑	-	←	←	←	←	-	-	-
TDRecIGMNn	←	-	-	←	←	←	←	←	↑	←	↑	-	-	←	←	←	-	-	-

Table B.24: Comparison of obtained runtimes for all algorithms in the U.S. interest rate experiment.

APPENDIX C RESUMO EM PORTUGUÊS

C.1 Introdução

O aprendizado de sequências temporais é uma tarefa que se aplica a áreas como previsão de séries temporais e controle. Às vezes é necessário que os modelos aprendidos estejam sempre atualizados (aprendizado agressivo) com um fluxo constante de dados (aprendizado incremental) enquanto em operação (aprendizado online). Este é o caso com sistemas de monitoramento e controle em robótica, por exemplo.

Os algoritmos Incremental Gaussian Mixture Model (IGMM, ou Modelo de Mistura de Gaussianas Incremental, em português) (ENGEL; HEINEN, 2010) (ENGEL; HEINEN, 2011) e Incremental Gaussian Mixture Network (IGMN, ou Rede de Mistura de Gaussianas Incremental, em português) (HEINEN; ENGEL, 2010a) (HEINEN; ENGEL, 2011a) (HEINEN; ENGEL; PINTO, 2011) (HEINEN, 2011) foram recentemente propostos como novos algoritmos baseados em redes neurais para clustering e classificação/regressão, respectivamente. Estes algoritmos permitem um aprendizado agressivo online e incremental, enquanto evitam em grande parte o ajuste manual de parâmetros críticos para este aprendizado.

Apesar de terem sido aplicados com sucesso a muitos problemas, incluindo controle de robôs
(HEINEN; ENGEL, 2011b) (HEINEN; ENGEL, 2010b), o IGMM e a IGMN são algoritmos estáticos ou puramente espaciais, o que significa que eles não conseguem lidar com problemas com estados internos ou memória (a menos que esses estados sejam explicitamente adicionados aos dados manualmente). Possuindo uma versão dinâmica da IGMN, seria possível lidar com problemas dinâmicos e não-markovianos de uma forma online e incremental, com uma única passada sobre os dados, que é o objetivo deste trabalho. Existem ao menos duas abordagens para atingir este objetivo, a saber, estender algoritmos online incrementais temporais (como redes neurais recorrentes) para aprendizado agressivo, ou estender algoritmos online incrementais agressivos para processamento temporal, que é a abordagem usada aqui.

C.1.1 Principais Contribuições

Este trabalho propõe algumas variações espaço-temporais para o algoritmo IGMN. Devido ao IGMM poder ser visto essencialmente como um algoritmo avançado de clustering, espera-se que técnicas já verificadas para estender algoritmos de clustering (especialmente os Mapas Auto-Organizáveis) para o domínio temporal devam funcionar também com o IGMM, e portanto com a IGMN. Neste sentido, três novos algoritmos são propostos: Echo-State IGMN (ESIGMN), Merge IGMN (MIGMN) e Recursive IGMN (RecIGMN). Além disso, a IGMN com linhas de atraso é avaliada contra eles e outros algoritmos clássicos, e é chamada de Time-Delay IGMN (TDIGMN) aqui. Além dos novos algoritmos para estender temporalmente a IGMN estática, uma nova técnica é também introduzida para melhorará-la mesmo em sua forma estática, o que também remove um de seus parâmetros que, de outra forma, deveria ser selecionado manualmente. Sumarizando as contribuições:

- Echo-State Incremental Gaussian Mixture Network (ESIGMN)
- Merge Incremental Gaussian Mixture Network (MIGMN)
- Recursive Incremental Gaussian Mixture Network (RecIGMN)
- Criação de componentes baseada em "outliers"

C.2 Conclusões

Este trabalho apresentou três novas extensões temporais para o algoritmo IGMN, bem como uma nova regra para criação de componentes.

O capítulo 2 apresentou todos os algoritmos usados como base para as novas extensões, cobrindo a maioria dos aspectos necessários para implementá-los.

