A COMPARISON OF TIME SERIES FORECASTING LEARNING ALGORITHMS ON THE TASK OF PREDICTING EVENT TIMING

UNE COMPARAISON DES ALGORITHMS D'APPENTISSAGE DE PREVISION DES SERIES TEMPORELLES SUR LA PREDICTION DU TEMPS DES EVENEMENTS

A Thesis Submitted to the Division of Graduate Studies
of the Royal Military College of Canada
by

Najmatoullahi Amadou Boukary, B.A.Sc

In Partial Fulfillment of the Requirements for the Degree of
Master of Applied Science in Computer Science

January, 2016

Acknowledgements

The author is very grateful to GOD ALMIGHTY for without His grace and blessings, this study would not have been possible.

Immeasurable appreciation and deepest gratitude for the help and support are extended to the following persons who in one way or another have contributed in making this study possible.

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Francois Rivest, for his ongoing support, his patience, motivation, enthusiasm, and for providing me with an excellent atmosphere for doing research. His guidance and words of encouragements helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor. He introduced me first to the machine learning field, and gave me the freedom and the encouragement to pursue my own ideas.

I would like especially to thank my fellow labmates Rene Sturgeon and Richar Kohar in particular who took the time to read my thesis and gave me valuables feedbacks. I am fortunate to have been a part of such an incredibly fantastic machine learning group.

A special thanks to the members of the department of Mathematics and Computer science at the Royal Military College of Canada. The atmosphere, faculty, and students were outstanding in all dimensions.

I want to thank the committee Dr. Kelly, Dr. Liang, Dr. Grygoryev and Dr. Jaansalu for agreeing to read and evaluate my thesis.

I am indebted to my husband, Boubacar Housseini, for his support and to my parents, for the love and support they provided me through my entire life.

Abstract

Time series analysis is widely used in various fields such as business, economics, finance, science, and engineering. A time series is a sequence of values or observations of a variable recorded at sequential points in time. One of the main purposes of time series data is that past observations of the data can be used to forecast future values. Many time series forecasting algorithms from machine learning and statistics have been proposed in the literature. This thesis focuses on a particular form of time series: the binary time series that contains instantaneous events that occur over time. However, predicting future upcoming events is extremely difficult. The sum squared error (SSE) is a common measure of performance that has difficulties for predicting time series with long-term dependences. In fact, the majority of machine learning algorithms try to minimize the sum squared error of each output at every time step. Therefore, a new computational approach using interval timing has been proposed in the literature. The idea of this approach, inspired by how animals learn, suggest to predict *"when"* a particular event will occur in the future instead of *"what"* event will occur at the next time step by minimizing the timing error for each event.

An online learning algorithm namely time adaptive drift-diffusion model (TDDM) derived from this concept has been developed by Rivest et al. [1]. The TDDM algorithm is based on a bounded accumulation process similar to drift-diffusion models (DDM) that models animal decision making. It learns to predict the time remaining before an event occurs using binary input streams.

This thesis aims at first developing an offline regression-based TDDM algorithm that learns to predict the time remaining before instantaneous events occur, and second at comparing some state-of-the art time series forecasting models from statistics (such as the multivariate vector auto-regressive moving average (VARMA) algorithm) and machine learning (such as the echo state network (ESN) algorithm) to TDDM on predicting when instantaneous events will occur.

A detailed description and classification of the three algorithms is given while highlighting their strengths and drawbacks. We compare these three regression-based algorithms on three real world datasets namely the Bach Chorales for music notes, the NASDAQ stock prices, and the MIT-BIH Arrhythmia database. Two sets of experiments were conducted. In the first one, the algorithms tried to predict binary instantaneous event streams at each time step while in the second they learned to predict the time remaining before the events occur. It was found that the three algorithms have difficulties learning to predict binary event stream but perform much better in the prediction of the time remaining before the occurrence of events.

Résumé

L’analyse de séries chronologiques est largement utilisée dans divers domaines tels que le commerce, l'économie, la finance, la science et l'ingénierie. Une série chronologique ou temporelle est une séquence de valeurs ou d'observations d'une variable enregistrée en des points successifs dans le temps. L'un des objectifs principaux des données de séries chronologiques est que les observations passées peuvent être utilisées pour prévoir les valeurs futures. Beaucoup de modèles de prévision des séries chronologiques provenant du domaine de l'apprentissage machine et celui de la statistique ont été proposées dans la littérature. Cette thèse s’intéresse à une forme particulière de séries temporelles: la série de temps binaire qui contient des événements instantanés qui se produisent au fil du temps. Cependant, la prévision des événements futurs à venir est extrêmement difficile. La mesure d’erreurs au carré (SSE) est une mesure commune de performance qui se révèle être un mauvais choix pour ce type de prédiction. En fait, la majorité des algorithmes d'apprentissage machine essayent de minimiser la SSE de chaque sortie, à chaque pas de temps. Par conséquent, une nouvelle approche de calcul en utilisant la synchronisation d'intervalle a été proposée dans la littérature. L'idée de cette approche, inspirée par la façon dont les animaux apprennent, suggèrent de prévoir *"quand"* un événement particulier se produira dans l'avenir au lieu de *"quoi"* qui se produira au prochain pas de temps en minimisant les erreurs de synchronisation pour chaque événement.

Un algorithme d'apprentissage en ligne basé sur les modèles d'adaptation de dérivé -diffusion (TDDM) dérivé de ce concept a été développé par Rivest et al. [1]. L’algorithme TDDM est basé sur un processus d'accumulation similaire aux modèles (DDM) inspirée de l’apprentissage animale décision. Ce model apprend à prédire le temps restant avant qu’un événement se produit en utilisant des entrées binaires.

Cette thèse vise à développer d'abord un algorithme *hors-ligne* basé sur la régression qui apprend à prédire le temps restant avant que des événements instantanés se produisent, et ensuite à le comparer avec certains modèles connus pour prévision des séries chronologiques d’abord en statistique (à savoir l’algorithme vecteur auto régressive moyenne mobile (VARMA)) puis en apprentissage machine (avec l’algorithme du réseau de neurones (ESN)) sur la prévision des événements instantanés.

Une description détaillée des trois algorithmes est donnée tout en soulignant leurs avantages et leurs inconvénients. Nous comparons ces trois algorithmes basées sur la régression sur les trois ensembles de données du monde réel à savoir Bach Chorales pour les notes de musique, les cours boursiers NASDAQ, et la base de données MIT-BIH d’arythmie. Deux séries d'expériences ont été réalisées. Dans la première, les algorithmes vont essayer de prédire des événements instantanés binaires à chaque pas de temps alors que dans le second, ils apprendront à prédire le temps restant avant que les événements se produisent. Il a été constaté que les trois algorithmes ont des difficultés d'apprentissage pour prédire des événements binaire, mais beaucoup plus performants dans la prédiction de la durée restante avant la survenance d'événements.

Contents

List of Tables

List of Figures

List of Symbols

$\boldsymbol{x}_{\mathbf{t}}$ **,** $\boldsymbol{x}(\boldsymbol{t})$: input binary time series vector of *N* dimensions at time step *t*

$\boldsymbol{y}_{\boldsymbol{t}}, \boldsymbol{y}(\boldsymbol{t})$: output time remaining time series vector of *M* dimensions at time step *t*

$\boldsymbol{z}_{\mathbf{t}},\ \boldsymbol{z}(\boldsymbol{t})$: output events time series vector of *M* dimensions at time step *t*

$\widehat{\boldsymbol{y}}_{\boldsymbol{t}}, \widehat{\boldsymbol{y}}(\boldsymbol{t})$: Time remaining forecasts vector of *M* dimensions for time *t*

$\widehat{\boldsymbol{z}}_{\boldsymbol{t}}$ **,** $\widehat{\boldsymbol{z}}(\boldsymbol{t})$: Events forecasts vector of *M* dimensions for time *t*

$x_i(t)$**:** observation of variable x_i at time step *t*

$y_j(t)$**:** observation of variable y_j at time step *t*

$z_j(t)$**:** observation of variable z_j at time step *t*

$\hat{y}_j(t)$: Time remaining forecast of variable y_j at time step *t*

$\hat{z}_j(t)$: Event forecast of variable z_j at time step *t*

t: time step

i: variable index for $\boldsymbol{x}(\boldsymbol{t})$ from 1…*N*

j: variable index for $\boldsymbol{y}(\boldsymbol{t})$ or $\boldsymbol{z}(\boldsymbol{t})$ from1…*M*

$\boldsymbol{\varepsilon}_{\boldsymbol{t}}$: white noise vector

p: order of the Autoregressive model

q: order of the moving average model

$\alpha_1, \dots, \alpha_p$: parameters of the autoregressive model

A: matrix of parameters for the autoregressive model

$\beta_1, \beta_2, \dots . \beta_q$: parameters of the moving average model

B: matrix of parameters for the moving average model

w*:* vector of weights or parameters

$\boldsymbol{w}_{\boldsymbol{j}}$*:* vector of weights or parameters for output *j*

e_t: error at time t

M: output dimension (number of variables in $\boldsymbol{y}(\boldsymbol{t})$ at time t)

N: input dimension (number of variables in $\boldsymbol{x}(\boldsymbol{t})$ at time t)

k: relative time index

List of Acronyms and Abbreviations

ANNs: Artificial Neural Networks

AR: Auto-Regressive

ARCH: Auto-Regressive Conditional Heteroscedastic

ARMA: Auto-regressive Moving Average

ARIMA: Auto-regressive Integrated Moving Average

BP: Back-propagation

ECG: Electrocardiogram

DDM: Drift-Diffusion Model

ESN: Echo State Networks

FFNN: Feed Forward Neural Network

LSTM: Long Short Term Memory Network

MA: Moving Average

MAE: Mean Absolute Error

MSE: Mean Squared Error

MLP: Multilayer Perceptron

MTSF: Multiple Time Step Forecasting

NAR: Nonlinear Autoregressive

NARX: Nonlinear Autoregressive with eXogenous inputs

RBF: Radial Basis Function

RNN: Recurrent Neural Network

SARIMA: Seasonal Auto-regressive Integrated Moving Average

SVM: Support Vector Machine

TAR: Threshold Autoregressive

TDDM: Time Adaptive Drift-Diffusion Model

TDNN: Time Delay Neural Network

TSF: Time Series Forecasting

VARMA: Vector Auto-regressive Moving Average

1 Introduction

Time series forecasting (TSF) is crucial to various practical domains. By definition, a time series is a sequence of observations on a phenomenon recorded over time (or space) [2] (see Figure 1). Often, these observations become available at discrete, equal time intervals. One purpose of time series data is that it can be used in forecasting. The use of observations from a time series available at time t to predict its value at time $t + l$ is called forecasting; where l is called the forecasting horizon or lead time [3]. The forecasting horizon is the number of time steps in the future for which the forecasts must be produce.

Forecasting provides a basis for decision making, economic and business planning, inventory management, and the control and optimization of industrial process. For example, applications of TSF in economy and finance are forecasting gross domestic product (GDP), sales forecasting, prediction of stock markets ; in science, one can found earthquake forecasting, weather forecasting; in management, there are room bookings forecasting, emergency calls forecasting; in the medical field, one can find predictions of infectious diseases and so on.

A forecasting method or algorithm is a procedure that computes forecast $\hat{y}_t$ for time t from past values ($y_{t-1}, y_{t-2}, y_{t-3}$ ...). Numerous TSF algorithms from machine learning and statistics have been proposed. The standard statistical forecasting models are the most predominant in the literature followed by the machine learning artificial neural networks (ANNs) models. Most of the latter usually focus on forecasting the output value of the time series at each time step by minimizing the sum squared error (SSE) cost function.

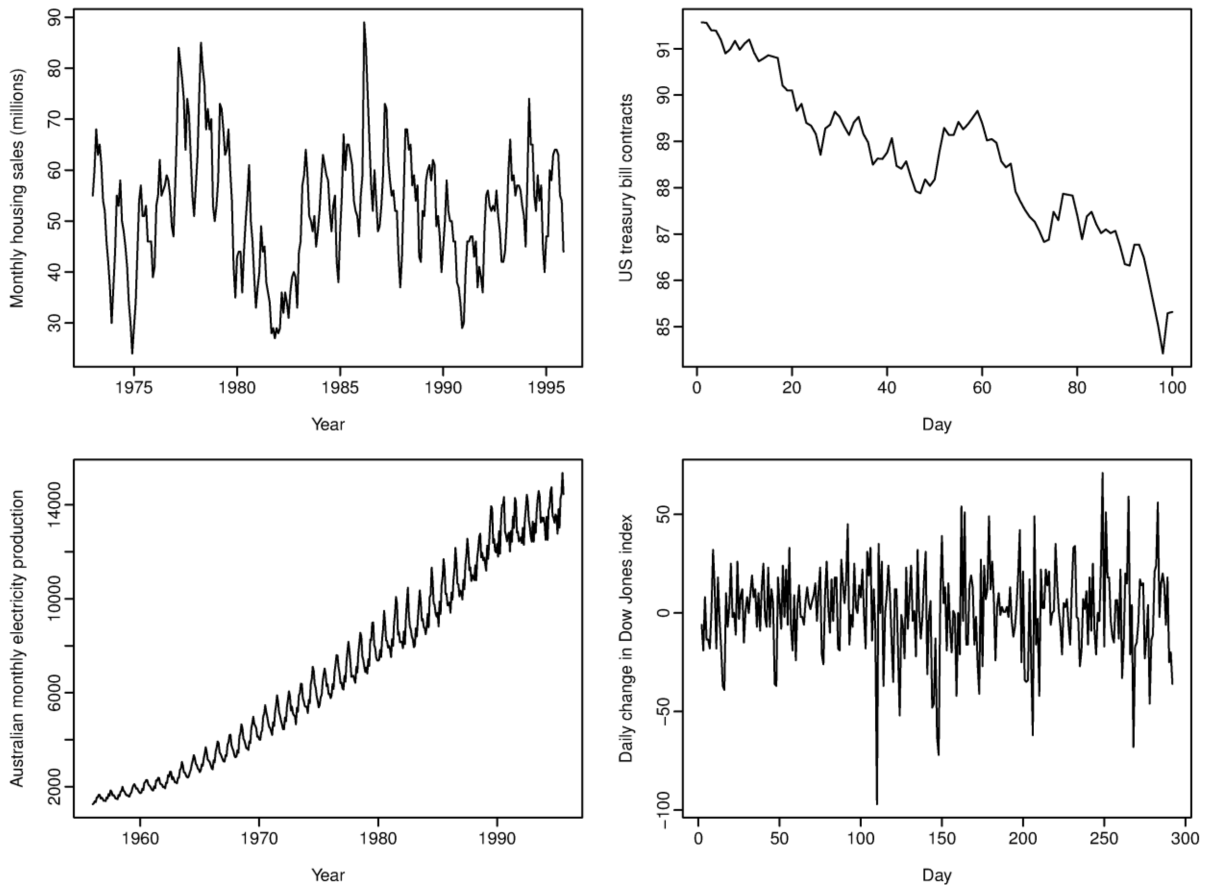


Figure 1: Examples of different time series plots from [4]

The SSE is one of the most widely used cost functions in machine learning and in time series forecasting. It is obtained by summing the square of difference between known observations and predicted observations. The goal of many algorithms is to minimize this cost function using a gradient descent algorithm or other similar optimization techniques. However, this computational approach ends up being a poor choice for learning algorithms since a gradient-descent approach is numerically unstable for tasks requiring a long history or memory of previous observations [5]. As a matter of fact, training ANNs to learn long term dependencies is very difficult [6] . One of the tasks is to learn the precise timing of events using binary data. Although there has been some successful results with these applications [7] using ANNs, the training process remains computationally expensive (slow) and in some cases the network is shown not to learn at all [8].

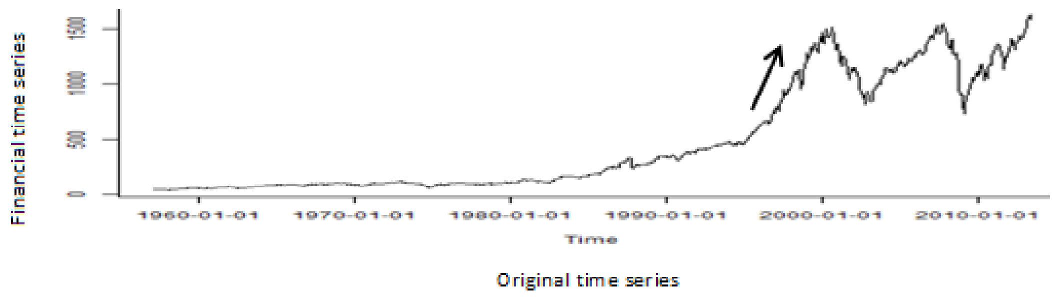
On the other hand, animals learn the timing between consecutives events very easily [9]. The success of animals learning timing events is underlying by the fact that they are interested in knowing when the event will occur instead of what will happened at each time step. An approach has recently been proposed by Rivest et al. [10] inspired by the animal model. Instead of minimizing the sum squared error of each output at every time step (SSE), to minimize directly the squared timing error (STE) for every event.

In the case of instantaneous events, instead of predicting the complete time series at every time step, Rivest et al. [10] make a single prediction for the length of time after which the event will occur instantaneously. In order to learn event timing, the original time series has to be transformed into binary time series and time remaining series (see Figure 2). In this way, instead of having access to the entire time series observations, the algorithm will have access to only relevant observations and will learn to predict the time remaining before these relevant observations occur.

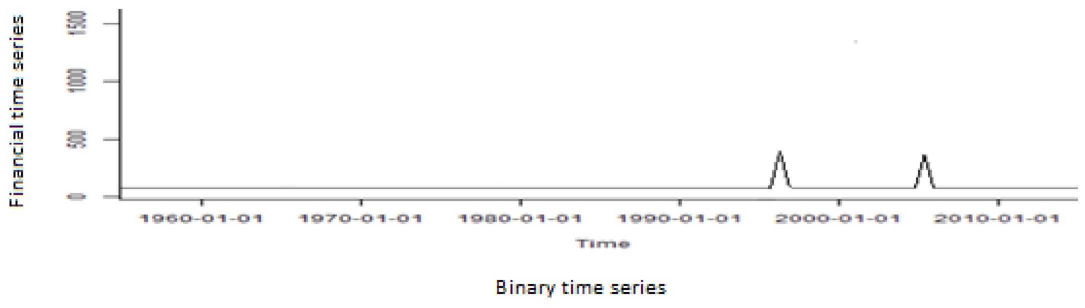
An online time-adaptive drift-diffusion model [1] (TDDM) learning algorithm derived from this concept has been developed. The TDDM algorithm is based on a bounded accumulation process similar to drift-diffusion models (DDM) of decision making. It learns to predict the time remaining before an event occurs using binary input streams by learning the relative event-rate of each input stimulus for each event type. Then it uses each observation to accumulate evidence that a specific event is going to occur.

In the light of this, an offline version of the time adaptive drift-diffusion model (TDDM) algorithm proposed by Rivest & al. [1] has been developed and compared to the multivariate vector auto-regressive moving average (VARMA) [11], a standard well known statistical algorithm, and echo state network (ESN) [12] , a standard recurrent neural network algorithm on two tasks. The first task focuses on training the three algorithms to learn the precise timing of events i.e. to learn to predict a specific value (1) at the time step the event shall occur. This task as stated before is difficult to learn [8]. The second task is an alternative proposed by Rivest & al., instead of learning the precise timing of events, to learn the time remaining before they occur. All three methods will be using regression to make the best possible timing predictions. Experiments are performed on three real world datasets including the NASDAQ stock prices the MIT-BIH Arrhythmia database and the Bach Chorales with the target of predicting the day of significantly higher stock price, the next heartbeat, and the note onset and offset respectively.

a



b



c

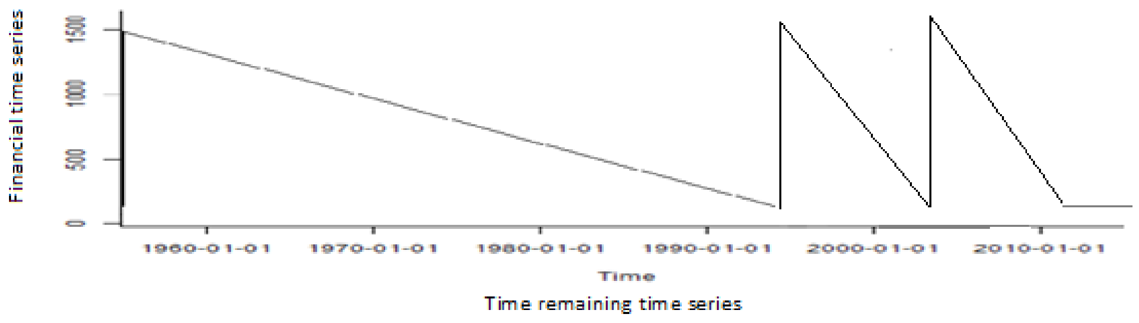


Figure 2: Example of transformations of time series data **(a)** into binary time series of events **(b)** and into time remaining time series **(c)**.

1.1 Thesis Statement

The research objectives are as follows:

a) First an offline TDDM learning algorithm for forecasting time series is developed. It is derived from the online time adaptive drift-diffusion model algorithm (TDDM) [1] which is inspired by animal learning model. It is based on a bounded accumulation process similar to drift-diffusion process (DDM). The offline TDDM algorithm will be developed as a regression based model and expended to predict the time remaining before an event occurs using binary input stream and time remaining data.

b) The predictive ability of the state-of-the art VARMA and ESN algorithms is evaluated on the task of :
    - First, forecasting events on precise time steps.
    - Second, forecasting the time remaining before an event occurs.

1.2 Motivation

Although there has been extensive research carried out in the time series forecasting field, the search for simple and fast algorithms that produce reliable forecast has never ceased. The current state-of-the art algorithms in the forecasting domain still have limitations. For example, traditional statistical models such as the autoregressive AR (p) and the moving average MA (q) are linear models. They are appropriate for only some type of time series data (eg. autoregressive models are more appropriate for stationary series) [13] adding to that their inabilities to extract complex relationships in the data [14].

The use of the SSE cost function to obtain prediction intervals for long term forecasts or to learn long-term dependencies may simply not be adequate. For learning long term dependencies, machine learning algorithms that try to learn this task are computationally expensive in the training process. Examples are the standard recurrent neural networks which have shown their inability to successfully learn timing events due to the problem of the vanishing gradient descent of the error through time [8] .

In the quest for simple, practical, fast and accurate learning algorithms, a new perspective suggests to give up or change the SSE. Thus instead of clinging to the hope of minimizing the SSE for each output at every time step of a time series which involved long term dependencies in data Figure 2(a) such as the events time series Figure 2(b), this thesis will look at the idea of directly minimizing the squared timing error for every particular event Figure 2 (c). A particular event for example can be when the stock prices will go high or down past set thresholds when trying to do a stock price forecasting.

This idea has been inspired by how animals learn. In fact, researchers have proved that animals have shown their capability to learn events timing very easily and rapidly unlike machine learning time series algorithms [9]. The biologically inspired approach of learning timing is an interesting example of prediction and a significant factor to take into account. F. Rivest et al [10], explain that the success of animal learning timing of events suggests that they are predicting events in a different way than we do in machine learning : “Animals are predicting when an event will occur by estimating the time remaining before particular events occurs instead of what event will occur” at the upcoming time steps . An online learning time series algorithm (TDDM) based on this concept has been proposed in the literature. It learns to predict the time remaining before an event occur using binary inputs stream by learning the relative event-rate of each input stimulus for each event type.

Using the forecasting timing concept, some state-of-the art time series forecasting algorithms in statistics (using the VARMA algorithm) and in machine learning (using the ESN algorithm) have been investigated in order to see if they could be applied to predict when an event will occur as well. In addition, based on the online time adaptive drift-diffusion algorithm (TDDM) [1] an offline linear regression based TDDM algorithm has been developed so that it can be used for performance comparison to study offline VARMA and ESN algorithms. The comparison was based on two tasks. First, the algorithms will try to predict instantaneous events at the precise time step they shall occur and second, the algorithms will try to predict the time remaining before instantaneous events occur. The results show that it is indeed much easier to predict the remaining time than to predict events at the precise time step they shall occur.

1.3 Contributions

The contributions of this thesis are the following:

- Provides a review and a comparison of the existing time series forecasting algorithms in the statistical and machine learning fields.
- An evaluation of the statistical VARMA algorithm and the ESN algorithm on predicting timing events.
- An offline TDDM algorithm for learning timing events.

- A comparative study of the performance of the three algorithms on two different formulation of the problem:
    - Predicting the events at precise time steps,
    - Predicting remaining time at each time step.

The results show that predicting the events at precise time steps is hardly learned by all the algorithms as oppose to the time remaining predictions which was in general learned by the three algorithms.

1.4 Organization

This thesis is organized as follows. Chapter 2 provides a literature review of time series forecasting algorithms in the statistical and machine learning literature, including their descriptions, advantages and drawbacks. In Chapter 3, the problem of forecasting timing event and time remaining are formulated. In Chapter 4, the TDDM offline algorithm is described. Chapter 5 describes the methodology and the evaluation procedure used in this research as well as the experimental results from training the three algorithms on three real world datasets, namely the NASDAQ-100 stock closure prices time series, the Bach Chorales time series and the heartbeats. The last chapter states and summaries the results of the thesis.

2 Literature Review

In this chapter, from research literature, a description of univariate and multivariate time series forecasting models is first presented followed by their application, their advantages, and lastly their drawbacks. Section 2.1 start with an overview of forecasting methods. The standard statistical forecasting models and the machine learning forecasting models, especially artificial neural networks, are presented respectively in Section 2.2 and Section 2.3. Finally in Section 2.4, some hybrid models found in the literature that are a combination of statistical and machine learning models are given.

2.1 Forecasting

Forecasting is the act of predicting the future based on the history. Using forecasting methods or algorithms, past data or information of a time series are processed in order to predict the future trends or outcomes of these data or time series.

In statistics, the primary goal of forecasting is to provide valuable information for decision making, economic and business planning, inventory and production control, control and optimization of industrial process. There are two approaches to forecasting in this field: qualitative and quantitative approaches. The qualitative methods also called judgemental or subjective forecasting methods includes techniques that are based on the intuition, the judgement or the opinion of consumers, experts, commercial knowledge and any other relevant information. A common qualitative method is the Delphi technique [15]. Qualitative techniques are frequently used when past data are not available. The quantitative approach covers univariate and multivariate forecasting techniques. In univariate forecasting, forecasts are made depending only on present and past data of the single series being forecasted. However for many problems in economics, business, physical and environmental sciences, the time series data may be available on several related variables, which in this case, multivariate forecasting techniques are applied. Moreover in the multivariate case, one can find methods where forecasts of a given variable depend, at least partly, on values of one or more additional time series variables, called predictor or explanatory variables [2]. This section will focus on presenting univariate and multivariate forecasting models.

In machine learning, the main advantage of forecasting is to develop automated algorithms that can learn and make some predictions from data. Many machine learning methods of forecasting have been proposed in the literature. The common used method is based on ANNs which will be discussed in detail in section 2.3.

2.2 Statistical Forecasting Models

This section covers some common algorithms used for time series forecasting in statistics. The forecasting field has been influenced, for a long time, by linear statistical methods such as the autoregressive (AR) model, the moving average (MA) model, and hybrid models that derive from them such as ARMA (autoregressive moving average), ARIMA (autoregressive integrated moving average), and SARIMA (seasonal ARIMA).

2.2.1 Autoregressive (AR) Model

In the autoregressive process, an output variable y_t depends linearly on its own previous values $(y_{t-1}, \dots, y_{t-p})$ and some white noise ε_t. By definition, a process $\{y_t\}$ is said to be an autoregressive process of order *p* denoted AR (*p*) if y_t can be described by:

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \cdots + \alpha_p y_{t-p} + \varepsilon_t \qquad (1)$$

where ε_t is white noise with mean zero and fixed finite variance σ_Z^2 [2] , and $\alpha_1, \dots, \alpha_p$ are the parameters of the model. The order p of the model determines the number of past observations used to predict the current value. The simplest example of an AR process is the first-order case, denoted AR (1), given by

$$y_t = \alpha_1 y_{t-1} + \varepsilon_t \qquad (2)$$

In the multivariable case where there are multiple observations for each time step, then we can consider a multivariate autoregressive or a vector autoregressive (VAR) model. Consider *M* time series generated from *M* variables, a VAR (*p*) model is defined by the following equation

$$\boldsymbol{y_t} = \sum_{k=1}^{p} A_k \boldsymbol{y_{t-k}} + \boldsymbol{\varepsilon_t} \qquad (3)$$

where $\boldsymbol{y_t}$=[$y_t^{(1)}, y_t^{(2)}, \dots, y_t^{(M)}]^T$ is *M*-dimensional time series column vector at index *t*. Each A_k is an *M*-by-*M* matrix of parameters where $A^{(k)}{}_{i,j}$ is the element at position (*i*,*j*) in matrix A_k and $\boldsymbol{\varepsilon_t}$=[$\varepsilon_t^{(1)} \dots \varepsilon_t^{(M)}]^T$ is a column vector of white noises.

Equation (3) can be re-written in a matrix form as:

$$\begin{bmatrix} y_t^{(1)} \\ \vdots \\ y_t^{(M)} \end{bmatrix} = \sum_{k=1}^{p} \begin{bmatrix} A_{11}^{(k)} & \cdots & A_{1M}^{(k)} \\ \vdots & \ddots & \vdots \\ A_{M1}^{(k)} & \cdots & A_{MM}^{(k)} \end{bmatrix} \begin{bmatrix} y_{t-1}^{(1)} \\ \vdots \\ y_{t-1}^{(M)} \end{bmatrix} + \begin{bmatrix} \varepsilon_t^{(1)} \\ \vdots \\ \varepsilon_t^{(M)} \end{bmatrix} \tag{4}$$

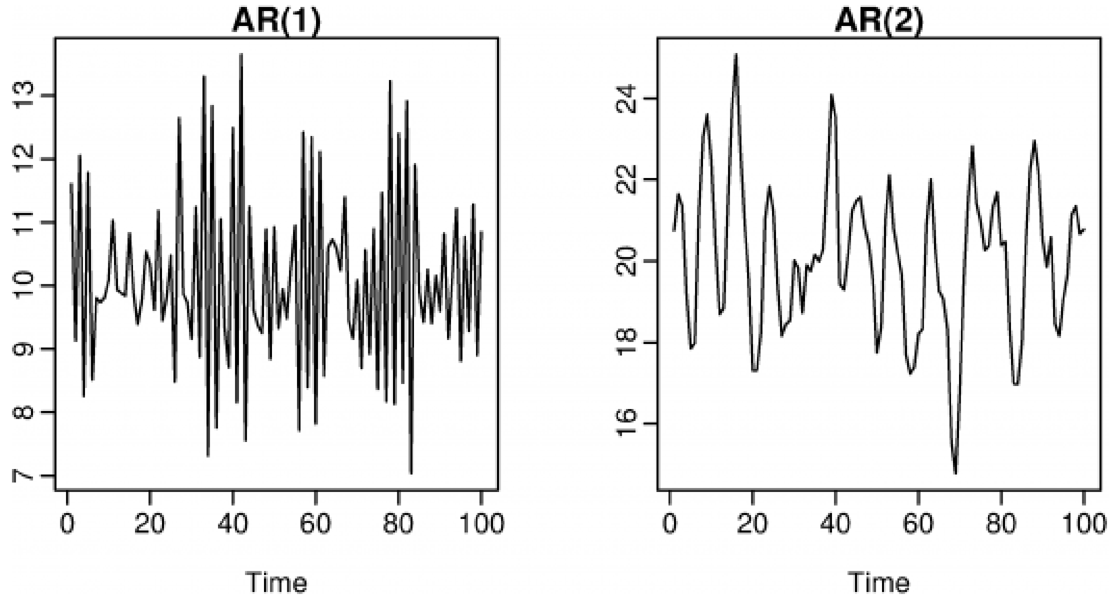


Figure 3: Two examples of data from autoregressive models with different parameters reprinted from [16]. Left: AR (1) with $y_t = 18 - 0.8y_{t-1} + \varepsilon_t$. Right: AR (2) with $y_t = 8 + 1.3y_{t-1} - 0.7y_{t-2} + \varepsilon_t$. In both cases, ε_t is normally distributed white noise with mean zero and variance one.

2.2.2 Moving Average (AM) Model

Suppose that $\{\varepsilon_t\}$ is a purely random process with mean zero and variance σ_Z^2 , then a process $\{y_t\}$ is said to be a moving average process $\{y_t\}$ of order q denoted MA(q)) if y_t can be expressed by

$$y_t = \varepsilon_t + \beta_1\varepsilon_{t-1} + \beta_2\varepsilon_{t-2} + \cdots + \beta_q\varepsilon_{t-q} \tag{5}$$

where $\beta_1, \beta_2, \ldots . \beta_q$ are parameters of the model [2], [11].

The moving average also describes a method where the next sample depends on the weighted sum of the past or present inputs of an exogenous time series $\{x_t\}$ of N dimensions described in the equation (6) below.

$$y_t = \beta_0 x_t + \beta_1 x_{t-1} + \beta_2 x_{t-2} + \cdots + \beta_q x_{t-q} \qquad (6)$$

Similar to the AR (p) model, in the case of multiple time series, a multivariate MA (q) model of M-dimension can be written as

$$\boldsymbol{y_t} = \sum_{k=0}^{q} B_k \boldsymbol{x_{t-k}} \qquad (7)$$

where $\boldsymbol{x_t}$ is an exogenous N-dimension time series and B_k are M-by-N matrices of parameters.

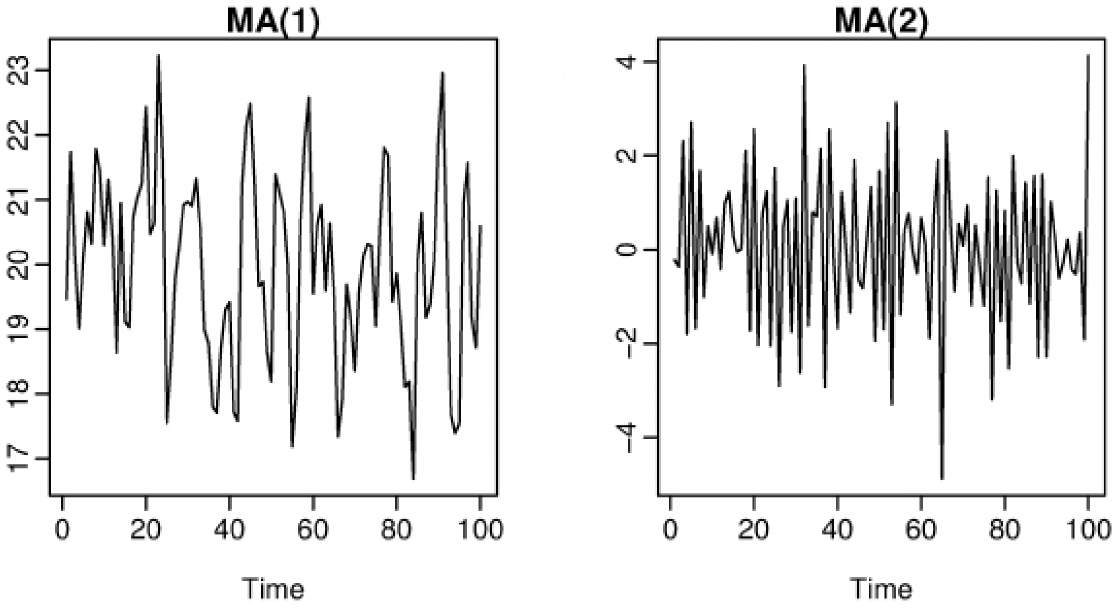


Figure 4: Two examples of data from moving average models with different parameters reprinted from [16]. Left: MA(1) with $x_t = 20 + \varepsilon_t + 0.8\varepsilon_{t-1}$. Right: MA(2) with $x_t = \varepsilon_t - 1\varepsilon_{t-1} + 0.8\varepsilon_{t-2}$. In both cases, ε_t is normally distributed white noise with mean zero and variance one.

2.2.3 ARMA (Autoregressive Moving Average) Model

The ARMA model is one of the most widely used models since it combines the advantages of the auto-regressive AR(p) and the moving average MA(q) models. The ARMA model was originally proposed in 1951 by Peter Whittle in his thesis “Hypothesis testing in time series analysis” and was adapted by George E. P. Box and Gwilym Jenkins in 1971 [11]. An ARMA (p, q) model of order (p, q) is defined by

$$y_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p} + \varepsilon_t + \beta_1 \varepsilon_{t-1} + \cdots + \beta_q \varepsilon_{t-q} \qquad (8)$$

where y_t is the original series and ε_t is a series of unknown random errors which are assumed to followed the normal probability distribution.

The multivariate version of the ARMA model is called vector auto-regressive moving average (VARMA) which is given by

$$\boldsymbol{y_t} = \sum_{k=1}^{p} A_k \boldsymbol{y_{t-k}} + \sum_{k=0}^{q} B_k\, \boldsymbol{x_{t-k}} \qquad (9)$$

where $\boldsymbol{y_t}$ is the output, $\boldsymbol{y_{t-k}}$ and $\boldsymbol{x_{t-k}}$ are respectively the past outputs variables and the past inputs exogenous variables and A_k and B_k are M-by-M and M-by-N matrices of parameters respectively.

2.2.4 ARIMA (Autoregressive Integrated Moving Average) Model

The models defined previously as AR, MA, and ARMA are used in stationary time series analysis [14]. A time series is said to be stationary, if the mean of the series and the covariance among its observations do not change over time and do not follow any trend [17]. In practice, most time series are non-stationary, so in order to fit stationary models, it is indispensable to get rid of the non-stationary sources of variation [2]. One solution to this, introduced by Box and Jenkins [11], is the ARIMA model which generally overcomes this limitation by introducing a differencing process which effectively transforms the non-stationary data into a stationary one [2], [17]. This is done by subtracting the observation in the current period from the previous one. For example a first order differencing is done by replacing y_t by $y_t = y_t - y_{t-1}$. Hence, the ARIMA model is called "Integrated" ARMA because of the stationary model that is fitted to the differenced data that has to be summed or integrated in order to provide a model for the original non stationary data. The general form of the ARIMA *(p, d, q)* process is described as

$$y'_t = \nabla^k y_t = \alpha_1 y'_{t-1} + \cdots + \alpha_p y'_{t-p} + \varepsilon_t + \beta_1 \varepsilon_{t-1} + \cdots + \beta_q \varepsilon_{t-q} \qquad (10)$$

where parameters *p*, *d*, and *q* are non-negative integers that refer to the order of the autoregressive part, the degree of first differencing involved, and the order of moving average part respectively. This capacity to deal with non-stationary process has made the ARIMA model one of the most popular and widely used approaches in time series forecasting [14].

2.2.5 SARIMA (Seasonal ARIMA) Model

SARIMA [2] is an extension to the ARIMA model. It is used when the data presents with a periodic characteristic which must be known ahead. For example, the seasonal component that repeats every *s* observations can be monthly $s = 12$ (12 in 1year) or quarterly $s = 4$ (4 in 1 year). The SARIMA model is usually termed as ARIMA *(p, d, q)* X $(P, D, Q)_s$ where *P*=number of seasonal autoregressive (SAR) terms, *D*=number of seasonal differences, Q=number of seasonal moving average (SMA) terms.

A standard example of seasonal time series is the airline passenger time series, from Box and Jenkins [11]. The dataset includes the number of international airline passengers (in thousands) for each month from January 1949 to December 1960 (see Figure 5 below).

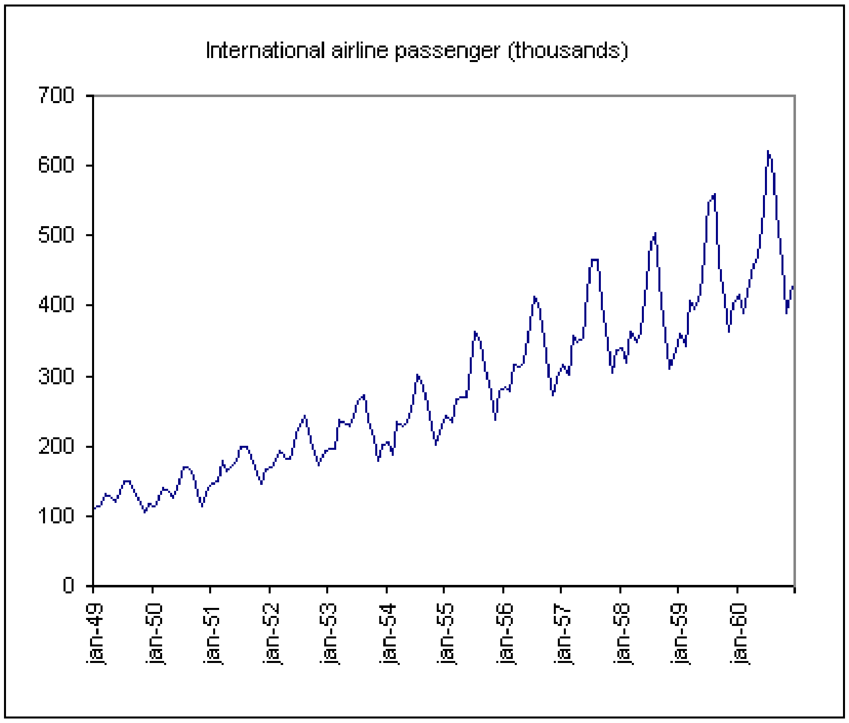


Figure 5: Time Series Plot of the Airline Passenger Seasonal Series from [11]

2.2.6 Training and Evaluation of Statistical Forecasting Methods

In the statistical literature, the data is often divided in two sets [16]. The first set is used for estimating the parameters of the model: this part is called training. The second set is called the testing set which contains unseen data by the model that is used for estimating the forecasts using the parameters. The test set gives us a way to try the model on data that was unavailable to the model when the parameters were first calculated. From this, we can see how well the model performs at predicting other data for which we know the real outcome in order to compare this to the predicted outcome. The size of the test set is typically about 20% of the last part of the total sample length although this value may change on how far ahead the forecasts are needed and on how long the sample is. Other statistical references call the training set the "in-sample data" and the test set the "out-of-sample data". The parameters are found in the training part by using mostly linear regression.

To identify the ability of the predictions, different forecast accuracy measures are used depending on the task and the model. An error measure is simply defined by the difference between a forecast and the actual value. Denote y_t as the t^{th} observation and $\hat{y}_t$ as the forecast of y_t . The forecast error is defined as $e_t = y_t - \hat{y}_t$, which is on the same scale as the data and cannot be used to make comparisons between series that are on different scales. Commonly used measures are the mean squared error (MSE), the mean absolute error (MAE), the root mean squared error (RMSE) and the mean absolute percentage error (MAPE). A list of commonly used forecast accuracy measures are show in the Table 1 [18] below.

MSE	Mean squared error	$=\text{mean}(e_t^2)$
RMSE	Root mean squared error	$=\sqrt{\text{MSE}}$
MAE Mean	Absolute error	$=\text{mean}(\lvert e_t\rvert)$
MdAE	Median absolute error	$=\text{median}(\lvert e_t\rvert)$
MAPE	Mean absolute percentage error	$=\text{mean}(\lvert p_t\rvert)$
MdAPE	Median absolute percentage error	$=\text{median}(\lvert p_t\rvert)$
sMAPE	Symmetric mean absolute percentage error	$=\text{mean}(2\lvert Y_t-F_t\rvert/(Y_t+F_t))$
sMdAPE	Symmetric median absolute percentage error	$=\text{median}(2\lvert Y_t-F_t\rvert/(Y_t+F_t))$
MRAE	Mean relative absolute error	$=\text{mean}(\lvert r_t\rvert)$
MdRAE	Median relative absolute error	$=\text{median}(\lvert r_t\rvert)$
GMRAE	Geometric mean relative absolute error	$=\text{gmean}(\lvert r_t\rvert)$
RelMAE	Relative mean absolute error	$=\text{MAE}/\text{MAE}_\text{b}$
RelRMSE	Relative root mean squared error	$=\text{RMSE}/\text{RMSE}_\text{b}$
LMR	Log mean squared error ratio	$=\log(\text{RelMSE})$
PB	Percentage better	$=100\ \text{mean}(I\{\lvert r_t\rvert<1\})$
PB(MAE)	Percentage better (MAE)	$=100\ \text{mean}(I\{\text{MAE}<\text{MAE}_\text{b}\})$
PB(MSE)	Percentage better (MSE)	$=100\ \text{mean}(I\{\text{MSE}<\text{MSE}_\text{b}\})$

Table 1: Commonly used forecast accuracy measures from [18]

2.2.7 Applications

The literature survey shows that most of these standard models for time series forecasting are rarely used alone. They are often mixed with others models in order to have more accurate and reliable forecasts. Nevertheless, we found several papers using only the AR model in the forecasting area. Hence, Forzieri et al. [19] used the AR model to forecast the spatiotemporal variability of terrestrial vegetation in environmental domain. An autoregressive model of order two (or AR(2)) with a MAPE value of 1.27% was found to be an appropriate model for forecasting the Malaysian peak daily load of electricity for the three days ahead prediction [20]. It is also widely used for short term load forecasting in the power generation domain [21], [22]. Yinghui et al. [23] explored the importance and predictive power of different frequency bands of subcutaneous glucose signals for the short-term forecasting of glucose concentrations in type 1 diabetic patients by using data-driven autoregressive (AR) models while Chen and al [24] used the AR model and the VAR (Vector Autoregressive) model to forecast three macroeconomic variables of Taiwan inflation based on consumer price index. Hsueh-Fang Chien et al. [25] forecast the monthly sales of cell-phone companies using the vector auto-regressive model. Applications of the ARMA models in the literature cover short load forecasting in power systems [26], [27] and [28] and short term traffic flow forecasting [29].

The ARIMA model is commonly used in the domain of finance and economics due to its capacities to handle non-stationary time series. In fact, several authors [17], [30] , [31] and [32] used ARIMA model to predict electricity price. In medical applications, Abraham et al. [33] worked on forecasting emergency admissions in hospitals. Another strength of ARIMA model is that it is mixed with others methods particularly artificial neural networks (ANN) models to produce forecasts in various domains. Hybrid models are discussed further in section 2.4. SARIMA models have been also widely used in the medical field for forecasting infectious diseases [14] such as hemorrhagic fever with renal syndrome [34], dengue fever [35], and tuberculosis [36].

The traditional statistical models we have seen so far including AR, MA, ARMA, ARIMA and SARIMA are all linear models over the previous inputs and/or states. Their advantages are that they are well understood, easy to compute, and provide stable forecasts. However, in the late 1970s and early 1980s, it became increasingly obvious that linear models are not adapted to many real applications [37]. Their major limitation is their pre-assumed linearity form of the data. Hence, despite their relative simplicity in understanding and implementation, linear models cannot capture nonlinear patterns [38]. To overcome the linear limitations of these time series models, a few models have been proposed in the literature. These include the bilinear model, the Threshold Autoregressive (TAR) model, and the Autoregressive Conditional Heteroscedastic (ARCH) model [38]. For example, the ARCH model has been recently used in [39] for forecasting Internet traffic. Even though some amelioration has been noticed with these nonlinear models, still, they cannot be generalized to the forecasting problem because they are developed for specific nonlinear patterns, and are not capable of modeling other types of nonlinearity in time series [38]. Therefore, development of ANNs in the late 80s raises some interest for more capable alternatives.

2.3 Machine Learning Forecasting Models

This section covers some machine learning models used in time series forecasting mainly artificial neural networks (ANNs). ANNs originated with the mathematical modelling of how the human brain works. Biological neurons (see Figure 6) provide the transmission of bioelectrical signals. They receive as input a signal set, and when these signals reach a certain threshold (sufficiently large enough); the neuron is activated and the signals are routed through the axon to generate an output.

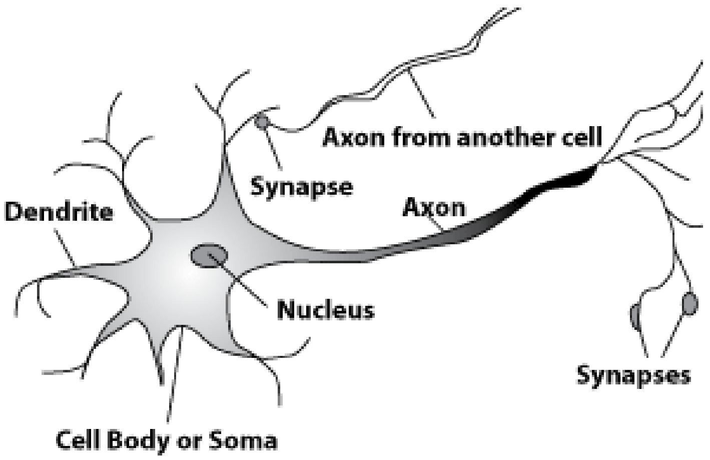


Figure 6: Biological Neuron from [40]

An artificial neuron (see Figure 7) comes to imitate the behavior of a biological neuron. In the very beginning, McCulloch and Pitts developed the first artificial neuron model in 1943. ANNs consist of a set of neurons or nodes connected together. Each node is associated with a set of weights and can be seen as a computational unit. The network receives inputs, and processes them through one or more layers of nodes to obtain an output see Figure 7 and Figure 8. In each layer, the weighted sum (the sum of the inputs, multiplied by its respective weight) is computed and then passed through a squashing or activation function. The activation function is a nonlinear function that is usually sigmoidal. The connections determine the information flow between nodes. By adjusting the weights of the nodes, one can obtain the specific or desired output for a given set of inputs. The weights are adjusted by a learning algorithm. This process is called training or learning.

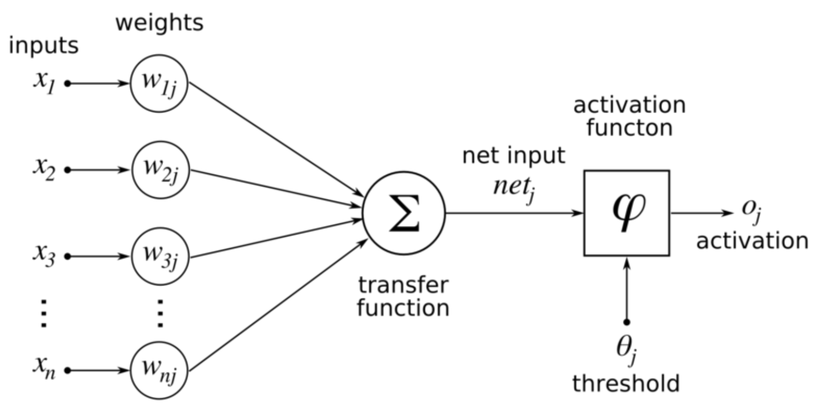


Figure 7: Basic Artificial neuron reprinted from [41]

More recently, ANNs have been extensively studied and used in time series forecasting. Unlike the traditional statistical model-based methods they have shown their flexible nonlinear modeling capabilities [42]. Moreover, ANNs can generalize in the sense that they are able to learn from experiences and then generalize to new experiences never seen before. Lastly, ANNs are data-driven, self-adaptive methods which mean that there are few *a priori* assumptions on the underlying process from which data are generated [43].

Since the model of McCulloch and Pitts, a wide variety of ANNs has been developed. The differences might be in the architectures, the activation function, the learning algorithms etc…depending on the practical problem to solve. Among others there are feed-forward neural networks (FFNNs) or multilayer perceptron's (MLP) (see Figure 8) which are the basic neural networks architecture [44].

There exist as well recurrent neural networks (RNNs); among them one can find simple recurrent neural networks, long short term memory network (LSTM) [45] and echo state networks (ESN) [12]. Unlike standard FFNNs, RNNs can deal with sequential input data, using their internal memory to process arbitrary sequences of inputs. This is possible by using feedback connections or loop between neurons hence making them more powerful than FFNNs [8]. One can also find among ANNs architecture radial basis function (RBF) [46], cascading neural networks [47] and support vector machines (SVM) [48]. This research will focus on three architectures that are widely used for time series forecasting and are dealing with long term dependencies, namely feed forward neural networks, LSTMs, and ESNs.

2.3.1 Feed-forward neural network

FFNNs (see Figure 8) are the standard artificial neural network architecture. They consist in general of three layers (one input layer, one hidden layer and one output layer).