O capítulo 3 descreveu todas as contribuições deste trabalho, começando com a nova regra para criação de componentes. Essa regra permitiu que a IGMN aprendesse com sucesso sem a necessidade do parâmetro ϵ_{max}, e resultou em melhores (mais intuitivas e econômicas) soluções em um experimento simples. Isto é muito importante para fins de obter um aprendizado verdadeiramente autônomo e genérico, já que o ajuste fino manual de parâmetros envolve conhecimento humano de cada problema. Eliminar os parâmetros restantes da IGMN é algo para ser explorado em trabalhos futuros. Depois disto, a TDIGMN, que usa uma janela deslisante com entradas passadas, foi revisada (ela não é um novo algoritmo, mas não foi descrita explicitamente como um algoritmo temporal em trabalhos anteriores) como um aperfeiçoamento temporal para a IGMN. Então, as três novas extensões foram descritas reusando as técncias mostradas anteriormente para outras redes neurais temporais: ESIGMN com sua camada de reservoir dinâmico inspirada pela ESN; MIGMN, que usa uma média móvel exponencial das entradas e saídas reconstruídas como seu contexto temporal, similar ao MSOM; e a RecIGMN, inspirada pelo RecSOM e a rede de Elman, acrescentando conexões de realimentação a partir das ativações de suas componentes Gaussianas de volta para a entrada.

No capítulo 4, esses novos algoritmos foram comparados com redes neurais temporais clássicas e a IGMN estática, para verificar seus desempenhos na tarefa de previsão de séries temporais, usando 8 diferentes séries-temporais. Os novos algoritmos foram testados com suas versões naïve e multivariadas, bem como com e sem linhas de atraso adicionais (como as usadas pela TDIGMN). Um total de 19 algoritmos foram testados e os resultados analisados. Nós descobrimos que a ESIGMN e a MIGMN são úteis como algoritmos online, incrementais, agressivos e temporais , até mesmo superando as redes neurais temporais clássicas em diversos experimentos. Nós achamos que a RecIGMN, apesar de não ter sido uma boa contribuição para este conjunto, tem potencial se corretamente implementada, com um parâmetro de ponderação entre passado e presente e talvez mais informações de suas componentes Gaussianas sendo realimentadas, tais como valores de erro. A MIGMN também foi apresentada sem um parâmetro de ponderação de passado e presente, e esta possibilidade deve ser explorada. A ESIGMN tem conexões externas para o seu reservoir apenas vindo das entradas, e uma versão com realimentação das saídas ainda está por ser explorada. Também descobrimos que linhas de atraso são benéficas em geral para os novos algoritmos, complementando suas memórias irrestritas e inexatas com uma memória limitada e exata (isto pode ser visto como os papéis complementares da memória de curto prazo e memória de trabalho em nossos cérebros, respectivamente). Todos os algoritmos propostos obtiveram resultados muito bons em termos de tempo de execução também, tornando-os bons candidatos para aplicações de tempo real. Ainda está por ser explorado como eles lidam com entradas de mais alta dimensionalidade, como vídeos. A inversão de matriz ainda é o gargalo de todas as versões multivariadas dos algoritmos apresentados, e as versões naïve provaram-se ineficientes para tarefas temporais, já que elas não conseguem capturar as covariâncias entre as entradas temporalmente aumentadas altamente correlacionadas dos novos algoritmos.

Portanto, concluímos que os objetivos iniciais deste trabalho foram atingidos, que eram de criar algoritmos online, incrementais, agressivos e temporais usando a IGMN como base. Sumarizando as contribuições e conclusões neste trabalho:

- Nova regra para criação de componentes para a IGMN e suas extensões, resultando em melhor performance e eliminando um parâmetro ajustável manualmente;
- Algoritmo ESIGMN, que obteve resultados muito bons em geral;
- Algoritmo MIGMN, que também obteve resultados muito bons;
- Algoritmo RecIGMN, que requer melhoramentos para tornar-se competitivo;
- Implementações em Matlab de todos os algoritmos propostos e da própria IGMN, que serão reusadas em trabalhos futuros e também pelo nosso grupo de pesquisa e pesquisadores externos (de fato, já estão sendo usadas).

Como trabalhos futuros, podemos enumerar os seguintes:

- Verificar os benefícios da inclusão de realimentação das saídas da ESIGMN, bem como melhores técnicas de inicialização do reservoir;
- Verificar os benefícios da inclusão de parâmetros de ponderação entre passado e presente na MIGMN e RecIGMN (e tentar ajustá-los automaticamente);
- Analisar diferentes contextos temporais para a RecIGMN, usando mais informações de suas componentes gaussianas;
- Aplicação dos algoritmos propostos em tarefas de aprendizado por reforço em processos Markovianos parcialmente observáveis (POMDP);
- Pesquisar memórias de mais longo prazo;
- Verificar a viabilidade dos algoritmos propostos em problemas de mais alta dimensionalidade e também em problemas mais do mundo real como robótica.