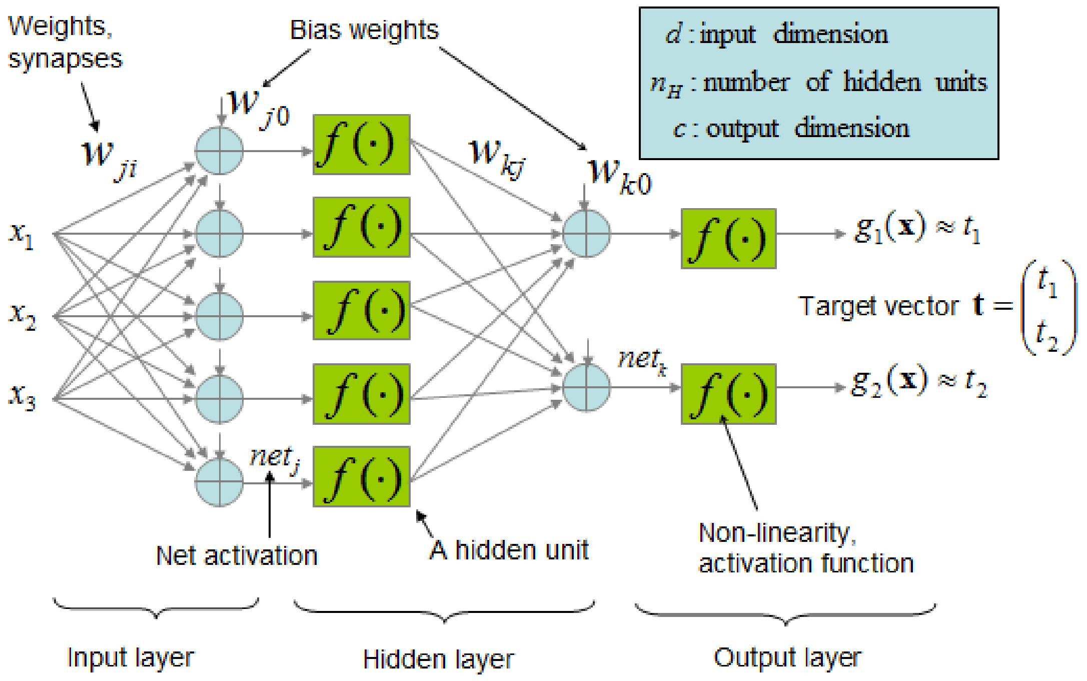


Figure 8: Feed-Forward Neural Networks [49]

Using FFNNs for forecasting consists of having the inputs to the network as past observations of the data and the output as the predictions of the future value of the series. In this sense, a neural network with at least one hidden layer is functionally equivalent to a nonlinear autoregressive model (NAR). The ANN aims to perform the following nonlinear function mapping from the past observations to the future value y_t

$$y_t = f\left(y_{t-1}, \dots, y_{t-p}; \boldsymbol{w}\right) + \varepsilon_t \qquad (11)$$

where $\boldsymbol{y_t}$ is the series output being forecasted, $(\boldsymbol{y_{t-1}}, \dots, \boldsymbol{y_{t-p}})$ are the past values of the series, $\boldsymbol{w}$ is a vector of weights or parameters and $\boldsymbol{\varepsilon_t}$ is the residual . Figure 9 (a) illustrates an example of forecasting with artificial neural networks and Figure 9 (b) shows a typical non-linear autoregressive neural network for forecasting univariate time series.

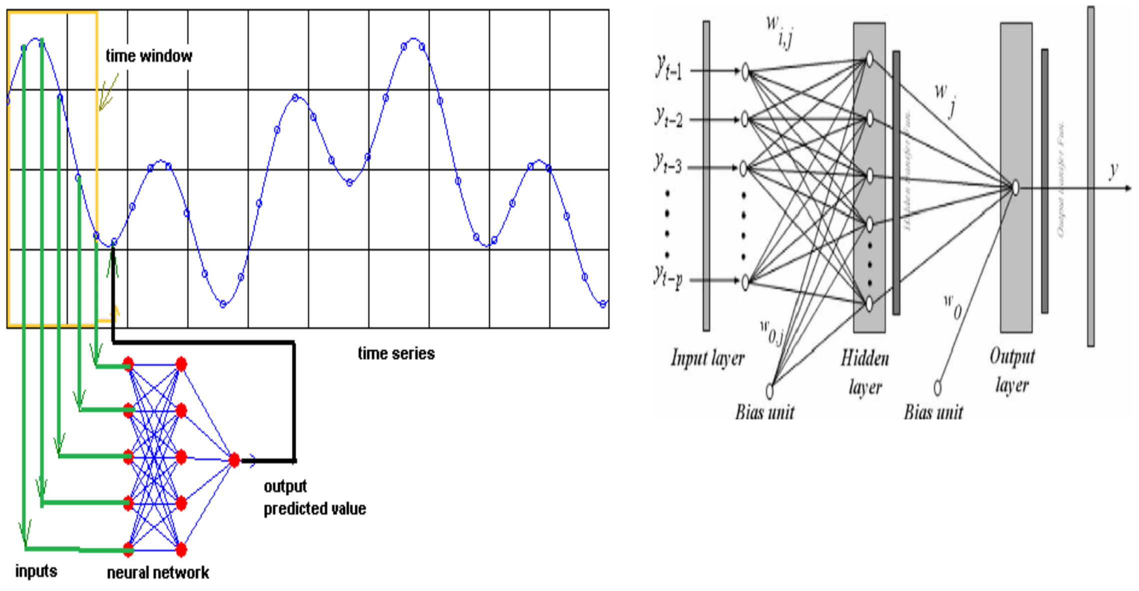


Figure 9: (a) Forecasting with ANNs reprinted from [44], (b) Typical non-linear autoregressive neural network for forecasting time series reprinted from [45].

Another type of FFNNs used for forecasting time series is the non-linear autoregressive with exogenous inputs neural network (NARX). The NARX **[50]** is a recurrent time delay network (TDNN) with a delay line (as in Figure 10) on the inputs and a feedback connection from the output to the inputs. A delay line is a lag of time in the inputs. The inputs like in the statistical VARMA model are a mixture of past values of the same time series, and past values of another independent time series. A NARX (see Figure 10) network can be mathematically represented as

$$y_t = f\left(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-q}; \boldsymbol{w}\right) + \varepsilon_t \quad (12)$$

where x_t *and* y_t denote, respectively, the input and the output of the model at discrete time t. The parameters p and q are memory delays, with $q < p$. The function f is a non-linear function of the input and output of the model. The predicted output y_t is regressed on the input values (exogenous) x_{t-k} and the output value y_{t-k}. Figure 10 shows the architecture of a NARX network with two hidden layer. This can be generalized to multiple inputs (N) and outputs (M).

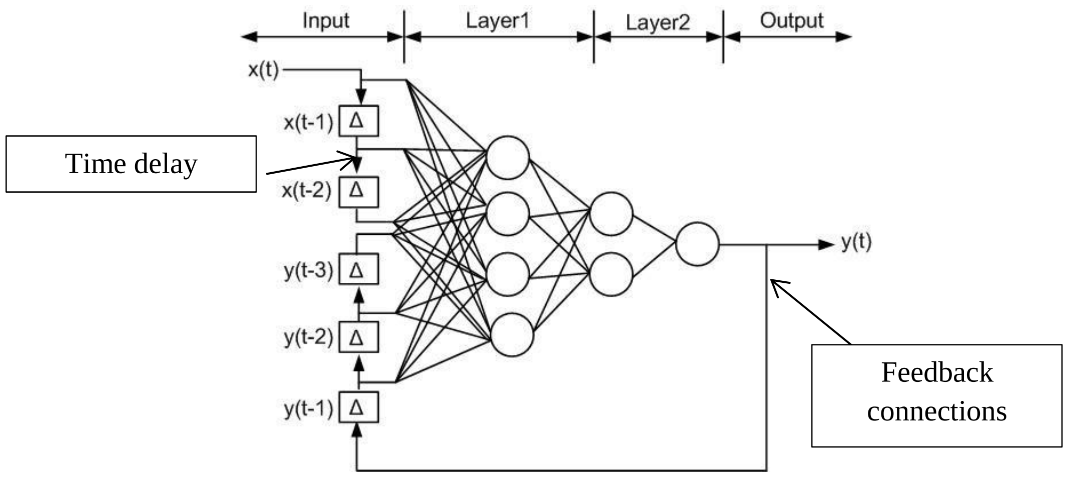


Figure 10: Typical architecture of NARX network with d_q delayed inputs and d_p delayed outputs [51]

Standard RNNs are extremely slow to learn and although in theory they have state memory, in practice they cannot really learn long term dependencies [6]. The training will be discussed in section 2.3.4.

2.3.2 LSTM Model

The LSTM networks are special kind of RNNs developed in 1997 by Hochreiter & Schmidhuber [45] as a neural network architecture for processing long temporal sequences of data [52]. Previous to this development, standard RNNs even though in theory can use information with long sequences, in practice are limited to capture only few time steps ago. This problem was explored in depth by Hochreiter [53] in 1991 and Bengio et al. [54] in 1994 who found some fundamental reasons why it might be difficult. The glaring reason for this is the decay of the back-propagated error when the size of the time lag between relevant information increases. This is called the "vanishing gradient" problem. LSTM were developed to overcome the problem of the vanishing gradient in RNNs by learning tasks involving long term dependencies [55]. LSTM has the particularity of having long short-term memory blocks which are constituted of memory cell units that are able to remember the value of a state for an arbitrary long time, as well as three different gate units that can learn to keep, utilize, or destroy a state when appropriate.

Although LSTMs are able to learn long term dependencies, they are extremely slow to train, often taking millions of iterations [8].

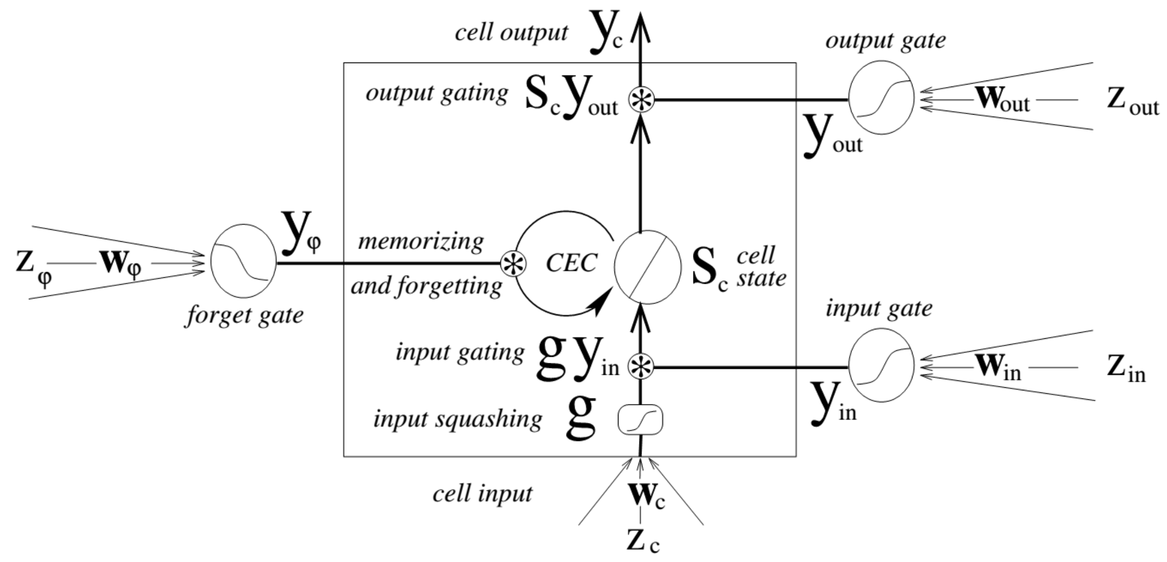


Figure 10: Example of LSTM memory block architecture reprinted from [7]

2.3.3 ESN Model

ESN is one of the key reservoir computing models. It has emerged in the last decade as an alternative to gradient descent based algorithms for training recurrent neural networks such as back-propagation through time (BPTT) and LSTM [12]. The reservoir (hidden layer) is a dynamical system which consists of a collection of recurrent neurons with randomly generated weights. The inputs feed the reservoir influencing the dynamics of the reservoir in a higher dimension. The training process consists of learning the reservoir output to connections meaning a simple readout mechanism where the state of the reservoir is read and mapped to the desired output. As opposed to standard RNNs, the hidden layer (or reservoir) is not trained. The output can be trained using fast regression instead of a slow gradient descend iterative learning method. Conceptually simple, practical and yet computationally inexpensive, ESN require nevertheless some insight and experience to be successfully applied [56]. In particular, the initialization of the weights of the reservoir needs to be wisely set in order to produce a correct timed output. Figure 11 illustrates the basic schema of ESN.

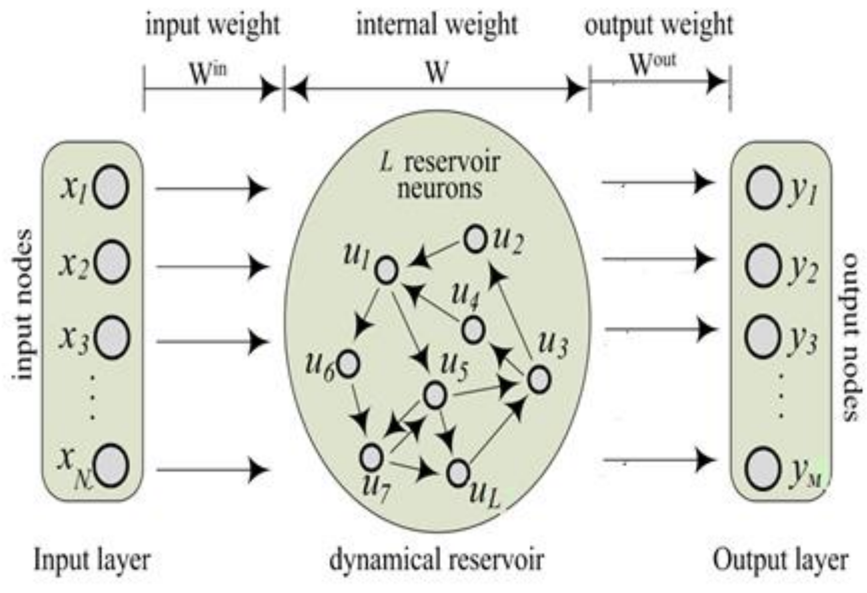


Figure 11: Basic schema of an Echo State Network modified from [57]

2.3.4 Training and Evaluation of ANNs forecasting models

There exist different methods for training neural networks based on the error cost function used to measure the error between the desired output and the forecasted output (see section 2.2.6). The most commonly used learning algorithm for training ANNs is the back-propagation (BP) algorithm proposed by Rumelhart, Hinton, & Williams in 1986 [58]. Back-propagation or "backward propagation of errors" is a supervised learning algorithm used in conjunction with an optimization method such as gradient descent. The training begins with random weights and the goal is to adjust them in order to minimize the forecasting error $E_t = (\hat{y}_t - y_t)^2$ for each time step. The algorithm consists of two passes. In the forward pass, a pattern (a set of input data) is processed through the network's layers and the outputs for each time step are computed. This returns the network final output vectors and the partial derivatives of the outputs with respect to the weights. In the backward pass, the gradient of the cost or error function (sum squared error for example) with respect to each weight is computed and the weights are modified along the downhill direction of the gradient (negative gradient) in order to minimize the error. This is repeated until the error stop reducing significantly.

Take, for example a univariate network forecast $\hat{y}_t = f(y_{t-1}, y_{t-2}, \dots, y_{t-k}, x_t, x_{t-1}, x_{t-2}, \dots, x_{t-k}; \boldsymbol{w})$ where f is the network. To minimize the error, the gradients with respect to $\boldsymbol{w}$ are computed for each time step. Since the error is the difference between the actual output $\hat{y}_j$ and the desired output y_j, and since $\hat{y}_j$ depends on the weights, these can be adjusted in order to minimize the error. Finally, gradient descent only requires the use of each weight-vector gradient generated for each time step forecast to update each of the corresponding weight-vector by gradient descent describe below:

$$\boldsymbol{w} = \boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}} E_t \qquad (13)$$

where $\boldsymbol{w}$ is a vector of weights, $\nabla_{\boldsymbol{w}} E_t$ is the gradient of the error E_t at time step t with respect to $\boldsymbol{w}$ and α is the learning rate. If y_t is a vector (multiple output), then $E_t = \frac{1}{M} \sum_{j=1}^{M} ((\hat{y}_t^{(j)} - y_t^{(j)})^2)$

Training can be done one time step at a time (on-line) or all changes can be summed up before updating the weights (this is called batch or offline learning). Multiple iteration of this process must be done until the error stops decreasing. To avoid overfitting, some data can be reserved to determine when to stop early (this is called early-stopping). For example, as the errors on those patterns start increasing, one should stop training. But these are not used in the evaluation of the gradient. Note that ESN as already mentioned is not subject to iteration methods and instead often relies on linear regression like most statistical methods.

After training the network and obtaining the appropriate weights, these later (weights) are tested on unseen data. This part is called testing. In machine learning, in order to assess performance of the network on unseen data, a cross-validation method is preferred [59]. Cross-validation is one of the most common approaches to automatically evaluate the performance of a model. In a nutshell, the cross-validation procedure allows for finding the best hyper-parameters and to measure how well the resulting estimated model is able to generalize the data. To reduce variance and bias on estimates forecast error, the data is split to allow different data to be used in training and testing. Cross-validation works by dividing the dataset into several subsets. Each one of these subsets is then used alternatively as a training set or as a testing set. The training data will be used to learn the weights or the parameters of the model while the testing set will validate how good or precise the weights are using only unseen data. Note that machine learning often aims at automated learning as opposed to statistical approach which aims at modeling knowledge about the task.

2.3.5 Applications

Artificial neural networks (ANNs) have been successfully applied to a number of time series prediction and modeling tasks. For example, Zhang et al. [60] did a survey of feed-forward ANNs applications in forecasting such as forecasting sunspot series, bankruptcy and business failure . Chen et al. [61] used ANNs to forecast the Taiwan stock index. Berardi et al. [62] studied bias and variance effect in neural network modeling for time series forecasting. A dynamic artificial neural network model for forecasting nonlinear processes such as Wolf's sunspot activity has been proposed by Ghiassi et al. [63] and Hosseini et al. [64] did a comparison of different feed-forward neural network for ECG signal diagnosis .

NARX is widely used for time series predictions such as forecasting peak air pollution level done by Pisoni et al. [65] using a polynomial NARX model. Jiang et al. [66] used NARX to predict chaotic time series. J.Maria et al. [67] showed that NARX network can successfully use its output feedback loop to improve its forecasting performance in complex time series prediction tasks such as long term time series for variable bit rate (VBR) video traffic time series. Andalib and Atry [68] used NARX for multiple step ahead electricity price forecasting. Xie et al. [69] proposed an time series prediction approach based on NARX.

Regarding LSTM network, Thierou et al. [70] worked on a bidirectional LSTM to predict the subcellular localization of proteins. Schmidhuber et al. [71] are interested on the vanishing gradient descent problem of recurrent networks and proposed an evolutionary LSTM model for sequence learning. Eck et al. used LSTM network for music composition [72]. LSTM were used to predict Reber’s Grammar [73]. The aim of this task was to predict the next character after a given sequence. Echo state network has been used for forecasting short term load electricity [74]. Decai et al. [75] used ESN for chaotic time series prediction.

ANNs have shown their intrinsic power in forecasting time series, sometimes far better than traditional statistic models [38]. Their advantages of handling and detecting nonlinear relationship among the data and dealing with noisy time series appear to explain their better forecasting performance [42]. Nonetheless ANNs have some limitations. Regarding FFNNs, their main drawback is the “lack of memory” meaning the inability to store information about past observations. As for RNNs, despite their great success in many natural language processing tasks, they have difficulties to learn long term dependencies i.e. dependencies between steps that are far apart. LSTM overcome to the vanishing gradient issue however the training still slows and computationally expensive. Regarding ESN, it requires some experience and insight to achieve a good performance notably the initial generation of the reservoir. But once set, the training is easier and fast. The major limitation of ANNs is that it is usually impossible to understand their solutions or what they are really looking for in the data.

2.4 Hybrid Forecasting Models

Using ANNs for time series forecasting has opened the door to new hybrid models. Hybrid models comprise a mixture of different methods from machine learning and standard statistical models. In the literature, diverse combinations of techniques have been proposed in order to overcome the limitations of single models. The idea behind the mixture of different models in forecasting is to build a strong architecture that will use the unique features of each model in order to capture different patterns in the data [52]. A good example is the combination of ARIMA and ANNs where an ARIMA process combines three different processes comprising an AR function regressed on past values of the process, MA function regressed on a purely random process, and an integrated (I) part to make the data series stationary by differencing them in order to deal with the non-stationary linear component while the neural network model deals with nonlinearity [38] and [71]. The literature in this domain has covered various applications. Khashei and Bijari [76] applied their hybrid model to exchange rate forecasting, and Mahdi et al., [77] self-organised multilayer perceptron neural network was inspired by the immune algorithm for financial forecasting. Luxhoj et al. [78] described a combined methodology using radial basis function networks (RBF) and the Box–Jenkins models. Armano et al. [79] presented a new hybrid approach that integrated artificial neural networks with genetic algorithms (GAs) and applied it to stock market forecast.

In recent years, several models combining fuzzy inference system (FIS) with neural networks and others methods have been proposed. A fuzzy inference system is the process of mapping a given input to an output using fuzzy logic (FL). FL is a problem-solving control system methodology that provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information. It can be designed either from expert knowledge or from data [80]. Castillo and Merlin [81] described the application of several neural network architectures to the problem of simulating and predicting the dynamic behavior of complex economic time series. Damousis and Dokopoulos [82] presented a fuzzy expert system that forecasts the wind speed at a wind energy conversion system and implement two genetic algorithms for comparing the fuzzy expert system. Potters and Negnevitsky [83] also describe in their paper an adaptive neuro-fuzzy inference system (ANFIS) for short term wind forecasting in Tasmania. Kasabov et al. [84] developed an online/offline evolving system for dynamic time series prediction.

Introduced by Boser, Guyon & Vapnik [48] in 1992, support vector machine (SVMs) are supervised learning algorithms used for classification and regression. There are several papers in the literature for time series prediction using SVMs such as Wu et al. [85], Zhang et al. [86], Hsu et al. [87], Dong et al. [88] and Pai et al. [89].

2.5 Summary

In this section, a brief overview of several models for forecasting time series was given. Used in various applications, all of them are important and bring their strength to the time forecasting domain. Traditional statistics models have shown their efficiency for linear low order models, while artificial neural networks have shown their capabilities to master nonlinear patterns. Hence, hybrid models that are a combination of artificial neural networks, statistics models, and sometimes other models are an emerging domain. However, most of them are only looking at learning the precise time series (as in Figure 2(a)) making it difficult for RNNs. Advantages and challenges of forecasting methods are summarized in the Table 2 below.

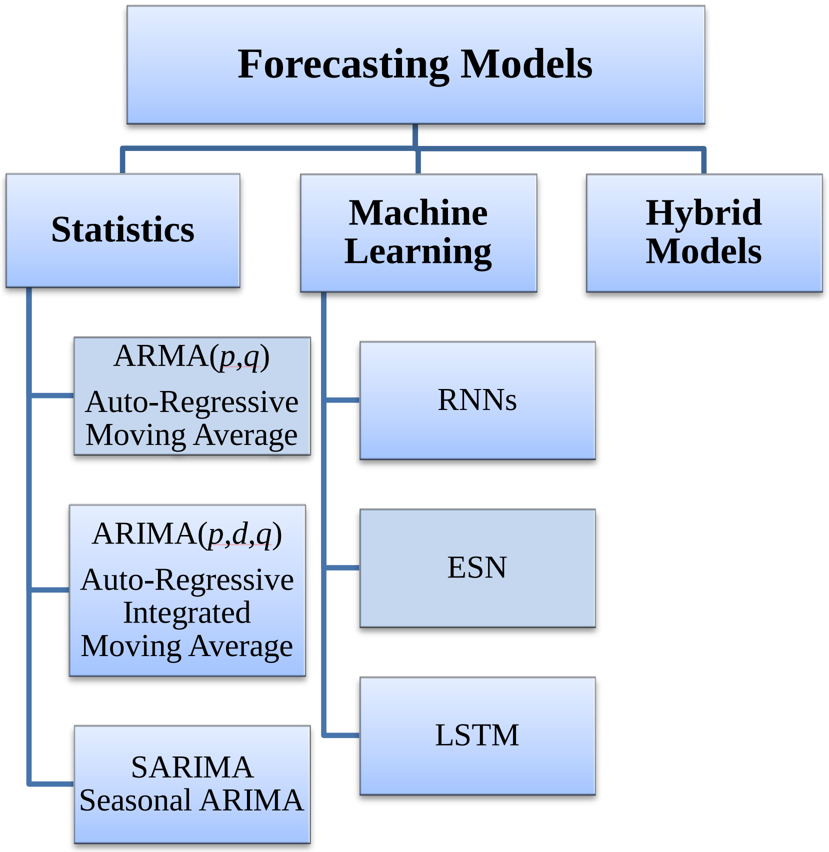

Forecasting Models
Statistics
Machine Learning
Hybrid Models
ARMA(p,q) Auto-Regressive Moving Average
ARIMA(p,d,q) Auto-Regressive Integrated Moving Average
SARIMA Seasonal ARIMA
RNNs
ESN
LSTM

Figure 12: Overview of time series forecasting models

Time Series Prediction Method	**Advantages**	**Challenges**
Standard Statistical Models	✓ Can be computationally efficient for low order models. ✓ Convergence guaranteed. ✓ Minimizes mean square error by design.	✓ Assumes linear, stationary processes. ✓ Can be computationally expensive for higher order models.
ANNs	✓ Not model dependent. ✓ Not dependent on linear, stationary processes. ✓ Can be computationally efficient for feed forward process. ✓ Capable of learning long term dependencies (LSTM). ✓ Can detect all possible, complex nonlinear relationships between input and outputs.	✓ Selection of free parameters usually calculated empirically. ✓ Not guaranteed to converge to optimal solution. ✓ Can be computationally expensive (training process). ✓ Neural networks have a "black box" nature. Therefore, errors within the complex network are difficult to target.

Table 2: Summary of advantages and challenges of classical and ANNs based time series prediction methods [90]

3 Problem Formulation

This section discusses the forecasting timing events problem being investigated and introduces the notation that will be used from this section until the end of this work. Note that from this section to the end of this work, the notation for time series index will be change from $\boldsymbol{x_t}$ to $\boldsymbol{x(t)}$ (and similarly, for the time series $\boldsymbol{y_t}$ and $\boldsymbol{z_t}$).

3.1 Forecasting the Time Remaining or the Precise Time Step of Events

Time series forecasting is difficult and predicting many steps ahead into the future is problematic because the larger the forecast horizon, the larger is the uncertainty. Unlike one-step ahead forecasting, multi-step ahead forecasting tasks are more challenging [91] since many issues arise such as accumulations of errors, and less accuracy [92]. The forecasting field has been influenced for a long time by linear statistical models; nevertheless, it becomes clear that linear statistical models are not adapted for some applications due to their linear limitations [18]. Machine learning algorithms on the other hand are computationally expensive when trying to learn long term dependencies notably event onsets. For example predicting the precise time step of instantaneous events (marked as a value of 1 in a series of 0) that are far apart can take millions of trials [7] .Yet, animals seem to learn the precise timing of events very easily by estimating the time remaining before particular events occurs [9], [10].

In the quest for simple, practical, fast and accurate learning algorithms, an online learning time series algorithm (TDDM) has been proposed in the literature. The idea behind this algorithm has been inspired by how an animal learns. F. Rivest et al [10] explain that the success of an animal learning the timing of events suggest that they are predicting events in a different way than we do in machine learning: “Animals are predicting when an event will occur instead of what event will occur” for each time step . The large majority of statistical and machine learning algorithms try to predict what event will occur at every time step. The proposed model suggests instead predicting when an event will occur.

This problem can then be reformulated to learn to predict the time remaining before an event occurs, instead of its occurrence at a particular time step. In order to predict event timing as an event stream or as time remaining, a preprocessing of the datasets must be performed. The input time series first needs to be transformed into a binary series and the target time series transformed into an event time series or into a time remaining time series. First, the original time series is transformed into a binary time series as input to the system. For each time series variable $x(t)$: $x(t) = 1$ when a stimulus is observed, and $x(t) = 0$

otherwise. For example, a data set of music could be transformed this way such that a sample is “1” if a note of music is heard or “0” otherwise. Then, a special binary time series called an event time series $z(t)$ is generated, such that for each time step, $z(t) = 1$ when the event occurs, and $z(t) = 0$ otherwise. As an example, a financial time series could be transformed into a binary event time series considering significant stock price increases as events. Finally, the event time series is used to generate a remaining time series $y(t)$ corresponding to the time remaining before the next event; i.e., the next time step *t* such that $z(t) = 1$. Figure 2(b) and (c) shows an example of a time series with its event time series and time remaining time series.

This thesis compares the performance of three offline algorithms that use regression (VARMA for statistical method, ESN for neural networks and TDDM for animal model) on two different formulations of the forecasting event timing problem that are forecasting binary stream of instantaneous events and forecasting the time remaining before instantaneous events occur. The TDDM algorithm will be adapted for offline learning in section 4. The goal is to evaluate both the problem formulated and the algorithms’ performances.

4 TDDM (Time-Adaptive Drift-Diffusion Model)

This chapter introduces the online multivariate time-adaptive drift-diffusion model algorithm proposed by F. Rivest et al. [1] from animal learning theory. Next, the developed offline version of the TDDM algorithm is described. This allows TDDM to work as a linear regression based model in the same way as VARMA and ESN.

4.1 TDDM Description

The online multivariate time-adaptive drift-diffusion model [1] algorithm is derived from drift-diffusion models (DDM). It has been inspired by how animals learn. In fact, research has shown that animals could learn the timing between events very easily and rapidly with a number of trials independent of the time-dependencies [9] contrary to machine learning algorithms which require a huge number of trials, at least quadratic with the time-dependencies scale (e.g. [8]). For each specific event, a drift-diffusion process works by accumulating evidence about when that specific event will occur and by learning the event-rate of each input stimulus. The learning is performed through a set of weights and accumulation processes. The equations of the algorithm can be described as follows:

Given an N-dimensional *binary time series* $\{\boldsymbol{x}(t) = (x_1(t), \dots, x_N(t))\}$ of the observable stimuli, an M-dimensional *event time series* $\{\boldsymbol{z}(t) = (z_1(t), \dots, z_M(t))\}$ representing the occurrence of events such that a sample $z_j(t) = 1$ only when an event of type j is occurring at time t and an M-dimensional *time remaining time series* $\{\boldsymbol{y}(t) = (y_1(t), \dots, y_M(t))\}$. Let $t_{j,k-1}$ be the timing of the most recent $(k-1)^{th}$ event of type j. Then, for each output j, there is a weight (row) vector $\boldsymbol{w}_j$, such that the accumulator $\Phi_j(t)$ accumulates evidences until time step t:

$$\Phi_j(t) = \sum_{\tau=t_{j,k-1}}^{t} \boldsymbol{w_j} \cdot \boldsymbol{x}(\tau) \qquad (14)$$

Assuming the accumulator should reach a threshold (say 1, without loss of generality) about when the event occurs, then one can estimate the remaining time $\hat{y}_j(t)$ using the equation:

$$\hat{y}_j(t) = \frac{1-\Phi_j(t)}{\boldsymbol{w}_j \cdot \boldsymbol{x}(t)} \qquad (15)$$

For each output stream j, assuming the accumulators in j are reset as soon as an event occurs on stream j, an estimation of the duration of when a given stimulus has been observed since the last event at time $t_{j,k-1}$ could be made using the formula $a_{j,i}(t) = \phi_j(t)/w_{j,i}$. Given durations $\boldsymbol{a_{j,i}}(t) = \left[(a_{j,1}(t), \ldots, a_{j,N}(t))\right]$, the appropriate weights would be one such that

$$\Phi_j(t_{j,k}) = \boldsymbol{w_j}\boldsymbol{a_j^T}(t_{j,k}) = 1 \qquad (16)$$

at the time of the next event $t_{j,k}$ of type j. This constraint generates at time $t_{j,k}$ of type an $(N-1)$-dimensional hyperplane of possible solutions for $\boldsymbol{w}_j$. One can move $\boldsymbol{w}_j$ perpendicularly toward that hyperplane using

$$\boldsymbol{w}_j = \boldsymbol{w}_j - \alpha d \boldsymbol{a}_j(t_{j,k}) \qquad (17)$$

where

$$d = \frac{(\boldsymbol{w}_j \cdot \boldsymbol{a}_j{}^T(t_{j,k}) - 1)}{\boldsymbol{a}_j(t_{j,k}) \cdot \boldsymbol{a}_j{}^T(t_{j,k})} \qquad (18)$$

is the distance to the solution hyperplane, and $0 < \alpha < 1$ is a small learning rate.

This model has only a single hyper-parameter α and has $O(NM)$ parameters and requires $O(NM)$ memory for the accumulation process.

The offline TDDM algorithm is developed to use a linear regression method to improve all predictions $\hat{y}_j(t)$ made at each time step instead of correcting only the last prediction as in the online model. Given the accumulation $a_{j,i}(t) = \sum_{\tau=t_{k-1}}^{t} x_i(\tau)$, where *k-1* is the last event of type *j* observed, the estimated remaining time $\hat{y}_j(t)$ can be written using equation (14) as:

$$\hat{y}_j(t) = \frac{1 - \sum_{i=1}^{N} w_{j,i} . a_{j,i}(t)}{\sum_{i=1}^{N} w_{j,i} \cdot x_i(t)} \qquad (19)$$

A linear regression method can be applied independently for each event *j* using every available time step *t* to find the weights that can best fit the data. Equation (19) has been transformed into a set of linear equations for given target values $y_j(t)$ for each time step *t* in replacing $\hat{y}_j(t)$ by $y_j(t)$.

$$\sum_{i=1}^{N} w_{j,i} \left[\hat{y}_j(t) x_i(t) + a_{j,i}(t)\right] = 1 \qquad (20)$$

$$\left[y_j(t)x_1(t) + a_1(t), \quad y_j(t)x_2(t) + a_2(t), \cdots\cdots, y_j(t)x_N(t) + a_N(t)\right] * \boldsymbol{w_j^T} = 1 \qquad (21)$$

with j=1 to *M* outputs. The set of regression equations (21) for all available time steps *t* for a given stream *j* minimizes the error: $(\% \text{ of remaining time evidences } + \% \text{ of elapsed time evidences } - 1)^2$.

4.2 Summary

In this section we presented the online TDDM algorithm and developed an offline TDDM algorithm. These algorithms do not minimize the SSE cost function to compute the forecast at each time step; instead they minimize the errors in the percentage of remaining time added to the percentage of elapsed time to 1. The predictions are obtained through a set of weights and accumulation processes. The online version has a single hyper parameter α. The offline version uses a linear regression method to estimate the weights and does not need any hyper-parameter. The main drawback of TDDM for both the online and the offline version [1] is that unlike ANNs, its mathematical form may not allow it to discriminate cases where two events of the same type are preceded by distinct temporal patterns. But at least this expanded version of TDDM is trained using the same process as VARMA and ESN making it more comparable. From now on, the TDDM that will be used and discussed in the upcoming section is the TDDM offline unless it is specified otherwise.

5 Experiment Procedure and Results

This chapter includes a description of the three real world datasets used, the methods used for evaluating the three algorithms on the tasks of predicting timing event, and lastly, the experiments results. The experiments were devised for two tasks. First the prediction of binary instantaneous event streams at each time step and second the prediction of the time remaining before the events occur by the three algorithms. All the experiments, the datasets transformation and the preprocessing are done in MATLAB. Each algorithm is represented in oriented object MATLAB class.

5.1 Data sets

All the datasets (music, finance and heartbeats) used for the experiments are composed of sets of multivariate time series. The forecasting timing event approach suggests a transformation of the datasets into binary time series $\boldsymbol{x}(\boldsymbol{t})$, event time series $\boldsymbol{z}(\boldsymbol{t})$ and time remaining time series $\boldsymbol{y}(\boldsymbol{t})$ in order to predict the event itself; in other words, to estimate the time remaining before an event occurs. The transformations of the datasets raise issues regarding some unknown remaining time. These missing values came from the last portion of the time series where no more events happen. The remaining time would be undefined, hence the use of NaN value (a notation that is used in MATLAB to denote Not a Number). Yet, linear regression methods do not work with data expressed as NaN for the simple reason that NaN data invalidates any arithmetic operation. In order to overcome this issue, all the variables of type $\boldsymbol{x}(\boldsymbol{t}), \boldsymbol{z}(\boldsymbol{t})$ and $\boldsymbol{y}(\boldsymbol{t})$ of each dataset were trimmed at the first NaN.

5.1.1 Music

The music dataset Bach Chorales from the UCI Machine Learning Repository **[93]** is composed of 100 chorales (musical pieces). Each piece of music consists of approximately eight bars using notes from C4 (midi 60) to G5 (midi 79) and each music note is a time series. The three main attributes of the data are the following: a start-time which shows the starting point for each of the 20 music notes; MIDI-Pitch numbers from C4 (midi 60) to G5 (midi 79); and a duration that indicates the time during which the note was played. For predicting the timing of a single note, the inputs for the algorithm are the binary time series $\boldsymbol{x}(\boldsymbol{t})$ indicating whether each note is on or off. The event time series $\boldsymbol{z}(\boldsymbol{t})$ are made of the onsets and offsets of each note. Note onsets were extracted and their indices were computed assuming sampling at 1/16th of a beat. As mentioned above, notes that never occur in a piece were removed from the $\boldsymbol{x(t)}$, $\boldsymbol{z(t)}$ and $\boldsymbol{y(t)}$ time series for that piece reducing the input dimension *N*. Trimming at the first NaN in each piece generates a huge loss of data for the music dataset in many time series up to 51% of their initial lengths. In the worst case, some time series lost 90% of their initial length. In order to reduce the amount of data loss, new pieces were made by repeating twice the same piece before trimming the end as if the music pieces were played twice. The previous piece and its repetition were separated by free bars added at the beginning and the end of piece. Repeating the piece guaranteed that each note in the first half at least reoccurs once in the second half. Figure 13 shows an example of the transformation of one piece of music in binary, events and time remaining time series as well as the new piece generated by the process of repeating the piece see Figure 14.

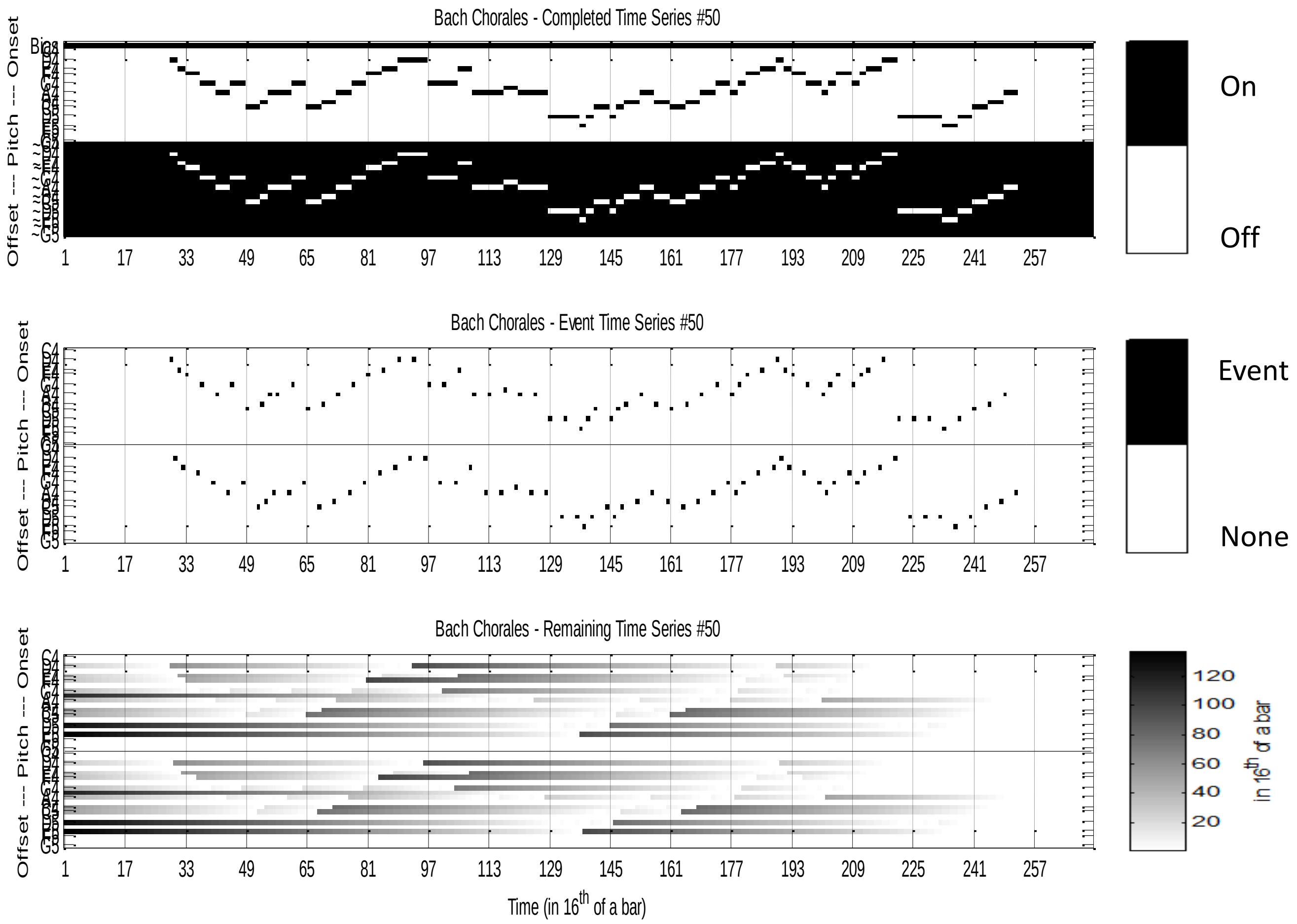


Figure 13: Time series transformation respectively in binary time series, event time series and time remaining time series for the multivariate Bach Chorales 50.

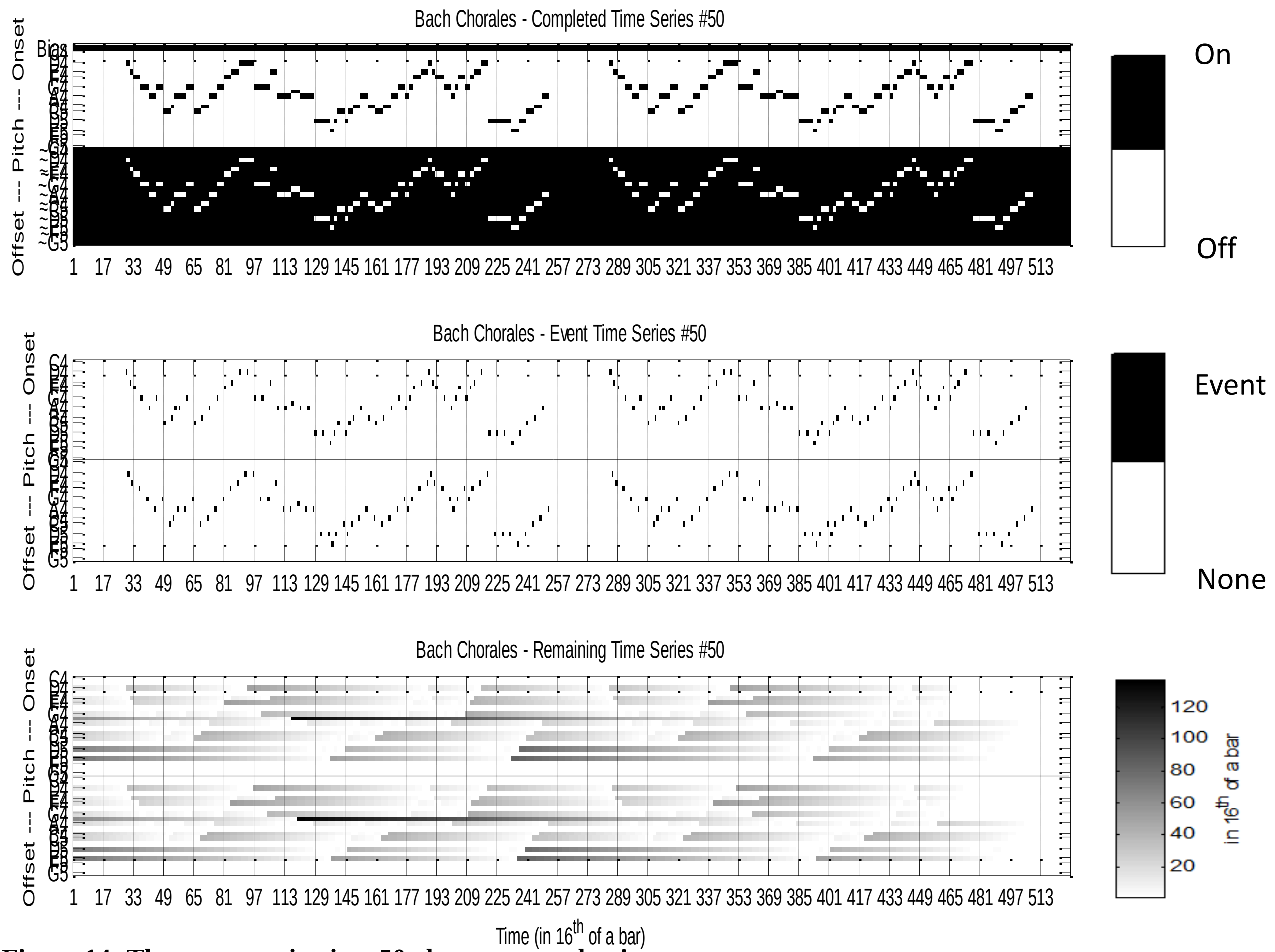


Figure 14: The same music piece 50 above repeated twice.

5.1.2 Finance

The NASDAQ financial dataset [94] includes stocks prices from 100 different companies from April 1^{st} 2011 to March 31 2014. Each stock represents a time series. For each day t of each stock time series $\{s_j\}$, the mean $\mu_{s_j}(t)$ and standard deviation $\sigma_{s_j}(t)$ was calculated for the previous fourteen trade days ($\mu_{s_j}(t) = \mu_{s_j}(14)$ and $\sigma_{s_j}(t) = \sigma_{s_j}(14)\ \forall\, t < 14$). Then, the entire series was transformed into two input binary time series: one series indicates if the closing price $x_j(t) = s_j(t) > \mu_j(t)$ was above average and the other if it was below $x_j(t) = s_j(t) \leq \mu_{s_j}(t)$. Finally, in order to detect a significant increase in stock values for representing the events, a threshold of $\theta_{s_j}(t) = \mu_{s_j}(t) + 0.7\sigma_{s_j}(t)$ was assigned. Therefore, event time series is obtained by generating events on days when the closing price is crossing this threshold ($\theta_{s_j}(t) = \mu_{s_j}(t) + 0.7\sigma_{s_j}(t)$) as proposed in [1]. The data set has five attributes: the company name, the stock symbol, the closing price for a particular stock, the company sector and its corresponding number. The companies were grouped by sector or category with each sector forming one multivariate time series, for a total of 11 time series. Figure 15 shows an example of the transformation of the NASDAQ category #5 in binary, events and time remaining time series.

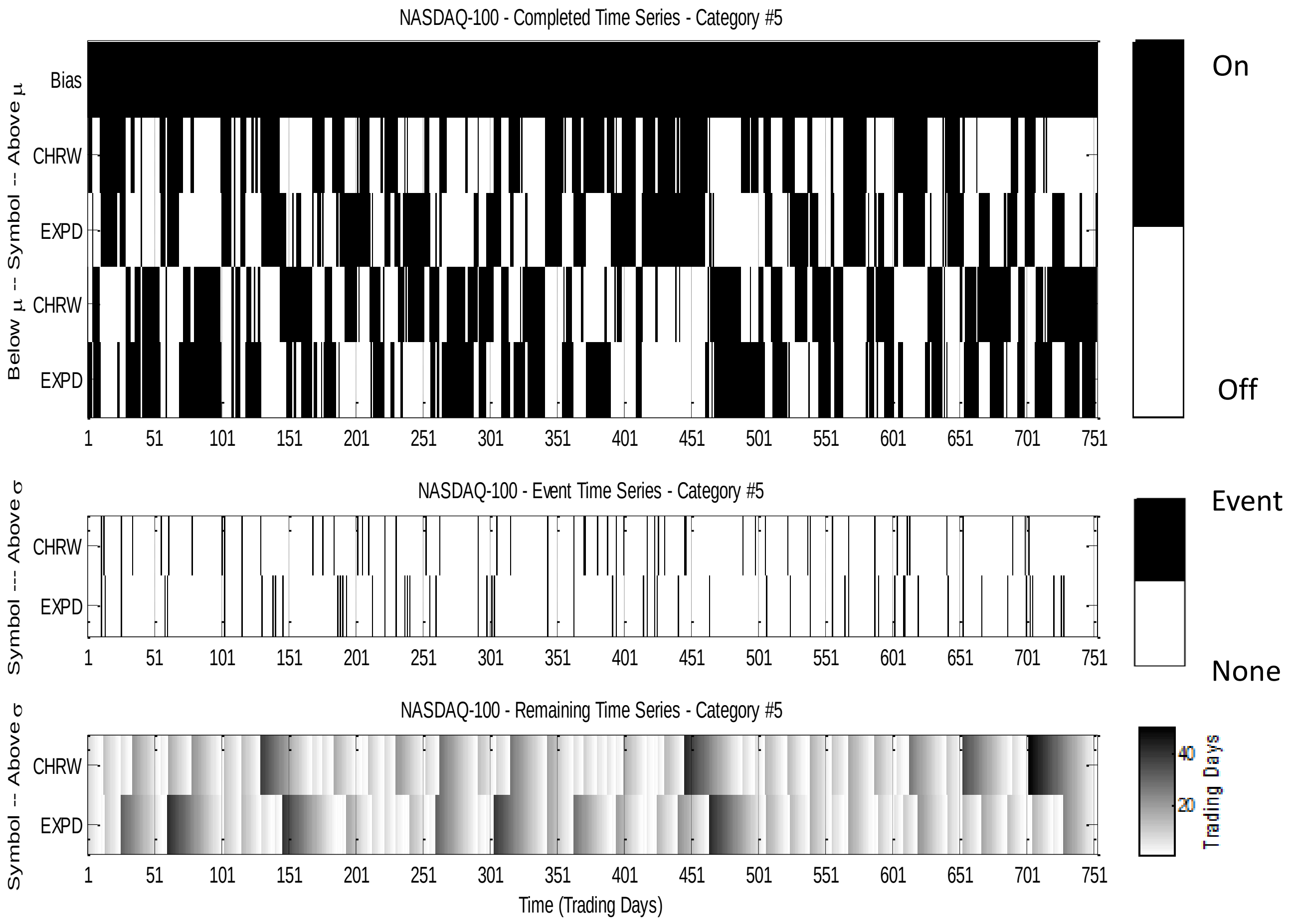


Figure 15: Time series transformation respectively in binary time series, event time series and time remaining time series for the multivariate stock sector 5 taken from 2011-2014.

5.1.3 Heart-beats

The MIT-BIH Arrhythmia database [95], [96] and [97] is composed of forty eight electrocardiograms (ECG) digitized at 360 Hz listed from different patients who had arrhythmias. Each heartbeat in the database has been converted into a binary time series: one if a heartbeat occurred at the time step, and zero otherwise. From the 48 records, 16 were selected because these records had less than ten abnormal beats in any 20 second window [98]: 100, 101, 103, 105, 108, 112, 113, 114, 115, 116, 117, 121, 122, 123, 215, and 230. For each individual heartbeat time series, the only input is a bias, and the events are the heartbeats themselves, with the output being the estimated time until the next heartbeat. The heartbeat dataset contains 16 univariate time series.

5.2 Methods

The three algorithms on which we based our experiment namely ESN, TDDM and VARMA are trained in an offline manner by performing a linear regression method on each of them. The offline or batch learning technique allows access to the entire training set and for the algorithms to compute the weights at once. The linear regression equation is expressed by

$$\boldsymbol{u} = V\boldsymbol{b} + \boldsymbol{\varepsilon} \qquad (21)$$

where $\boldsymbol{u}$ is the outputs linear regression vector, V is the linear regression inputs matrix, $\boldsymbol{b}$ is the vector of weights or coefficients and ε is the residual. From equation (21), one can derive the coefficients $\boldsymbol{b}$ vector as follows:

$$V^T\boldsymbol{u} = V^T V\boldsymbol{b} + \boldsymbol{\varepsilon} \qquad (22)$$

$$(V^T V)^{-1}\, V^T\boldsymbol{u} = \boldsymbol{b} + \boldsymbol{\varepsilon} \qquad (23)$$

$$\boldsymbol{b} \cong (V^T V)^{-1} V^T\boldsymbol{u} \qquad (24)$$

The training and testing errors are obtained by computing the root mean squared error (RMSE) for both experiments which measures the root squares of averaged errors over every output and every time step for each multivariate set of time series in the folds.

$$\mathrm{MSE} = \frac{1}{nT}\sum_{j=1}^{n}\sum_{t=1}^{T}\big(\hat{y}_j(t) - y_j(t)\big)^2 \qquad \text{with} \quad \mathrm{RMSE} = \sqrt{\mathrm{MSE}} \quad (25)$$

where n is the number of output variables in the time series, T is the length of time series or the number of time steps in the series, $\hat{y}_j(t)$ the predicted output and $y_j(t)$ the real output. For example, in a music piece with 10 notes and tested on 20 time steps, $nT = 200$.

The three algorithms were implemented in MATLAB and run independently on every multivariate time series. Regarding VARMA and TDDM, the \ operator has been used in order to compute the weights using linear regression. The \ operator performs a least-squares regression. The least squares fitting technique is the simplest and the most commonly applied form of linear regression. It provides a solution to the problem by finding the best fitting weights for a set of data. As for ESN, a MATLAB ESN toolbox written by H. Jaeger et al. [99] was used.

5.2.1 Procedure of Experiment

In order to find the best hyper-parameters for each algorithm, we performed a K-folds (with K=10) cross-validation procedure on the time series data. Each fold depending on the dataset contains a set of multivariate time series. K-fold cross-validation (see section 3) is a technique to assess the overall performance of an algorithm given a set of possible hyper-parameter values, and to evaluate the results of the model on unseen data. The K-folds cross-validation procedure used in this research is described as follows: It consists first of randomly partitioning the original dataset into 10 subsets or folds. Then nine folds are taken for training on the first 75% of the series and testing on the last 25% in order to find the best hyper-parameters. Once the best hyper-parameters are found, they are used on the last fold for final training and testing on those remaining and unseen time series. The cross-validation (see Table 3) technique allows us to evaluate the overall predictive performance of each algorithm as if the model was trained on new unseen time series using the best hyper-parameters.

Therefore, the music dataset of 100 multivariate time series will be divided in 10 folds with K=10 and each fold contains 10 multivariate time series. The NASDAQ dataset is composed of 11 multivariate time series with each fold containing at least one multivariate time series and finally the heartbeat dataset contains 16 univariate time series with each fold containing also at least one univariate time series. The hyper-parameters of VARMA (p,q) are the order p and q of the VARMA model, the ESN hyper-parameter is the number of internal neurons of the network, and the TDDM algorithm has no hyper-parameter. The range of (p,q) used for VARMA is (p,q)$\in [1..15] \times [1..15]$ for a total of 225 hyper-parameters values. As for ESN, the following vector of hyper-parameters was used: $\boldsymbol{v} = [10,20,30,40,50,60,70,90,100]$. Regarding the heartbeat dataset, it uses less hyper-parameters on the VARMA algorithm since the simulations were very slow. The couple of VARMA hyper-parameters chose for the heartbeats are (p,q) = (1,1), (1,15),(15,1),(15,15).

The procedure below is a general cross-validation procedure that we used for the three algorithms.

```
Given a dataset $D$ of all multivariate time series :
Divide D into K=10 folds or subsets.
Let's denote $D_i$ a subset at fold $i$, for $i = 1, \dots, 10$ such that $\cup D_i = D$ and $\cap D_i = \emptyset$
    • For each fold $i$
        o T-test (testing set) =$D_i$ ;
        o T-train (training set) = $D \backslash D_i$
            ▪ For each multivariate time series in T-train :
                • For each possible value of hyper-parameter
                    o Find the best weights using the first 75% of the time series
                    o Compute the predictions using the weights from above using the last 25% of the time series
                    o Compute the testing errors
            ▪ Choose the best hyper-parameters based on the predictions errors
            ▪ For each multivariate time series in T-test :
                • For the best hyper-parameter
                    o Find the best weights using the last 75% of the time series
                    o Compute the predictions using the weights from above using the last 25% of the time series
                    o Compute the testing errors
```

Table 3: The Cross-Validation Procedure

5.3 Results

Two sets of experiments were devised in this research. In the first experiment, the algorithms tried to predict binary instantaneous event streams at each time step i.e. to learn to predict the value (1) at the time step the event shall occur. In this case, the events correspond to: the note's onsets and offsets for Bach Chorales, the next significant jump in price for the stocks and the next heartbeat for MIT-BIH heartbeats database. The second experiment, as suggested by the forecasting timing event concept discussed in section 3, the algorithms learned to predict the time remaining before the events occur. Hence, for the music notes, the algorithms will learn to predict the time remaining before each note's onsets and offsets occur. In the case of the stock market data, the algorithms will predict the time remaining before the next significant jump in price and lastly in the case of heartbeats, the algorithm will learn to predict the time remaining before the next heartbeat.

5.3.1 Experiment I:

All the algorithms receive as inputs the binary time series $\boldsymbol{x}(\boldsymbol{t})$ and as target output the events time series $\boldsymbol{z}(\boldsymbol{t})$ they should predict.

The VARMA algorithm estimates its predictions using multiple linear regressions applied on the model VARMA as stated earlier in equation (9) in section 2.2.3. Note that the time series $\boldsymbol{y}(\boldsymbol{t})$ could be replaced by $\boldsymbol{z}(\boldsymbol{t})$ in the equation below in order to apply the VARMA model for predicting instantaneous events stream.

$$\hat{\boldsymbol{z}}(t) = \sum_{k=1}^{p} \mathrm{A}(k)\boldsymbol{z}(t-k) + \sum_{k=0}^{q} \mathrm{B}(k)\,\boldsymbol{x}(t-k) \qquad (26)$$

TDDM for its part estimates the predictions using the weights and the binary inputs series as stated in the equation (27). An event occurs when the accumulator Φ_j crosses a threshold:

$$\hat{z}_j(t) = 1 \; if \;\; \Phi_j(t) > 1 \; and \;\; \Phi_j(t-1) \leq 1 \qquad (27)$$

Lastly, the ESN algorithm receives as input the binary series $\boldsymbol{x}(\boldsymbol{t})$ and as target output the events time series $\boldsymbol{z}(\boldsymbol{t})$ and uses regression to forecast the next event at each time step.

Table 4 below shows the overall algorithms performance of each algorithm for Experiment I by calculating the average test root mean squared error over the 10 cross-validation folds as well as the majority vote results. The majority vote is a decision method that always chooses the alternatives which have a majority. In this case, because the majority of the observations (data) are “0”, the majority vote would always say 0 which means no events are occurring. For this experiment, the best hyper-parameters of VARMA on the Bach Chorales, the stock data, and the heartbeats are respectively (1,1), (1,2) and (15,15). As for the ESN algorithm, the best hyper-parameters are respectively 10 internal neurons for both stock and Bach Chorales datasets and between 50 to 80 internal neurons in general for the heartbeat dataset.

Algorithms	**ESN**	**TDDM**	**VARMA**	**Majority vote**
Bach Chorales	0.14 ± 0.04	0.20 ± 0.01	0.12 ± 0.01	0.16 ± 0.01
Stocks	0.25 ± 0.02	0.28 ± 0.02	0.26 ± 0.03	0.27 ± 0.02
Heartbeat	0.057 ± 0.005	0.065 ± 0.008	0.057 ± 0.005	0.057 ± 0.005

Table 4: Averaged test root mean squared error per prediction for all the algorithms on the task of predicting events onsets.

In order to know if the three algorithms errors’ in Table 4 are significantly different from each other, an analysis of variance (ANOVA) has been made. ANOVA is used to determine whether there are any significant differences between the means of three or more independent groups. The ANOVA for the Bach Chorales had a *p* value of 7.9x10^{-08}. A post-hoc Scheffe test (with α=0.05) reveals that all 4 means are significantly different from each other.

The ANOVA on stocks gives a *p* value equal to 0.036 and the post-hoc Scheffe test reveals that TDDM is worst than VARMA and ESN, however it is not worst than majority vote. Only ESN is significantly better than majority vote.

For heartbeat, the analysis of variance gives *p* value of 0.0056 and the post-hoc Scheffe test shows that ESN and VARMA errors are likely the same, but TDDM is significantly worst than the two other .

Because only binary values are used for this experiment, all the average errors in the table are relatively small. One can try to draw a conclusion based on these results but the errors only do not give much information on the performance of the algorithms on forecasting “1” i.e the event at a precise time step; hence the importance of the majority vote results. For each dataset all the errors are close to the majority vote errors (Table 4). This shows that all the algorithms are seldom predicting “1” where they should be as shown in Figure 16, Figure 17, Figure 18 and Figure 19 below.

Figure 16 below shows the forecasts of the three algorithms on the Bach Chorales dataset for the music notes #2 from music piece 29. In the testing part, we observe an attempt by the three algorithms to predict the events. VARMA and ESN are able to find where the events occur but are not able to predict them entirely i.e. their predictions are not above 0.5 and barely reach 1. As for TDDM it misses some events but it was able to predict some of them just before they happened. In Figure 17 below, the ESN predictions are higher than “1” in the testing part.

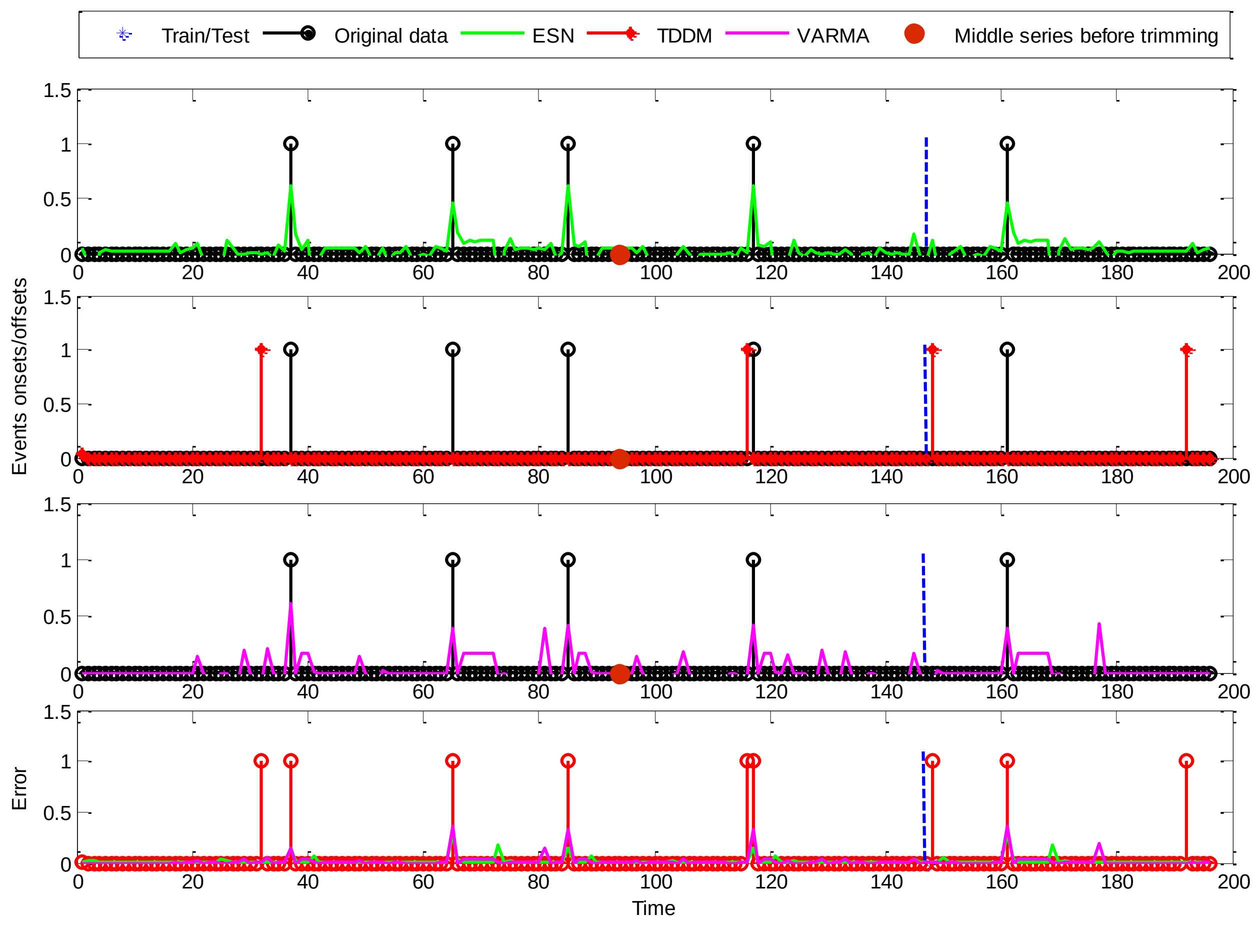


Figure 16: ESN, TDDM and VARMA algorithms prediction of events of note 2 from Bach Chorales 29. The train/test set separation is indicated by a blue marker. The middle of the music piece before trimming, i.e. the repetition point is illustrated by a brown dot.

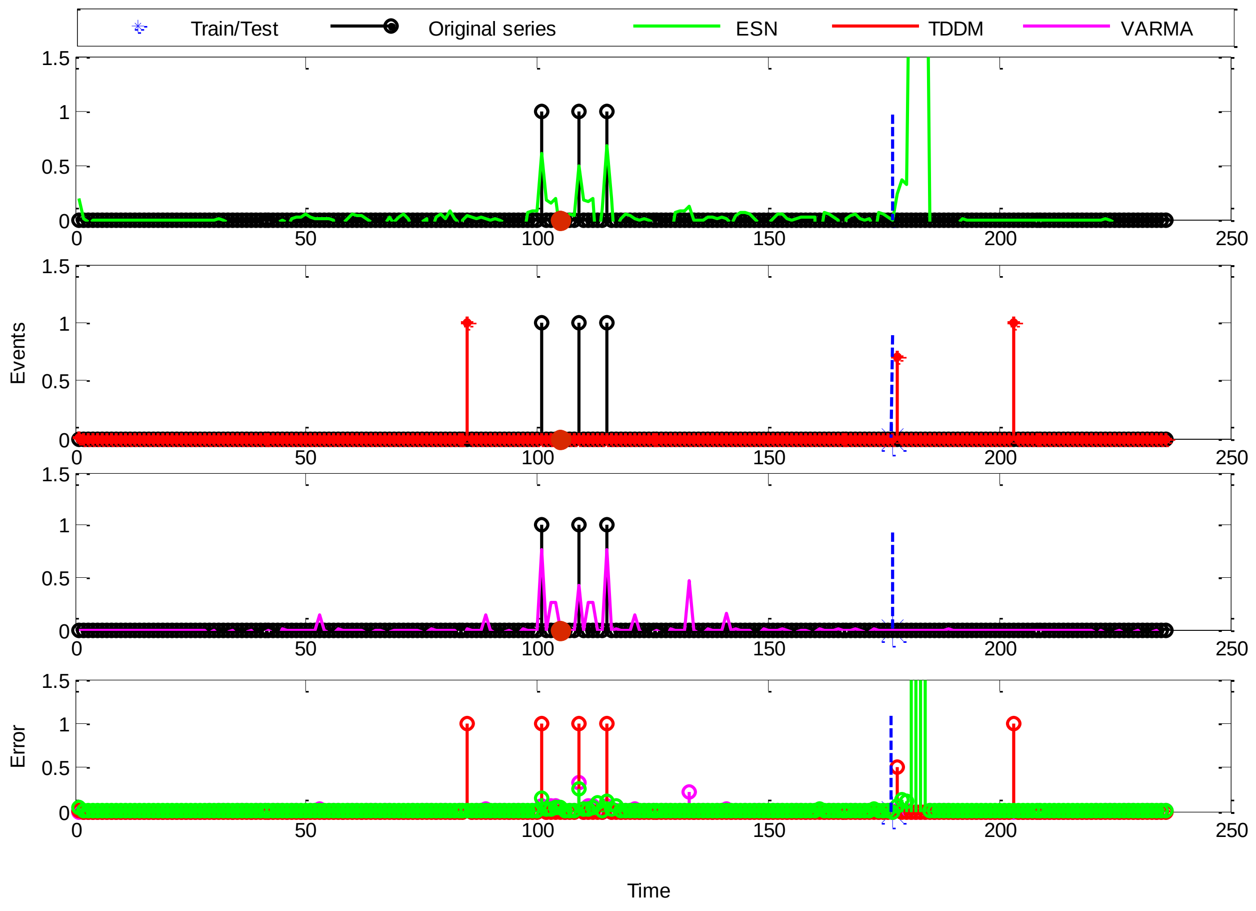


Figure 17: ESN, TDDM and VARMA algorithms prediction of events of note 11 from Bach Chorales 4. The train/test set separation is indicated by a blue marker. The middle of the music piece before trimming, i.e. the repetition point is illustrated by a brown dot.

For the stocks dataset (see Figure 18 below), the predictions are worst for VARMA and ESN which are in most of the cases predicting zeros meaning no events. As for TDDM, it predicts the events just before they occur but also misses generally some.

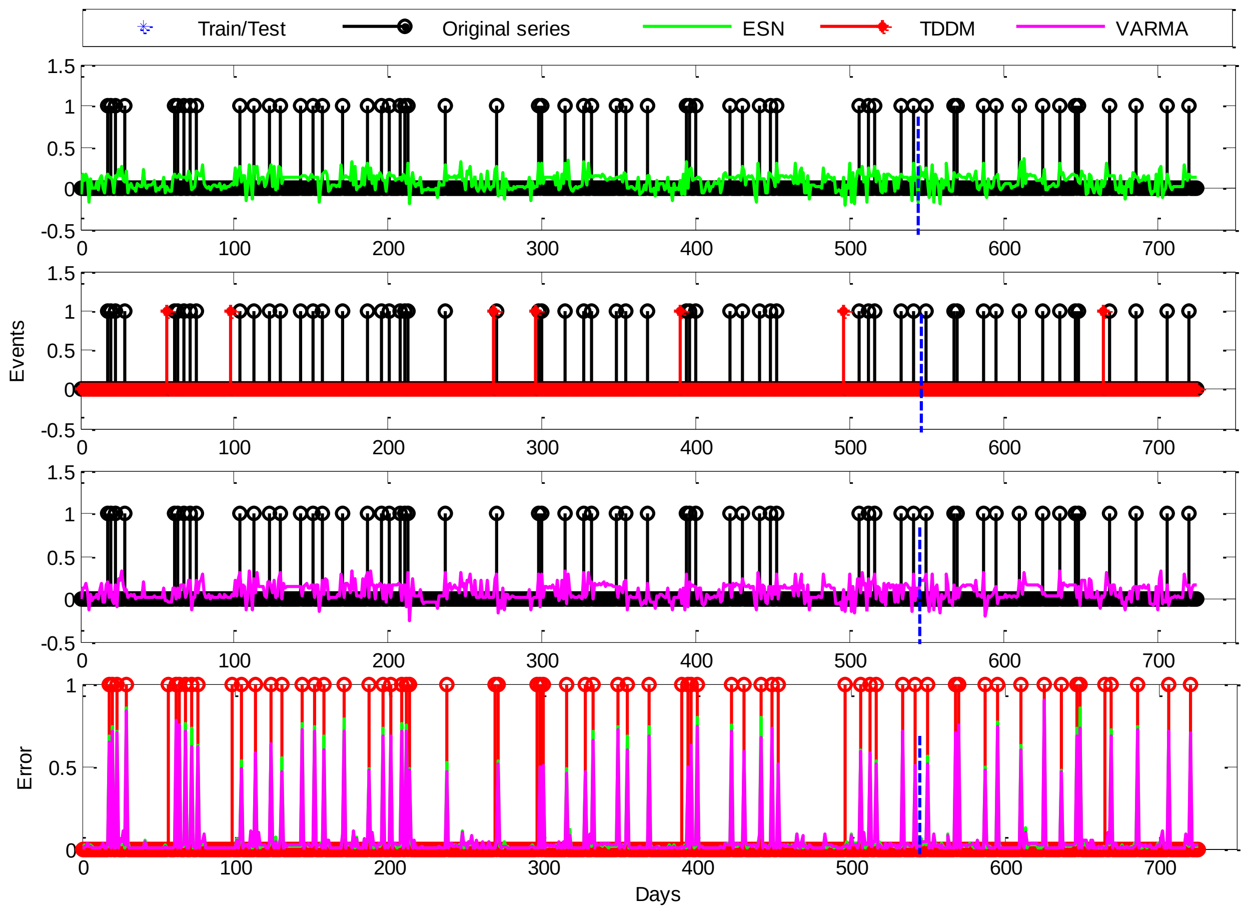


Figure 18: ESN, TDDM and VARMA algorithms prediction of events of time series 3 from stock 9. The train/test set separation is indicated by a blue line.

For the heartbeats, ESN and VARMA are predicting "0" all the time while TDDM predict some events but at the wrong place. Figure 19 below shows that the three algorithms are not learning at all to predict the events. One particularity to the heartbeats dataset is that each time series contains approximately 7×10^5 instances. In order to have a good visualization for the plots, only a portion of the time series i.e. 2×10^4 instances around the train/test boundary are shown in Figure 19 .

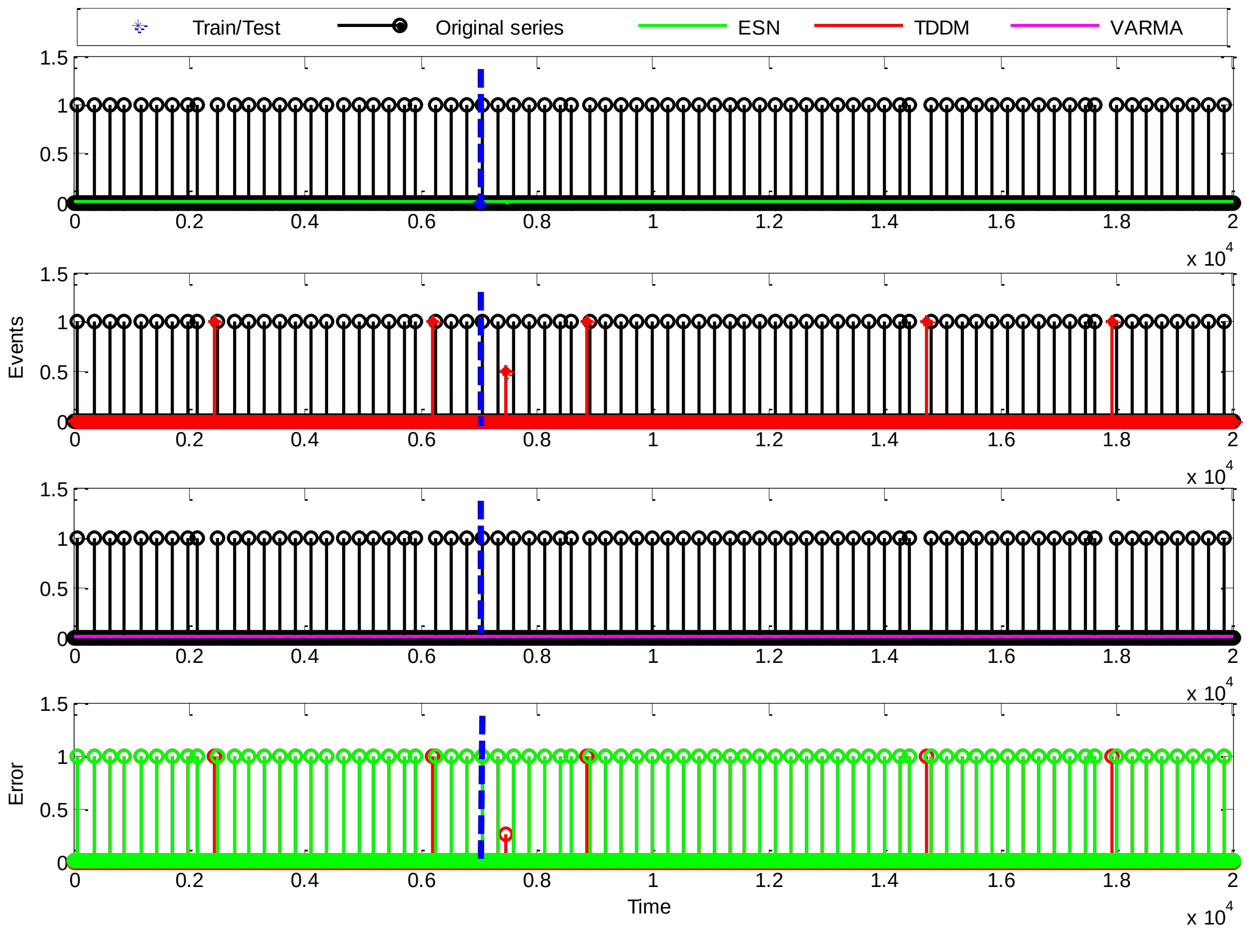


Figure 19: ESN, TDDM and VARMA algorithms prediction of events of heartbeats 10. The train/test set separation is indicated by a blue marker. 20000 instances are shown in this plot.

The fact that ESN and VARMA are not predicting completely the events meaning that it does not reach 1 in music and finance datasets, leads to the following question: Is VARMA and ESN trying to predict the events or are they predicting zeros all the time meaning that predicting the events is a new task for them and they are not learning it at all? For that reason, a confusion matrix has been made for each algorithm on each dataset to see how well or not the algorithms are classifying 1 as 1 and 0 as 0. A confusion matrix [100], also called table of confusion is a table that allows visualization of the performance of an algorithm. Mostly used in machine learning, it describes the performance of a classification model. The matrix contains information about actual and predicted classifications done by a classification system, for example the three algorithms in the case of this experiment. Using a confusion matrix, it will be easier to see if the algorithms are confusing two classes or confounding them one as another. In the context of this experiment, the two classes are “0” and “1”with “0” representing negative and “1” representing positive.

The forecast of each algorithm on each dataset were transformed to 1 if $\hat{z}_j(t) \geq 0.5$ and 0 otherwise. The following table shows the confusion matrix for a two class classifier. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice-versa). The entries in the confusion matrix have the following meaning in the context of this experiment:

I=Number of instances (or observations)		Predicted data or Predictions	
		Negative	Positive
Actual or Real data	Negative	*u*	*v*
	Positive	*r*	*s*

Table 5: Confusion Matrix

- u is the number of **correct** predictions that an instance is **negative**,
- *v* is the number of **incorrect** predictions that an instance is **positive**,
- *r* is the number of **incorrect** of predictions that an instance **negative**,
- *s* is the number of **correct** predictions that an instance is **positive**.

Using the information above, one can estimate the percentages of how well an algorithm can distinguish two classes. Several standard terms have been defined for the two class confusion matrix:

- **True Positive Rate (TP):** When the actual data are yes, how often does the system (in this case the algorithm) predict yes?
  - $TP = s\,/(\,r + s)$
- **False Positive Rate (FP):** When the actual data are no, how often does the algorithm predict yes?
  - $FP = v\,/(\,u + v)$
- **True Negative Rate (TN):** When the actual data are no, how often does the algorithm predict no?
  - $TN = u\,/(\,u + v)$
- **False Negative Rate (FN):** When the actual data are yes, how often does the algorithm predict no?
  - $FN = r\,/(\,r + s)$
- **Misclassification Rate or Error rate (M):** Overall, how often the algorithm is wrong?
  - $M = \;(FP + FN)\,/\,I$

The Table 6 and Table 7 below show the percentages of the confusion matrix and the misclassification rate obtained by averaging over the 10 cross-validation folds for each algorithm and each dataset respectively.

Confusion matrix percentages												
Algorithms	ESN				TDDM				VARMA			
Percentages	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
Music	0.37%	98.87%	1.12%	69.01%	0%	98.58%	1.41%	69.3%	0.08%	98.96%	1.03%	69.3%
Finance	0.06%	99.94%	0.05%	99.93%	0%	99.63%	0.36%	100%	0.05%	99.77%	0.22%	99.94%
Heartbeat	0%	100%	0%	100%	0%	99.90%	0.09%	100%	0%	100%	0%	100%

Table 6: Average percentages of the confusion matrix for the three algorithms over the 10 folds for all the datasets

Misclassification Rate			
Algorithms	ESN	TDDM	VARMA
Music	3.63%	3.95%	3.55%
Finance	7.62%	7.91%	7.78%
Heartbeat	0.33%	0.42%	0.33%

Table 7: Algorithms average misclassification rate over the 10 folds for all the datasets

For example, for Figure 18 of stock above, the total instances in the testing part is 182 with number of positive instances equal to 14, and negative instances equal to 168. The misclassification rate for ESN is 0.082%, which is normal as more 0 exist than 1 in the time series. These tables are good at showing how well the 0 are classified for each algorithm with the average true negative (TN) percentage equal approximately to 99 %. One can notice that ESN and VARMA on the heartbeat are not learning at all. The true positive (TP) is 0% and the TN is 100%. The TP for VARMA on music is very small like the ESN on finance. As for TDDM the TP is 0% on all dataset. To investigate how well the positive instances "1" are predicted correctly by each algorithm the tables below are generated which contain the percentage of real positive instances and the percentage of predicted positive instances for each fold and for each dataset. These tables clearly show per fold how well the "1", which represent the events, are predicted by each algorithm.

Fold	Music: Percentages of positives instances			
	Actual data	Forecast data		
		ESN	TDDM	VARMA
1	2.90%	1.20 %	1.42%	1.18%
2	2.71%	1.05 %	1.37%	1.02%
3	2.14%	0.74 %	1.51%	0.80%
4	2.67%	0.99 %	1.32%	1.002%
5	2.15%	1.26 %	1.18%	0.70%
6	2.78%	1.05 %	1.33%	1.054%
7	2.86%	1.34 %	1.66%	1.25%
8	2.70%	1.09 %	1.06%	0.84%
9	2.11%	1.10 %	1.42%	0.92%
10	2.68%	0.94 %	1.52%	1.001%
Average	2.57%	1.07%	1.37%	0.97%

Table 8: Percentages per fold of predicted and actual positives observations for the music dataset

For the music dataset, TDDM has the higher average positive instances predictions of 1.37% compare to ESN with 1.07% and VARMA with 0.97%. This shows that the algorithms are barely predicted the half of the events correctly.

Fold	Finance: Percentages of positives instances			
	Actual data	Forecast data		
		ESN	TDDM	VARMA
1	7.18%	0%	0.55%	0%
2	7.84%	0.17%	0.17%	1.50%
3	7.22%	0%	0%	0%
4	7.18%	0%	0.53%	0%
5	8.08%	0%	0.43%	0%
6	8.22%	0.07%	0.15%	1.25%
7	6.48%	0%	0.54%	0%
8	7.06%	0.54%	0.54%	0%
9	8.28%	0.10%	0.45%	0%
10	5.37%	0%	0.53%	0%
Average	7.29%	0.08%	0.38%	0.27%

Table 9: Percentages per fold of predicted and actual positives observations for the finance dataset

For the finance dataset, the results are showing that all the algorithms have difficulties in predicting even 1% of the events. The best among them is TDDM with an average percentage of predicted events equal to 0.38%.

Fold	Heartbeat: Percentages of positives instances			
	Actual data	Forecast data		
		ESN	TDDM	VARMA
1	0.44%	0%	0.10%	0%
2	0.34%	0%	0.15%	0%
3	0.33%	0%	0.09%	0%
4	0.29%	0%	0.02%	0%
5	0.39%	0%	0.07%	0%
6	0.27%	0%	0.04%	0%
7	0.28%	0%	0.07%	0%
8	0.31%	0%	0.14%	0%
9	0.23%	0%	0.06%	0%
10	0.34%	0%	0.23%	0%
Average	0.32%	0%	0.09%	0%

Table 10: Percentages per fold of predicted and actual positives observations for the heartbeat dataset

Regarding the heartbeat dataset, the ESN and VARMA are not at all learning to predict the events and TDDM has an average of 0.09% which is again very small.

Experiment I shows that it is indeed very difficult to learn to predict event occurrences. Even though the percentage of events is small in the time series, the algorithms were hardly predicting correctly even one third of the events. TDDM which has been made to predict timing events was also not good enough on predicting event onsets. Moreover, although TDDM performs worst in RMSE, one can see that it is attempting to miss fewer events. But TDDM timing error increases its RMSE in a potentially unfair way compared to majority vote. In short, in this experiment, results are bias toward making no predictions at all rather than miss predicting the timing of an event. Given the poor results of Experiment I, an alternative to the prediction of timing events will be to learn to predict the time remaining before the occurrences of events as suggested by the forecasting event timing concept based on animal learning. The learning of time remaining may be easier as oppose to learning the binary events stream. For that reason, Experiment II is performed.

5.3.2 Experiment II:

The procedure of the second experiment is similar to the first one but the difference is in the time series being used. As shown in Experiment I, the prediction of events is very difficult for all the algorithms. An alternative to this problem is to predict the time remaining before the event occurs. Therefore, in this experiment, all the algorithms receive as inputs the binary time series $\boldsymbol{x}(\boldsymbol{t})$ and as target output the time remaining time series $\boldsymbol{y}(\boldsymbol{t})$ instead of the events time series $\boldsymbol{z}(\boldsymbol{t})$ and should output an estimate of the remaining time before an event occurs.

The forecasts $\hat{y}_j(t)$ are estimated by VARMA using equation (9) as described in section 2.2. One can notice that VARMA uses the real time remaining $y_j(t)$ for the forecast estimation.

$$\hat{\boldsymbol{y}}(\mathrm{t}) = \sum_{k=1}^{p} \mathrm{A}(k)\boldsymbol{y}(t-k) + \sum_{k=0}^{q} \mathrm{B}(k)\,\boldsymbol{x}(t-k) \qquad (9)$$

The TDDM algorithm, as opposed to VARMA estimates the predictions using only the weights and the binary input series as described in equation (18) of section 4. Note that the accumulations a_i imply that TDDM uses the past events $\boldsymbol{z}(\boldsymbol{t})$ to estimate its predictions.

$$\hat{y}_j(t) = \frac{1-\sum_{i=1}^{n} w_{j,i} . a_i(t)}{\sum_{i=1}^{n} w_{j,i} \cdot x_i(t)} \qquad (18)$$

The ESN algorithm receives as input the binary series $\boldsymbol{x}(\boldsymbol{t})$ and as target output the time remaining series $\boldsymbol{y}(\boldsymbol{t})$. Table 11 below shows the overall performance for each algorithm for Experiment II.

Datasets \ Algorithms	ESN	TDDM	VARMA
Bach Chorales ($^1/_{16}$notes)	35 ± 5	94 ± 113	596 ± 601
Stocks (days)	10 ± 3	721 ± 2193	4.3 ± 0.6
Heartbeats ($^1/_{360}$ second)	95 ± 16	33 ± 16	18 ± 2

Table 11: Average root mean squared errors and standard deviation over the 10 folds for all the algorithms.

The analysis of variance of Table 11 above gives a *p* value of 0.0024 for the Bach Chorales. A post-hoc Scheffe test (with α=0.05) reveals that VARMA is significantly worst than ESN and TDDM. The ANOVA on stocks had a *p* value of 0.36. Yet, a post-hoc Scheffe test suggested all means for the three algorithms to be significantly different. On heartbeat, the analysis of variance gives a *p* value of 8.2x10^{-13} and the post-hoc Scheffe test reveals that all the three algorithms are significantly different from each other. In the light of the ANOVA and post-hoc Scheffe test results, a comparison can be then make between the algorithms errors.

The ESN and TDDM algorithms perform better on the Bach Chorales while VARMA has the lowest error on the stocks data and heartbeats. Nonetheless, it is important to mention that the comparison is not totally fair since VARMA relies on knowing the exact remaining time at the previous time step to make its prediction while TDDM and ESN are generating their estimates based only on the previous inputs.

5.3.2.1 Bach Chorales

In music, even though the errors were smaller than the VARMA algorithm, it is on Bach Chorales 47 that TDDM failed (see Figure 20 below) by having a unique prediction which went into the thousands by having an error of $1{,}29 \times 10^3$ compared to an average error of 61. This piece has a long blank space (no sound for a few bars).

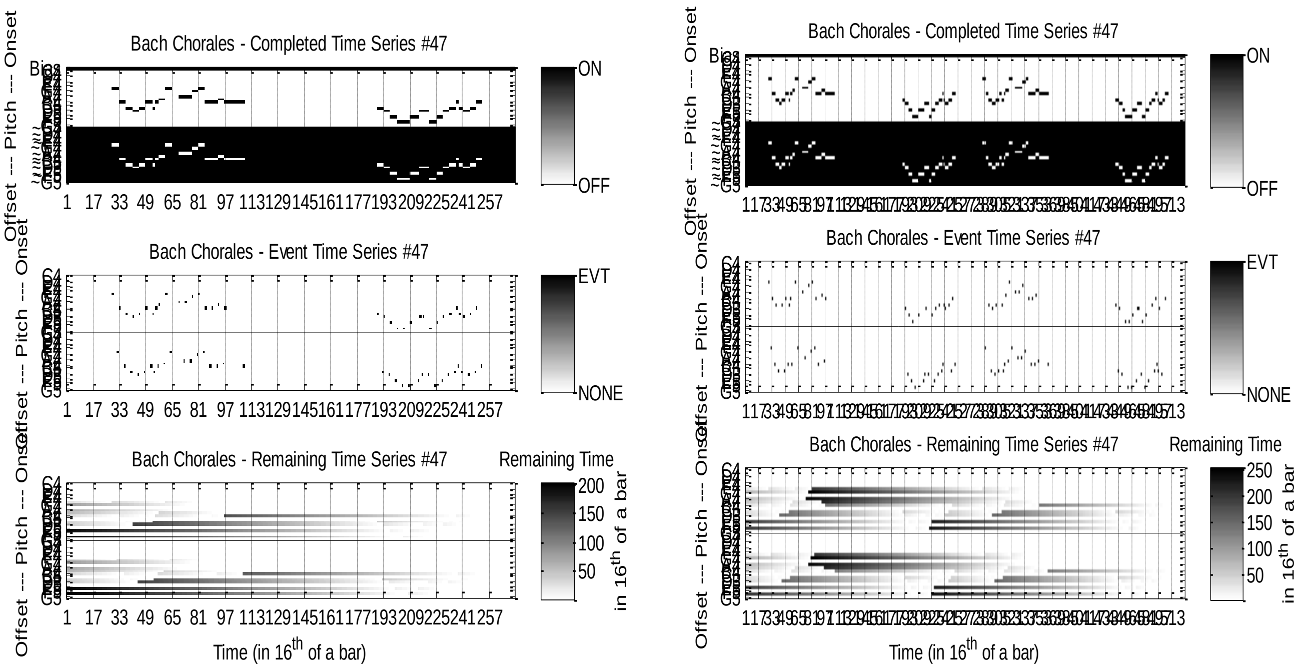


Figure 20: Music notes 47 causing the huge error in TDDM. On the middle right, one can see the two big blanks generating the huge error and the middle artificial blank.

An analysis of all the pieces in the Bach Chorales was made to see if any others pieces had a similar blank. This unique piece has a blank of approximately 90 steps unlike the other pieces which do not have except at the middle where an artificial blank was added. The space we added in each piece while doing the repetitions was approximately 12 time steps. This outlier may explain why the TDDM performs poorly in Music. This piece has been removed and new averages have been calculated and presented in Table 12.

Algorithms / Datasets	ESN	TDDM	VARMA
Bach Chorales ($^1/_{16}$notes)	35± 5	58 ± 17	596 ±601

Table 12: Average root mean squared errors and standard deviation over the 10 folds on Bach Chorales after removing the outlier time series.

Even though VARMA has the lowest error in the finance; its errors in Bach Chorales are considerable. Unlike the TDDM algorithm, the high errors of order of thousands for VARMA occur in more than three pieces. A close look at these pieces didn't show any irregularities that could explain the problem. The chorale #47 that causes the huge error in TDDM works fine in VARMA. The VARMA best *p* and *q* couple of hyper-parameters over the 10 cross-validation folds for Bach Chorales are listed in the table below. The folds that have the less prediction errors in the whole dataset respectively are F6, F7 and F2. One can notice that the best (*p,q*) for these folds are mainly (14,12). On the other hand, the fold that has the greatest error in the order of thousands (3160) is the fold 9 whose (*p,q*) is (1,10). It appears that when the order of the autoregressive part is higher, the predictions are better. Contrary to the general rule that the first order of an AR (*p*) process is satisfactory enough [2] , here, in the case of the prediction of timing event, the model needs higher orders that can go farther in the past to predict the next time step.

Table 13 below shows the best (*p,q*) for each fold on the music time series. Note that 14 time steps is almost a full bar (16 time steps).

Fold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
P	**8**	**14**	**8**	**4**	**5**	**14**	**14**	**14**	**1**	**10**
Q	**15**	**12**	**13**	**15**	**14**	**12**	**12**	**12**	**14**	**12**

Table 13: Best (*p,q*) for each fold on the music time series

Figure 21 below show the experiment results for the three algorithms on the Bach Chorales dataset. The three algorithms are mimicking the overall pattern of the actual remaining time with some errors. One can notice that for this time series, the TDDM algorithm predictions in the testing part look overall better than VARMA and ESN.

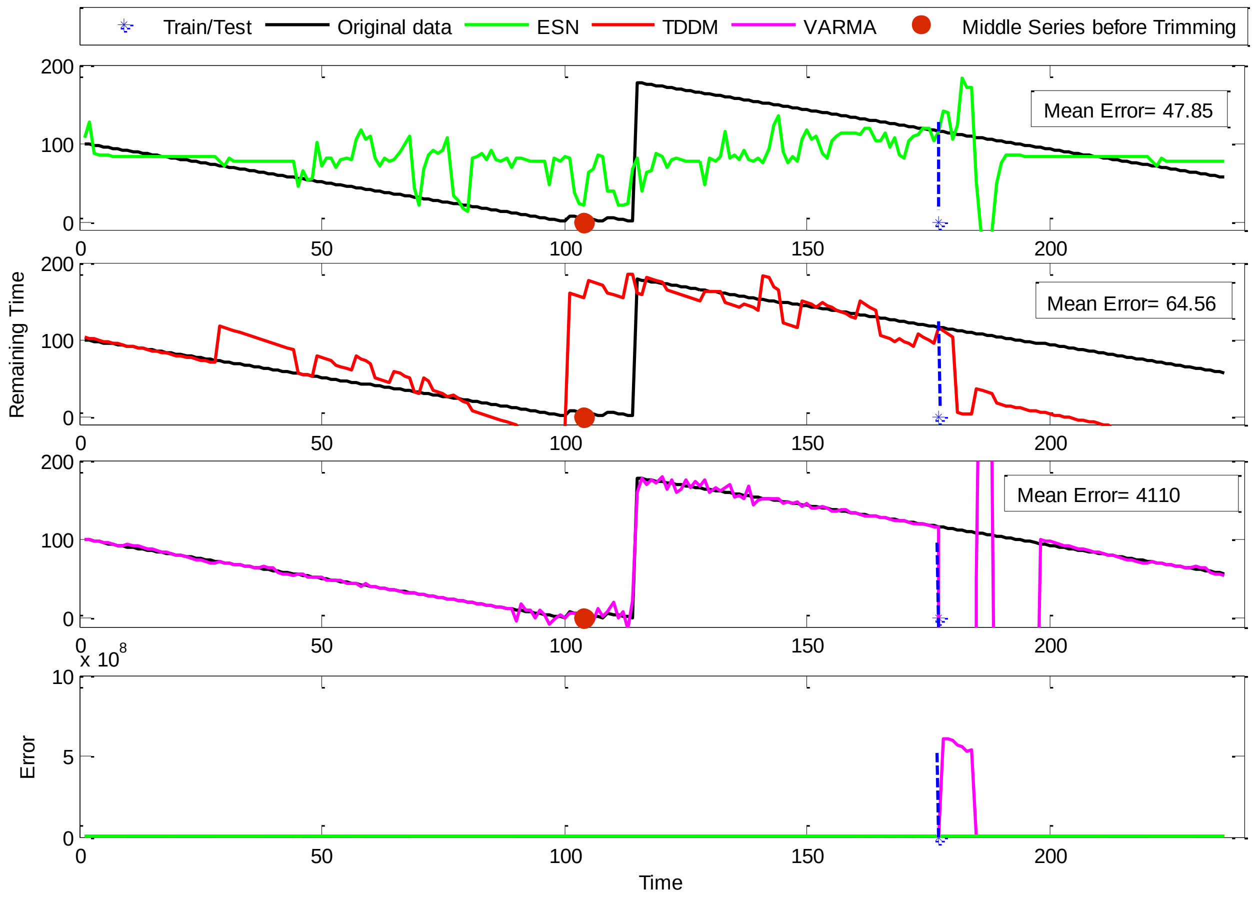


Figure 21 : ESN, TDDM and VARMA algorithms prediction of remaining and the mean squared error of note 11 from Bach Chorales 4. The train/test set separation is indicated by a blue marker. The middle of the music piece before trimming and repetition is illustrated by a brown dot.

5.3.2.2 Stock dataset

In finance, the TDDM had a unique prediction with an error of 10^5 orders of magnitude higher than any other prediction error. A numerical stability problem with a division by a value near zero is suspected. A close look at this stock shows that among the 40 time series of the stock, one particularly has an error of the order 10^5 (time series # 6) and a second one (time series # 33) has an error of the order 10^4. For these time series, the huge errors occur at time step t=78 and t=64 respectively, which have not only their weights equal to zero but also the summation of the dot product between the inputs series at these time steps and the weights ($\sum_{i=1}^{n} \boldsymbol{w_{j,i}} \cdot \boldsymbol{x_i}(t)$) is equal to 1.2×10^{-6} and -2.09×10^{-5} which are tending toward zero. Therefore, these two time series were removed and new averages were calculated which are presented in Table 14 below.

Algorithms / Datasets	**ESN**	**TDDM**	**VARMA**
Stocks (days)	10±3	30±40	4.3 ± 0.6

Table 14: Average root mean squared errors and standard deviation over the 10 folds on finance dataset after removing the outlier time series.

VARMA has the lowest error in the finance. Nevertheless VARMA in this experiment has an advantage over TDDM and ESN since VARMA can rely on the exact previous remaining $y_j(t-k)$ time to predict constant decrease at each time step while TDDM and ESN are generating their estimates based only on the previous binary inputs and their internal memory. In finance, VARMA best p and q couple of hyper-parameters over the 10 cross-validation folds are mainly p=1and q=1. Only one fold has a (p,q) of (1,2).

ESN uses a different approach by mapping the temporal pattern into a hidden-neurons activity pattern that can then be used to predict the remaining time. This advantage suggests to us that the use of more internals neurons allows the ESN to learn more features of the data. The use of 10 internal memory neurons was the best for this experiment. Table 14 summarizes the average root squared errors for the three algorithms with the outlier removed for TDDM.

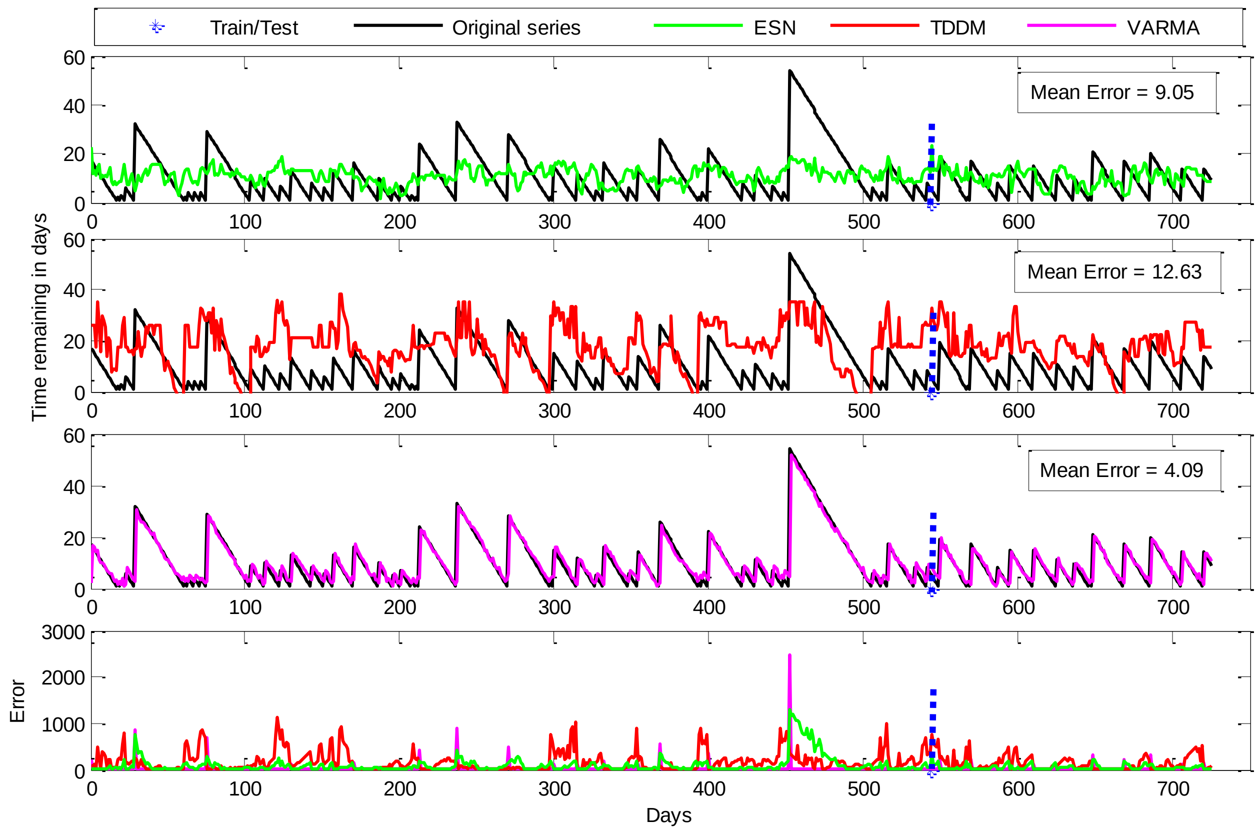


Figure 22 : ESN, TDDM and VARMA algorithms prediction of remaining time and the mean squared error of the time series 3 from the stock 9. The train/test set separation is indicated by a blue marker.

Figure 22 shows that the three algorithms are mimicking the overall pattern of the actual remaining time with some errors. Regarding TDDM, its mathematical form does not allow it to discriminate that two events from the same note can be preceded by different temporal patterns [1]. Therefore, although it has relatively good shape compared to ESN, it does not tend to restart at the appropriate time. In contrast, VARMA can rely on its previous remaining time to predict constant decrease at each time step. ESN uses a different approach by mapping the temporal pattern into a hidden-neurons activity pattern that can then be used to predict the remaining time. However, one can notice that despite the fact that ESN has less error; its predictions are not mimicking the original time series as well as VARMA and TDDM.

5.3.2.3 Heartbeat

On heartbeats, the VARMA algorithm has the lowest error followed by TDDM. ESN as shown in Figure 23 is having difficulties both on training and testing part.

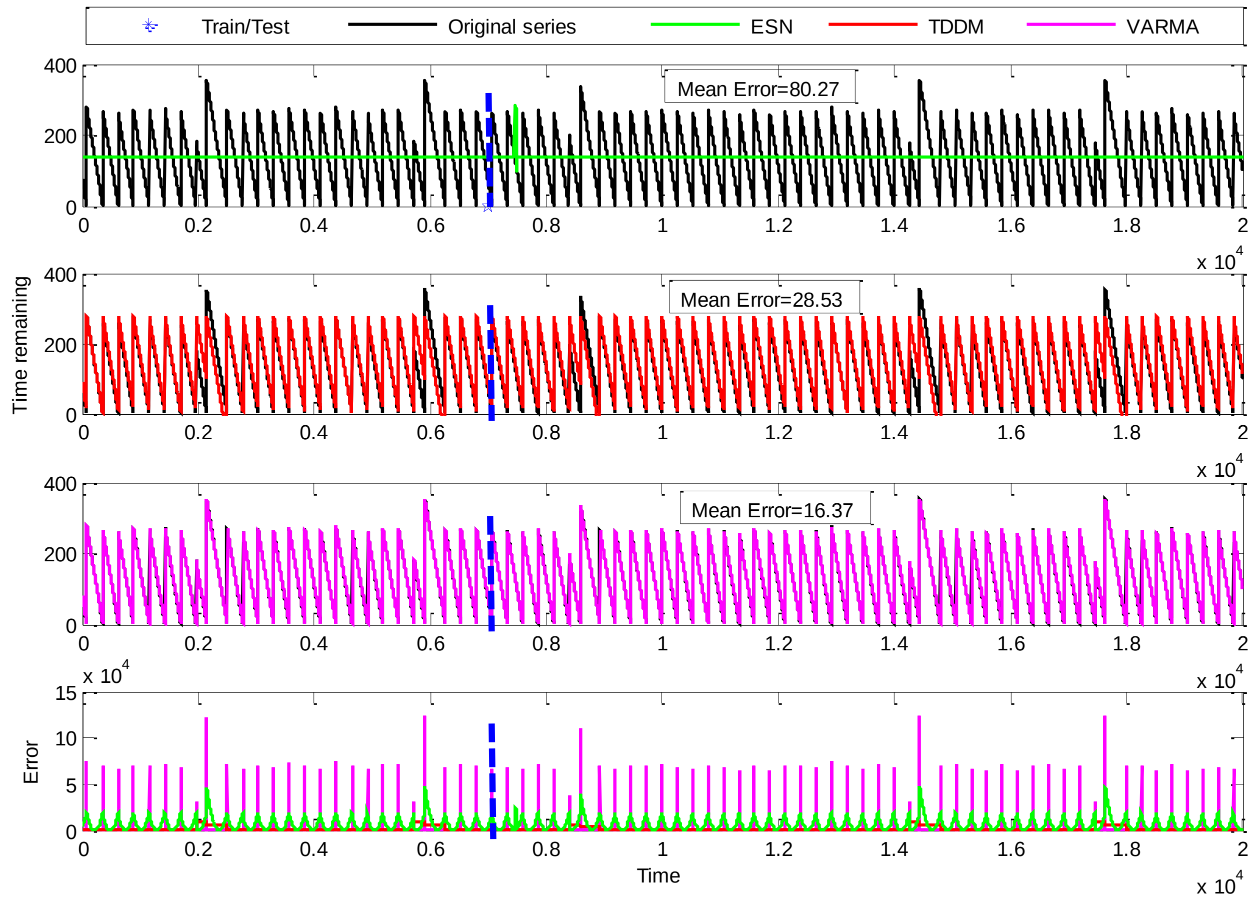


Figure 23: ESN, TDDM and VARMA algorithms prediction of remaining time and the mean squared error of heartbeat 10. The train/test set separation is indicated by a blue marker.

5.3.2.4 VARMA II

As mentioned at the beginning, the comparison between the three algorithms is not fair since VARMA can rely on the previous real remaining time to predict constant decrease at each time step while TDDM and ESN are generating their estimates based only on the previous binary inputs and also the previous events as for TDDM. For this reason, in order to put VARMA in the same environment as its two other counterparts, some changes were made. A new version of VARMA has been developed where the algorithm instead of using the real past remaining time in the testing part to estimate the next time step, will use its own past predictions to forecast the next time step if that one is in the test part (see equation 25 below). This new setup is called VARMA II. The difference is that in VARMA II, VARMA will receive as usual the inputs binary series and the time remaining outputs for the training however unlike experiment one, in the testing part, VARMA will use its owns past estimate of time remaining to estimate the next time step.

$$\hat{\boldsymbol{y}}(t) = \sum_{k=1}^{p} \mathrm{A}(k)\hat{\boldsymbol{y}}(t-k) + \sum_{k=0}^{q} \mathrm{B}(k)\,\boldsymbol{x}(t-k) \quad (25) \qquad \text{if time step } (t-k) \text{ is in test}$$

Table 15 below shows the root mean squared error comparison between the VARMA and VARMA II. It clearly appears that the VARMA algorithm performs poorly when its uses its own past predictions. In both datasets (stocks and music) the errors increased compare to the previous version of VARMA specifically the music dataset where the RMSE increases to the order of 10^{100}.

Algorithms	**ESN**	**TDDM**	**VARMA**	**VARMA II**
Bach Chorales ($^1/_{16}$notes)	35 ± 5 *	58 ± 17	596 ± 601	$410^{100} \pm 210^{101}$
Stocks (days)	10 ± 3 *	30 ± 43	4.3 ± 0.6	10 ± 3 *
Heartbeat ($^1/_{360}$ second)	95 ± 16	33 ± 16 *	18 ± 2	94 ± 16

Table 15: Averaged test root mean squared error per prediction for all the algorithms including VARMA II and outliers removed for TDDM. The * stand for significantly better algorithms (excluding VARMA).

Because at each time step the next output is estimated based on the previous estimates, the predictions errors keep propagated in the future which explain the high error of VARMA II on Bach Chorales.

On Table 15 where the outliers of TDDM have been removed and the errors of VARMA II have been calculated, an ANOVA and T-test has been performed. A T-test like ANOVA is a statistical technique used to verify if two population or two different methods means are reliably different from each other. The T-test is performed on the Bach Chorales and the ANOVA on the stocks and heartbeat. The T-test results for Bach Chorales are: the null hypothesis can be rejected with *p*=0.012; that means that ESN is significantly better than TDDM.

The ANOVA on stocks gives a p value of 0.14 while the post-hoc Scheffe test shows that TDDM is significantly different from ESN and VARMA II. Regarding the heartbeat dataset, the same conclusions as stocks were made. With p= $1.65\text{x}10^{-09}$, the errors of ESN and VARMA II are the same as oppose to TDDM's error which is significantly different from the others. Note that although TDDM is worst on stocks, it is better on heartbeat.

In Figure 24 below, the error of VARMA went very high in the testing part on the music notes showing by that the algorithm is not predicting anymore the actual remaining time unlike in the training part where it used the real time remaining to compute the parameters (weights).

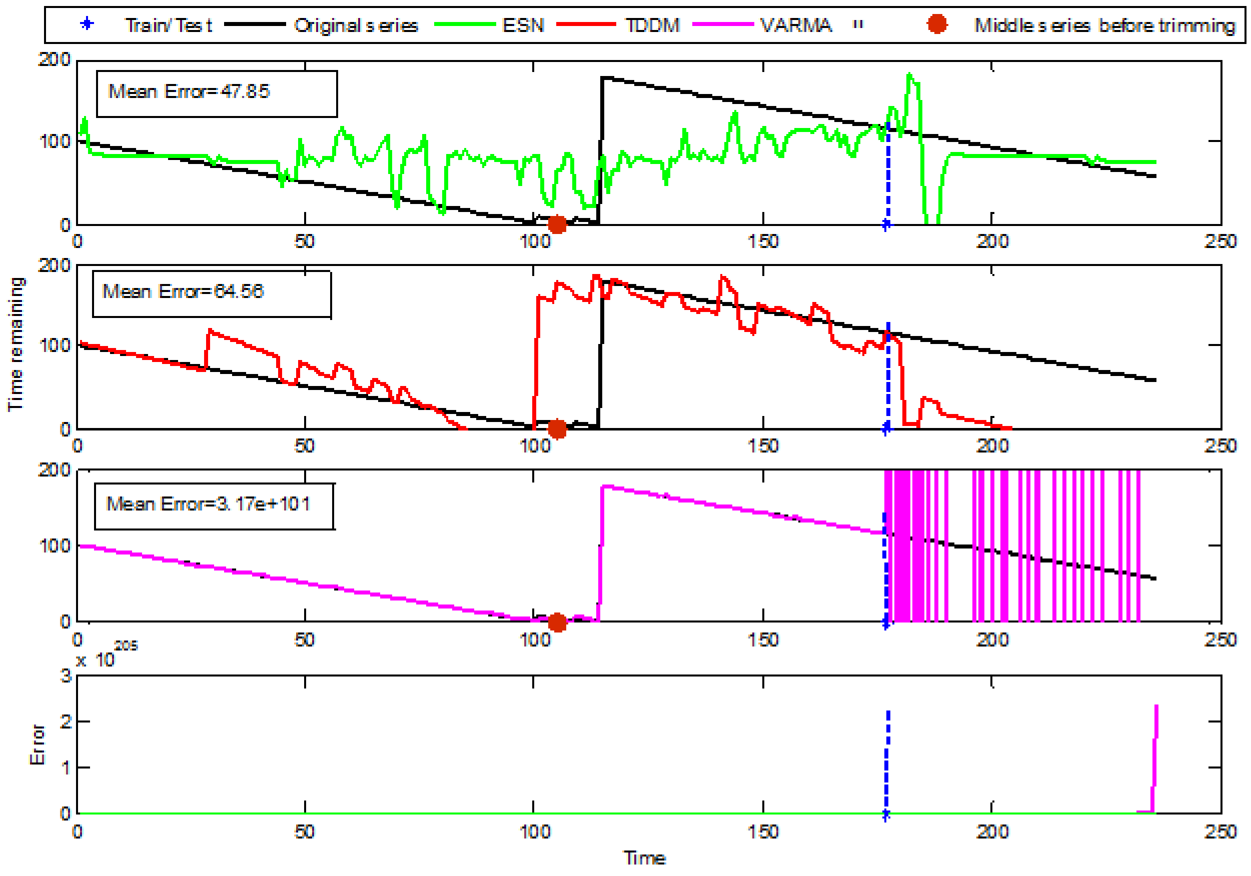


Figure 24: ESN, TDDM and VARMA II algorithms prediction of remaining time and the mean squared error of note 11 from Bach Chorales 4. The train/test set separation is indicated by a blue marker. The middle of the music piece before trimming and repetition is illustrated by a brown dot.

Regarding the stocks, one can still remark that the algorithm predictions are poor as soon as it reaches the testing part (see Figure 26 compare to Figure 22). Nonetheless the overall average root error (see Table 15) for VARMA II on stock is still lower than TDDM and similar to ESN.

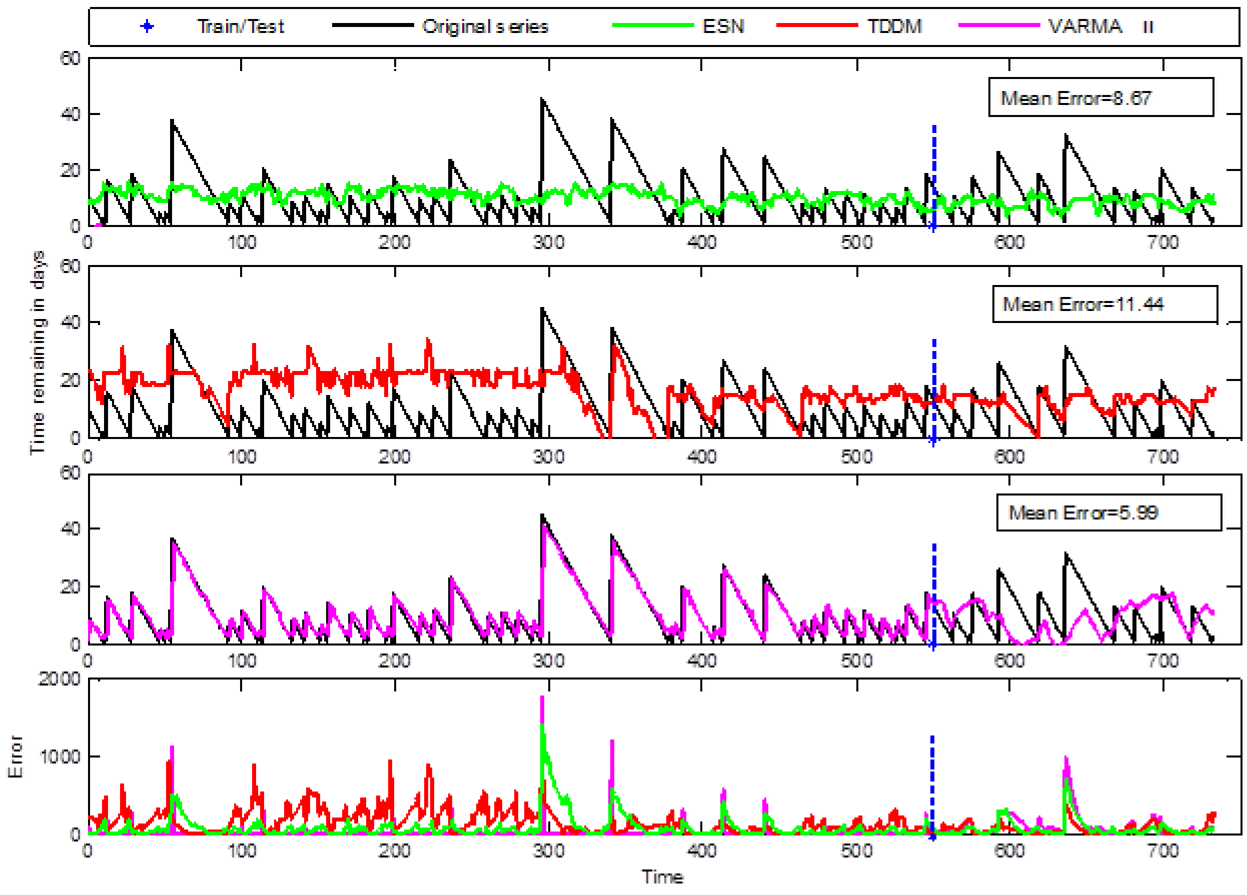


Figure 25: ESN, TDDM and VARMA II algorithms prediction of remaining time and the mean squared error of the time series 5 from the stock 8. The train/test set separation is indicated by a blue marker.

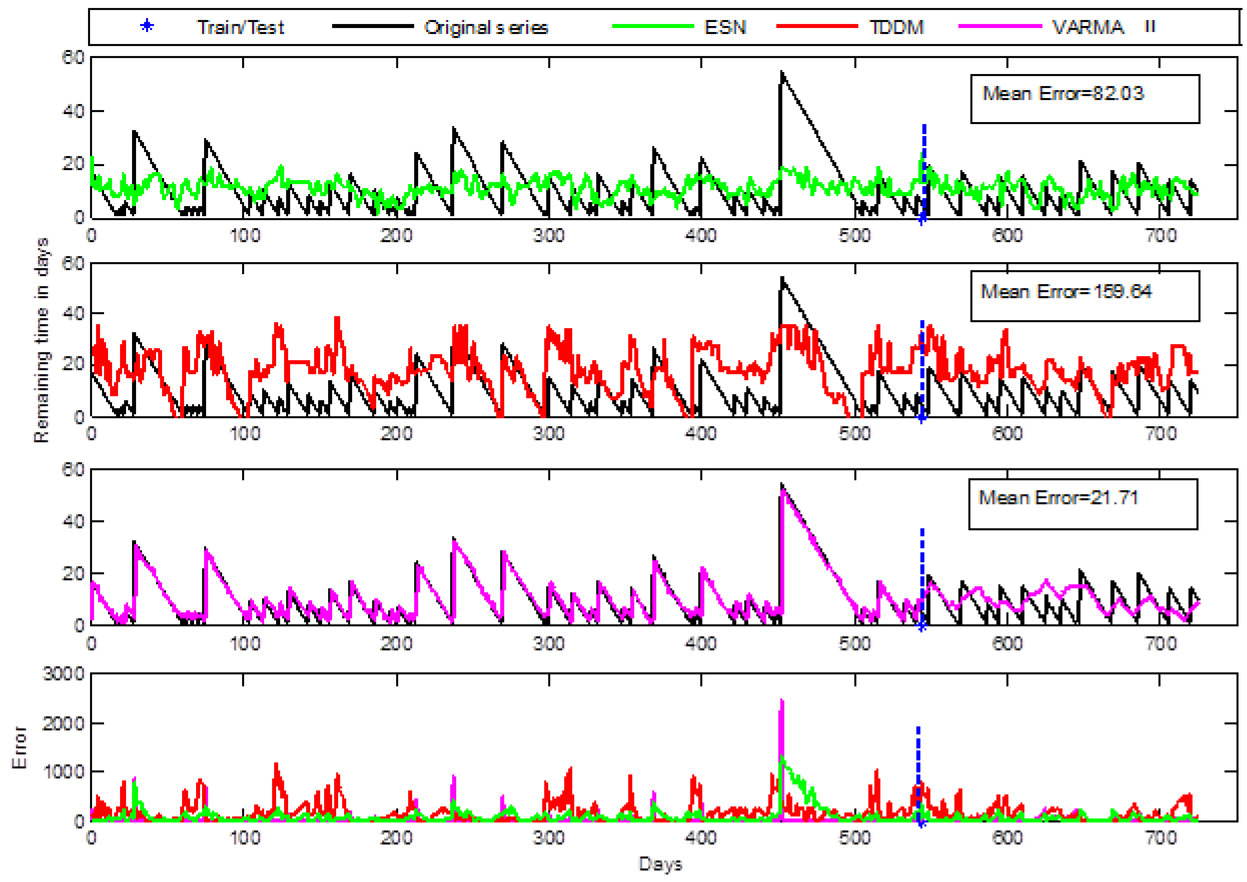


Figure 26: ESN, TDDM and VARMA II algorithms prediction of remaining time and the mean squared error of the time series 3 from the stock 9. The train/test set separation is indicated by a blue marker.

On the heartbeat, the error increased from approximately 18 to 94, which is significant. One can also observe (see Figure 27 below) that as soon as the testing is reached, VARMA II was unable to generate any good predictions. In summary, VARMA II has difficulties in predicting the time remaining before an event occurs in a multiple time step forecasting setup.

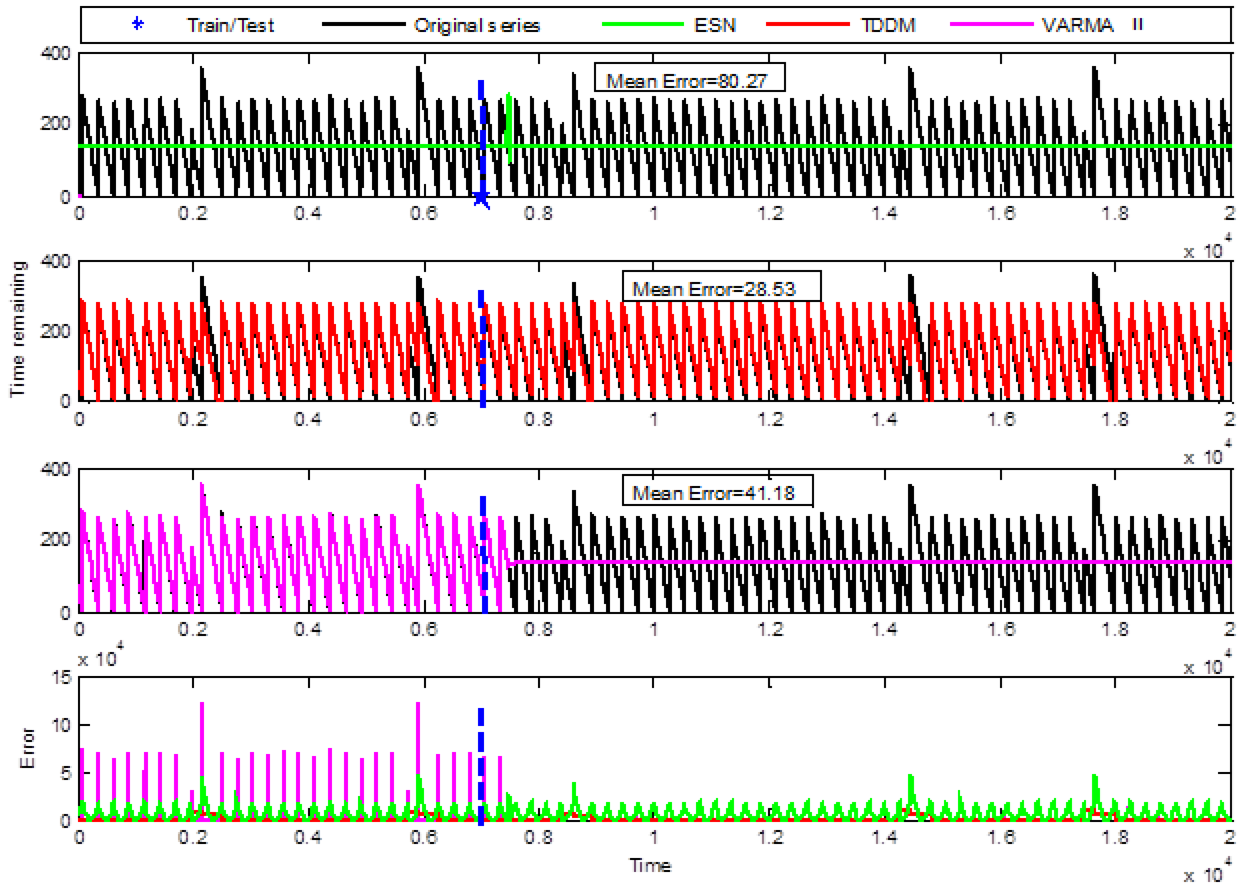


Figure 27: ESN, TDDM and VARMA II algorithms prediction of remaining time and the mean squared error of the heartbeat 10. The train/test set separation is indicated by a blue marker.

Overall VARMA II and ESN had similar performance on stocks and heartbeat.

5.3.3 Comparison of the two experiments

At the end of the experiments, a comparison of both experiments was attempted by changing the binary predictions of Experiment I in time remaining and then the latter was compared with the time remaining predictions of Experiment II. This means that in experiment I, the portion of the time series after the last event of type *j* occurs would be transformed in NaN as the time remaining will become undefined. However, when the percentage of NaN in each time series over the 10 folds was calculated, it was observed (in Table 16) that the bulk of the data become useless as most of it will be NaN.

Percentage of NaN over the 10 folds	**ESN**	**TDDM**	**VARMA**
Bach Chorales	76.87%	93.28%	94.91%
Stocks	97.91%	99.03%	96.56%
Heartbeat	100%	7.9%	100%

Table 16: Average percentages of NaN on the last 25% of the series over the 10 folds for each datasets and each algorithm, when using the predicted event stream $\hat{\boldsymbol{z}}_{\boldsymbol{j}}(\boldsymbol{t})$ to generate remaining time $\hat{\boldsymbol{y}}_{\boldsymbol{j}}(\boldsymbol{t})$

Nonetheless, using the event-rate, one can still draw a best case scenario in order to compare Experiment I and Experiment II. The event-rate is the occurrence of an event every K time steps. Using the percentages of real and predicted positive instances shown in Table 8, Table 9 and Table 10 from Experiment I, one can estimate the average K time steps predicted by each algorithm on each dataset. The real positive instances would produce the event-rate of an algorithm considering the best case scenario and the predicted positives instances produces the event-rate of the ESN, TDDM and VARMA algorithms. Therefore K can be calculated by doing the inverse of the average real and predicted positives instances for each dataset. The average K time steps for the predicted positive instances will be generated for all the datasets by choosing the TDDM averages in Table 8, Table 9 and Table 10 for the reason that TDDM has the highest percentages of predicted positive instances. By calculating the RMSE at every time step between the best case time series generated by the event-rate from the average of real positive instances and the predicted time series generated from the predicted positive instances of the TDDM algorithm, one can compare the obtained RMSE with the ones in Table 15 in order to see which Experiment (Experiment I or II) is better.

Therefore considering the Bach Chorales dataset, with an average of real positives instances equal to 2.57%, the $K_{\text{Bach Chorales}}$ would be equal to 38. Suppose that the events are occurring at constant intervals of time, a best algorithm would predict an event at every 38 time steps. With the average of predicted positive instances equivalent to the half of the real positives instances, meaning the three algorithms together are predicting 50% of the events, the K_{TDDM} will be 73 time steps. This implies that the algorithms are missing at least one event every two events see Figure 28(a). When using the best algorithm time series and the TDDM event-rate on Bach Chorales from Experiment I, an estimate time remaining error (RMSE) as in Experiment II can be obtained. The estimate RMSE for Bach Chorales from Experiment I is approximately equal to 27 of 1/16 of notes compare to the one in table 15 of Experiment II which best is 35 of 1/16 of notes. In this case it is better to predict 27 of 1/16 of notes compare to 35 of 1/16 of notes. Therefore, in Bach Chorales, it is not clear if Experiment I or Experiment II is better.

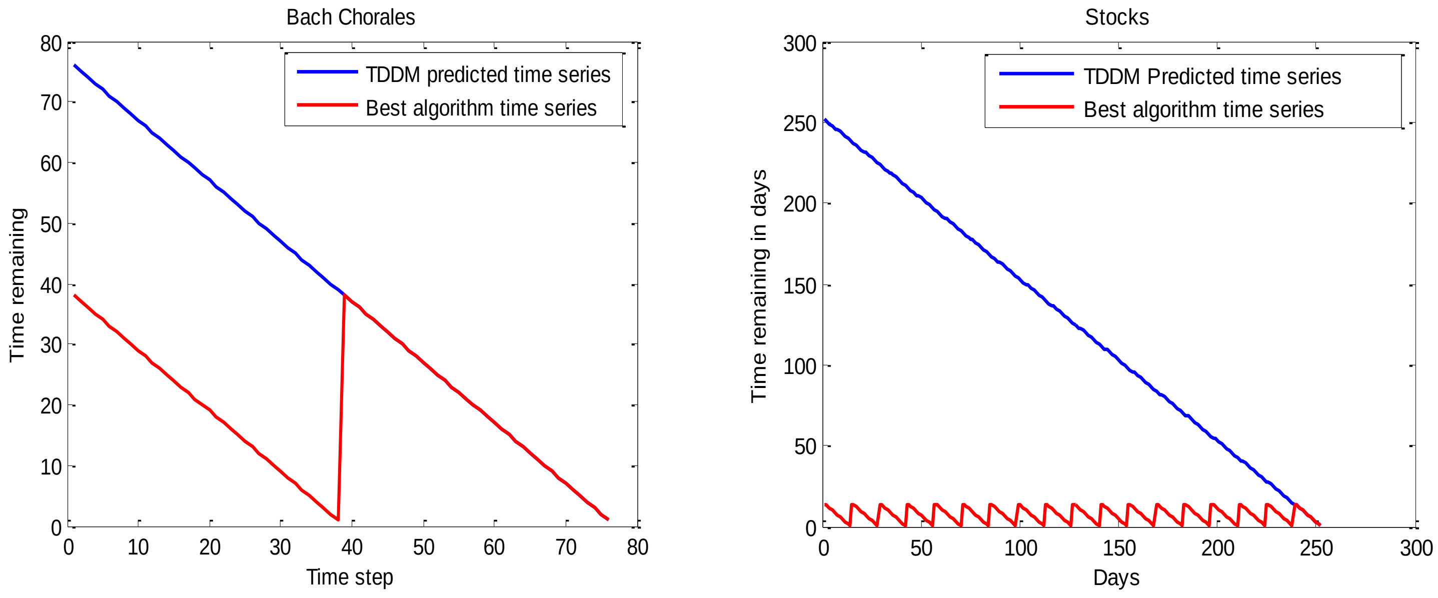


Figure 28: Best algorithm time series vs TDDM predicted time series: a) Bach Chorale and b) Stocks

The same analysis on the stock dataset will give K_{Stocks} =14 time steps for a best algorithm. The event-rate of TDDM which has the highest average predicted instances will give K_{TDDM} = 263 time steps which means that approximately 18 events will be missed before the algorithm catches as one see Figure 28(b). The RMSE approximated from Experiment I is approximately equal to 138 days compare to the one in table 15 which are 4, 10 and 30 days for VARMA, ESN and TDDM respectively. It is highly much better when predicting the time remaining before the stock prices jump to have an error of 4, 10 or 30 days than 138 days making Experiment II much better than Experiment I.

For heartbeat, by following the same analysis, $K_{\text{Heartbeat}}$ for a best algorithm is 312 time steps. The K_{TDDM} will be at every 1110 time step which means that approximately 3 events will be missed before the algorithm catches as one see Figure 29. The RMSE is approximately 400 in 1/360 second compare to respectively 33, 94 and 95 for TDDM, VARMA and ESN. Predicting the next heartbeat at 33 in 1/360 second is much better than predicting 400 in 1/360 second. Therefore, for heartbeat, Experiment II is much better than Experiment I.

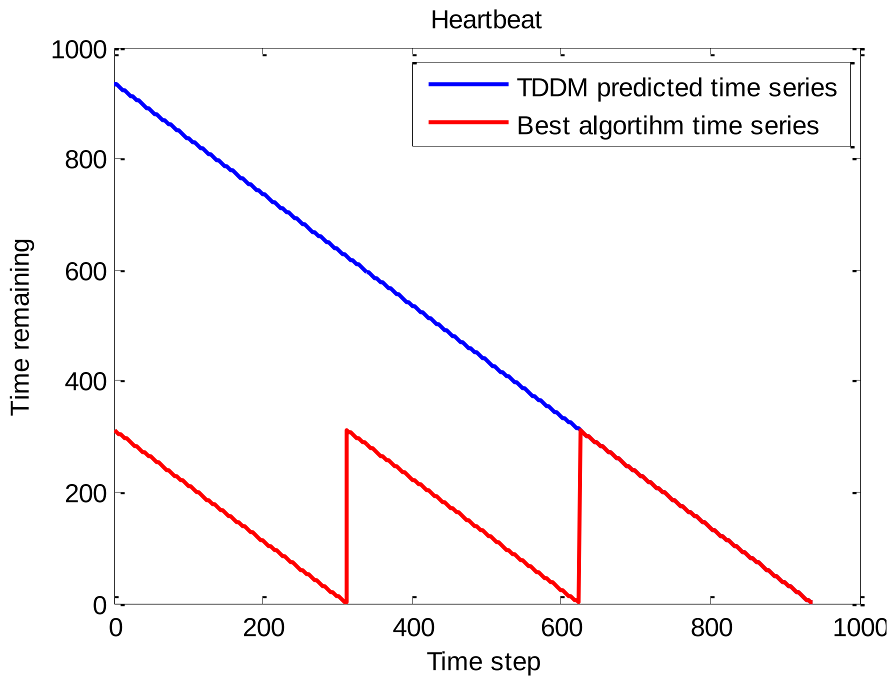


Figure 29: Heartbeat: Best algorithm time series vs TDDM predicted time series

This comparison scenario confirms results in Experiment I that predicting 1 at a precise time step is extremely difficult and of little use independently of the algorithms. Directly learning time remaining produces much better results allowing deeper analysis of the algorithms performances.

Experiment I compared to Experiment II confirms the importance of predicting time remaining versus predicting instantaneous events. This will bring a valuable contribution to time series prediction. It is in fact much better, easier and more useful to predict the time remaining before an event occurs than to predict the precise time step at which it will occurs. This seems in line with animals that can learn timing of events rapidly with a precision proportional to the time interval length [9]. The time remaining perspective gives the algorithms more information on the task to achieve. It is important to mention that the two tasks are completely different for the algorithms except for TDDM which uses the same weights to produce remaining time as well as instantaneous events. VARMA and ESN are producing different weights for both tasks.

When VARMA relies on real past time remaining, the predictions are good, but very bad when it comes to predict instantaneous events which support the event timing forecasting perspective suggested in this thesis. Moreover, in a multiple time steps setup as ESN, VARMA II was unable to learn time remaining appropriately.

TDDM was made for forecasting event timing hence it works the same way on both tasks. Actually, the two tasks are the same for TDDM as it still has the same weights but an important point here is that on Experiment II TDDM was far better in Experiment I. This again supports the idea that it is better to forecast the remaining time than the instantaneous events. Yet, in Experiment I, the conclusion on the TDDM algorithm was that it misses the events but predicts them in general just before they occur as oppose to ESN and VARMA whose predictions do not reach 1 at all in most of the cases.

ESN was poor on Experiment I, yet it outperforms TDDM on Experiment II in many cases. This ability came from the fact that ESN uses internal hidden neurons which allow capturing more features of the data.

5.4 Summary

In the first experiment, the three algorithms VARMA, ESN and TDDM were given as inputs the binary time series and event time series and should output a specific value(1) at the time step where the event occurs. The results of this experiment show that all the algorithms have difficulties in predicting the events time series. The predictions of ESN and VARMA were most of the time 0 and do not reach the value of 1. TDDM succeeds in predicting the values of 1 but most of the time at the wrong place or just before the event occurs. The conclusion of this experience is that the algorithms were not able to predict timing events by predicting the value of 1 where the events occur and 0 otherwise, except for TDDM which was able to fully predict 1 before it occurs. In conclusion, it is indeed very difficult [8] to predict instantaneous occurrences of events. The alternative proposed in this thesis is to predict the time remaining before the events occur instead of the events themselves.

The results of Experiment II show all the algorithms performances were much better as oppose to the first one. TDDM and ESN are able to learn timing events with some errors, with ESN being currently superior, suggesting a need for some form of hidden layers in the TDDM algorithm. Although the VARMA algorithm seems better in predicting the time remaining, we stated that we can't conclude on its ability to predict timing event because in this case it knows the exact previous remaining time through the auto-regression. For that reason, we changed the setup of VARMA to evaluate it on the case of multiple time steps forecasting when it uses its own past forecasts to predict the next time step. The results showed that the error of VARMA II algorithm on all the datasets increased as compared to VARMA and the two others. One can noticed that as soon as VARMA II lost the real time remaining and started to predict based on the past estimated remaining time, the predictions are no longer mimicking the original time series. This allows us to conclude that VARMA is unable to predict the timing events when it uses its own past predictions.

At the end, a comparison of both experiments was attempted and showed that Experiment II is much better that Experiment I. Below is a summary of the ability of the three algorithms to forecast timing event based on our experiments.

Time Series Forecasting Models	Forecasting instantaneous events at precise time steps	Forecasting time remaining before the events occur	Advantages	Drawbacks
VARMA	Unable	Unable on music	→ Easy to understand → Easy to compute →Computationally efficient for low order and linear models	→ Do not learn a high level representation needed to grasp structural regularities at the hundreds of time steps (or more) time scale → Linear models
ESN	Unable	Able	→Independent from non-linearity →Capable of learning long term dependency	→ Difficult to design.
TDDM (Note that in both experiments the same model is learned)	Barely	Able	→ Easy to compute → Computationally efficient	→ Limited representational power

Table 17: Summary on the advantages and limits of the three algorithms regarding the predictions of events

6 Conclusion and Recommendations

6.1 Conclusion

Improving time series forecasting accuracy is an important yet often difficult task facing decision makers in many areas. Therefore, in this thesis, an offline TDDM algorithm based on animal learning has been developed for predicting both instantaneous binary events stream and the time remaining before an event occurs. The performance of TDDM on both tasks was compared with the statistical VARMA and the machine learning ESN time series forecasting algorithms. Basically, the objective was to see whether or not these models could be used to forecast a binary event or a time remaining before an event occurs.

This work began first by a survey of state-of-the art time series prediction methods in statistics and machine learning. A comprehensive review of the advantages and disadvantages for each method and how the algorithms derived from these methods were trained and evaluated has been provided. The forecasting domain has been influenced, for a long time, by linear statistical methods. But, because of their limitation of being linear models, they were not suitable for many applications. Two decades ago machine learning methods have drawn attention and have established themselves as serious contenders to classical statistical models in the forecasting community [42]. The use of hybrid methods resulting by the combination of statistical methods and machine learning methods have become also an increasing phenomenon. All the methods have their benefits and drawbacks depending on the task to learn. Some methods are more adequate for some applications problems than others.

The forecasting timing event concept came from the innate ability of how animals learn. In fact, animals learn the timing of upcoming consecutive events very easily. A possible explanation for this ability is that animals may learn the *"when"* instead of *"what"* by learning the temporal relationship between events. An online TDDM algorithm was developed from this concept by Rivest et al. [1], [10]. The TDDM algorithm works by minimizing the interval timing between specific events. It is derived from drift-diffusion models of decision making. For each specific event, a drift-diffusion process works by accumulating evidences about when that specific event will occur and by learning the event-rate of each input stimulus. The learning is performed through a set of weights and accumulation processes.

The second step of this work was to develop an offline TDDM learning algorithm derived from the online one. It uses the same concept as the TDDM online but instead is based on linear regression methods like VARMA and ESN.

To address the forecasting timing event concept, a preprocessing of the datasets has been made by transforming each datasets respectively into binary time series $\boldsymbol{x}(\boldsymbol{t})$, event time series $\boldsymbol{z}(\boldsymbol{t})$ and time remaining time series $\boldsymbol{y}(\boldsymbol{t})$. The binary series correspond to the duration of the stimuli, the events time series contains events onsets corresponding to when the event occurred and the time remaining series correspond to the time remaining before the event happened.

The experiments consisted first of applying the three algorithms to predict a binary events stream; that is, “1” when the event occurs and “0” otherwise. The algorithms were given as inputs the binary time series $\boldsymbol{x}(\boldsymbol{t})$ and as output the events time series $\boldsymbol{z}(\boldsymbol{t})$. In the second experiment, the algorithms were applied to predict the time remaining $\boldsymbol{y}(\boldsymbol{t})$ before the occurrence of events. The three algorithms have been implemented in an offline mode using linear regression methods. In order to evaluate the overall performance of the algorithms on both tasks by finding the best hyper-parameters, a 10 folds cross-validation procedure on each algorithm has been performed. In addition to that, in Experiment I a confusion matrix and a table of positive predicted instances have been used to see the classification performance of each algorithm. In Experiment II, a root mean squared error has been calculated over the ten cross-validation folds for each dataset and each algorithm. The results for the first and second experiment are summarized below.

Experiment I:

In the first experiment, the three algorithms VARMA, ESN and TDDM were given as inputs the binary time series and event time series and should output a specific value(1) at the time step where the event occurs. The three algorithms were implemented in an offline mode using linear regression methods to make the best possible timing predictions. In order to evaluate the overall performance of the algorithms on the task of predicting timing events, a 10 fold cross-validation procedure on each algorithm was carried out as well as a confusion matrix to see the classification performance of each algorithm. The results of this experiment show that all the algorithms have difficulties in predicting the events time series. The predictions of ESN and VARMA were most of the time 0 and do not reach the value of 1. TDDM succeed in predicting the values of 1 but most of the time at the wrong place or just before the event occurs which is relatively acceptable. The conclusion of this experience is that the algorithms were not able to predict timing events by predicting the value of 1 where the events occur and 0 otherwise. It is indeed very difficult to predict instantaneous occurrences for events. The alternative proposed by in this thesis is to predict the time remaining before the events occurs instead of the events themselves.

Experiment II:

Experiment II consisted of evaluating the three algorithms on the task of forecasting the time remaining before a particular event occurs. The particular events were when a music note will play for the Bach Chorales, when stock prices would go up for the finance dataset, and when the next heartbeat would occur for the arrhythmia database. Each algorithm received the binary inputs time series and the time remaining outputs series. The results showed that all the algorithms performed much better than in Experiment I. The ESN algorithm in general performed better than VARMA and TDDM on Bach Chorales. This ability of ESN may be explained by the use of internal neurons by the ESN which allows it to capture more features of the data. TDDM failed on two time series in finance and music respectively by having its errors zooming off into the millions. The first one in finance occurred when the divisor was too small leading toward a division by zero, and the second, on a piece of music which has a great blank space in its time series. These time series were removed, and new averages were calculated. Nonetheless, TDDM was better than VARMA on the music time series and better than ESN and VARMA on heartbeat. VARMA had less error in finance, but performed poorly on music. The performance of VARMA was mainly explained by the fact that its predictions depend on the fact that the algorithm knows the exact remaining time at the previous time step while TDDM and ESN are generating their estimates based only on the previous inputs. Therefore, a second version of VARMA called VARMA II was developed and tested where the algorithm will rely on its past predictions to estimate the next time step. For this case, the results showed that all errors of VARMA increased considerably specifically on music dataset as compare to the first version. This showed that VARMA is unable, on some task, to learn the high level representation needed to grasp structural regularities. This showed that VARMA is unable to learn a high level representation needed to grasp structural regularities at the hundreds of time steps (or more) time scale.

At the end, a comparison of both experiments was attempted and demonstrated that the task II of predicting the time remaining before the events occur is better to learn than predicting instantaneous events at a precise time step.

The conclusion of both experiments is that it is better for the algorithms to learn to estimate the time remaining as suggested by TDDM forecasting event concept than to predict instantaneous occurrences of events. As for the TDDM algorithm, and the forecasting event timing concept, they are an important step in the forecasting domain but especially in the machine learning one where the goal is to build algorithms that would learn by themselves to produce good results.

6.2 Contributions

The forecasting timing events concept is a step toward developing simple, robust and accurate algorithms that can learn or bring an alternative to the long term dependencies problem while using finite memory.

In summary, this thesis

- Provides a review and a comparison of the existing time series forecasting algorithms in the statistical and machine learning field.
- Develops an offline TDDM algorithm for learning timing events.
- Demonstrates that it is difficult to learn to predict binary events stream.
- Introduces the concept of learning timing events by estimating the time remaining as a perspective for designing better algorithms for forecasting in statistics and machine learning fields.
- Demonstrates the ability of TDDM and ESN algorithms to predict the time remaining before the occurrence of instantaneous events.

6.3 Recommendations for Future Work

The following are ideas for future work:

1. More datasets preprocessing: Investigate more datasets in order to know which algorithm better fit the data or if there are any seasonality or trend which can affect the overall ability of the algorithms
2. Regarding the TDDM algorithm there are several potential problems that would be worthwhile to investigate.
    a. Investigates the stability of TDDM with zeros: As seen in Experiment II, the prediction error of the TDDM algorithm tend to increased highly when it comes to a division by nearly zero and when the time series has some significant blank.

b. The limited representational power: Unlike ANNs, TDDM cannot distinguish between two different temporal patterns that precede the same type of event.
c. Given the performance of ESN, it may be interesting for the TDDM algorithm to have some form of hidden layers or internal memory neurons.

7 Bibliography

[1] F. Rivest, R. Kohar and N. Amadou Boukary, "Learning to Predict Events On-line: A Semi-Markov Model for Reinforcement Learning," in *Autonomous Learning Robots Workshop, At NIPS 2014*, Montreal, 2014.

[2] C.Chatfield, The Analysis of Time series: An Introduction, Boca Raton: Chapman & Hall, 2004.

[3] C. M. Douglas , L. J. Cheryl and K. Murat, Introduction to Time Series Analysis and Forecasting, Wiley, March 2008.

[4] "https://www.otexts.org/fpp/6/1," 2013. [Online].

[5] L. Tsungnan , B. G. Horne, P. Tino and C. L. Giles, "Learning Long-Term Dependencies in NARX Reccurent Neural Networks," *IEEE Transactions on Neural Networks,* vol. 7, no. 6, pp. 1329-1337, 1996.

[6] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent," *Neural Networks, IEEE Transactions on,* vol. 5, no. 2, p. 157–166, 1994.

[7] F. A. Gers, N. N. Schraudolph and J. Schmidhuber, "Learning Precise Timing with LSTM Reccurent Networks," *Journal of Machine Learning and Research,* vol. 3, no. 1, pp. 115-143, 2002.

[8] A. Gers, N. Schraudolph and J. Schmidhuber, "Learning Precise Timing with LSTM Recurrent Networks," *Journal of Machine Learning Research,* vol. 3, no. 1, pp. 115-143, 2002.

[9] Gallistel and . J. Gibbon, "Time, Rate and Conditioning," *Psychology Review,* vol. 107, no. 2, pp. 289-344, 2000.

[10] F. Rivest and Y. Bengio, "Adaptive Drift-Diffusion Process to Learn Time Intervals," *arXiv:1103.2382.,* 2011.

[11] G. Box and G. Jenkins , Time series analysis: forecasting and control, New Jersey: Prentice-Hall, 1976.

[12] Jaeger and Herbert, "Echo state network," *Scholarpedia,* vol. 2, no. 9, p. 2330, 2007.

[13] G. Box, G. Jenkins and G. Reinsel, Time Series Analysis: Forecasting and Control, Wiley, 2008.

[14] X. Zhang, Y. Liu, T. Zhang, A. Young and X. Li, "Comparative Study of Four Time Series Methods in Forecasting Typhoid Fever Incidence in China," *PLoS ONE,* vol. 8, no. 5, May 2013.

[15] K. Kauko and P. Palmroos, "The Delphi method in forecasting financial markets," *International*

Journal of Forecasting, vol. 30, no. 2, pp. 313-327, 2014.

[16] R. J. Hyndman and G. Athanasopoulos, Forecasting: principles and practice, OTexts, 2013.

[17] I. Moghram and S. Rahman, "Analysis and Evaluation of Five Short-Term Load Forecasting Techniques," *IEEE Transactions on Power Systems,* vol. 4, no. 4, pp. 1484-1491, November 1989.

[18] R. Hyndman and J. De Gooijer, "25 years of time series forecasting," *International Journal of Forecasting,* vol. 22, p. 443–473, 2006.

[19] G. Forzieri, F. Castelli and E. Vivoni, "A Predictive Multidimensional Model for Vegetation Anomalies Derived From Remote-Sensing Observations," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 48, no. 4, pp. 1729-1741, April 2010.

[20] A. Fadhilah, S. Suriawati, H. Amir, I. Z.A. and S. Mahendran, "Malaysian day-type load forecasting," *3rd International Conference on Energy and Environment,* pp. 408-411, December 2009.

[21] D. Hill, M. Bridge and D. Infield, "Modelled operation of the Shetland Islands power system comparing computational and human operators' load forecasts," *IEE Proceedings-Generation, Transmission and Distribution,* vol. 142, no. 6, pp. 555-559, November 1995.

[22] G. Mbamalu and M. El-Hawary, "Load forecasting via suboptimal seasonal autoregressive models and iteratively reweighted least squares estimation," *IEEE Transactions on Power Systems,* vol. 8, no. 1, pp. 343-348, February 1993.

[23] L. Yinghui, A. Gribok, W. Ward and J. Reifman, "The Importance of Different Frequency Bands in Predicting Subcutaneous Glucose Concentration in Type 1 Diabetic Patients," *IEEE Transactions on Biomedical Engineering,* vol. 57, no. 8, pp. 1839 - 1846, April 2010.

[24] C. Chun-Chih and L. Hsiang-Wei, "The Advantages of Dynamic Factor Models as Techniques for Forecasting: Evidence from Taiwanese Macroeconomic Data," *International Journal of Economics and Finance,* vol. 3, no. 5, October 2011.

[25] C. Hsueh-Fang, C. Hsueh-Fang, L. Wen-chih and T. Yann-ching, "Forecasting Monthly Sales of Cell-phone Companies - the Use of VAR Model," *Proceedings of the Second International Conference on Innovative Computing, Informatio and Control ,* 2007.

[26] S.-J. Huang and K.-R. Shih, "Short-term load forecasting via ARMA model identification including non-Gaussian process considerations," *IEEE Transactions on Power Systems,* vol. 18, no. 2, pp. 673-679, May 2003.

[27] J. Fan and J. McDonald, "A real-time implementation of short-term load forecasting for distribution

power systems," *IEEE Transactions on Power Systems,* vol. 9, no. 2, pp. 988-994, May 1994.

[28] J.-F. Chen, W.-M. Wang and C.-M. Huang, "Analysis of an adaptive time-series autoregressive moving-average (ARMA) model for short-term load forecasting," *Electric Power Systems Research,* vol. 34, no. 3, p. 187–196, September 1995.

[29] C. Zheng and L. Lei, "The improvement of the forecasting model of short-term traffic flow based on wavelet and ARMA," *8th International Conference on Supply Chain Management and Information Systems (SCMIS),* pp. 1-4, October 2010.

[30] M. Zheng, Z. Yan, Y. Ni, G. Li and Y. Nie, "Electricity price forecasting with confidence-interval estimation through an extended ARIMA approach," *IEE Proceedings- Generation, Transmission and Distribution,* vol. 153, no. 3, pp. 187 - 195, 2006.

[31] J. Contreras, R. Espinola, F. Nogales and A. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Transactions on Power Systems,* vol. 18, no. 3, pp. 1014 - 1020, 2003.

[32] H. Zareipour, C. Canizares, K. Bhattacharya and J. Thomson, "Application of Public-Domain Market Information to Forecast Ontario's Wholesale Electricity Prices," *IEEE Transactions on Power Systems,* vol. 21, no. 4, pp. 1707 - 1717, 2006.

[33] G. Abraham, G. Byrnes and C. Bain, "Short-Term Forecasting of Emergency Inpatient Flow," *IEEE Transactions on Information Technology in Biomedicine,* vol. 13, no. 3, pp. 380 - 388 , 2009.

[34] Q. Li, N. Guo, Y. Han, B. Zhang, S. Q i, Y.-G. Xu, Y. Wei, X. Han and Y. Liu, "Application of an autoregressive integrated moving average model for predicting the incidence of hemorrhagic Fever with renal syndrome," *The American journal of tropical medicine and hygiene,* vol. 87, p. 364–370, 2012.

[35] P. Luz, B. Mendes, C. Codeco, C. Struchiner and G. AP, "Time Series Analysis of Dengue Incidence in Rio de Janeiro, Brazil," *American Journal of Tropical Medicine and Hygiene,* vol. 79, p. 933–93, 2008.

[36] M. Rios, J. Garcia, J. Sanchez and D. Perez, "A statistical analysis of the seasonality in pulmonary tuberculosis," *European Journal of Epidemiology,* vol. 16, p. 483–488, 2000.

[37] G. Bontempi, S. Ben Taieb and Y. Le Borgne, "Machine Learning Strategies for Time Series Forecasting," in *Business Information Processing,* Springer-Verlag, 2013, pp. 62-77.

[38] G. Peter Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing,* vol. 50, p. 159 – 175, 2003.

[39] S. Kim, "Forecasting internet traffic by using seasonal GARCH models," *Journal of*

Communications and Networks, vol. 13, no. 6, pp. 621 - 624, December 2011.

[40] R. Stuart J. and P. Norvig, Artificial Intelligence - A Modern Approach, Prentice Hall, 1994.

[41] "https://commons.wikimedia.org/wiki/User:Chrislb," [Online].

[42] G. Zhang, B. Eddy Patuwo and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting,* vol. 14, p. 35–62, 1998.

[43] W. Yan, "Toward Automatic Time-Series Forecasting Using Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 23, no. 7, July 2012.

[44] K. Hornik, "Multilayer Feedforward Networks are Universal Approximator," *Neural Networks,* vol. 2, pp. 359-366, March 1989.

[45] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computing,* vol. 9, pp. 1735-1780, 1997.

[46] D. S. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," *Technical report,* 1998.

[47] V. Ghate and S. Dudul, "Cascade Neural-Network-Based Fault Classifier for Three-Phase Induction Motor," *IEEE Transactions on Industrial Electronics,* vol. 58, no. 5, pp. 1555 - 1563 , 2010.

[48] B. Boser, I. Guyon and V. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," *ACM Workshop on Computational Learning Theory,* pp. 144-152, 1992.

[49] "Numerical learning library," [Online]. [Accessed 2015].

[50] Siek, M. and D. Solomatine, "Chaotic model with data assimilation using NARX model," pp. 2516-2523, 2009.

[51] S. Mahmoud, A. Lotfi and C. Langensiepen, "Behavioural Pattern Identification and Prediction in Intelligent Environments," *Applied Soft Computing,* vol. 13, no. 4, pp. 1813-1822, April 2013.

[52] D. D. Monner and J. A. Reggia, "A generalized LSTM-like training algorithm for second-order recurrent neural networks," *Neural Networks,* vol. 25, p. 70–83, January 2012.

[53] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen," *Diploma thesis, TU Munich,* 1991.

[54] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transaction on Neural Networks,* vol. 5, no. 2, pp. 157-166, 1994.

[55] F. Gers, "Long Short-Term Memory in Recurrent Neural Networks," *Phd Thesis, Department of Computer Science, Swiss Federal Institute of Technology,* 2001.

[56] M. Lukosevicius, "A Practical Guide to Applying Echo State Networks," in *Neural Networks: Tricks of the Trade*, Springer, 2012.

[57] G. Li, B.-J. Li, X.-G. Yu and C.-T. Cheng, "Echo State Network with Bayesian Regularization for Forecasting Short-Term Power Production of Small Hydropower Plants," *Energies* , vol. 8, no. 10, pp. 12228-12241, 2015.

[58] D. E. Rumelhart, G. Hinton and R. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, pp. 533-536, 1986.

[59] A. Sylvain and A. Celisse, "A survey of cross-validation procedures," *Statistics Surveys,* vol. 4, pp. 40-79, 2010.

[60] G. Zhang, B. Eddy Patuwo and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting,* vol. 14, p. 35–62, 1998.

[61] A. Chen, ,. Leung and D. Hazem, "Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index," *omputers and Operations Research,* vol. 30, p. 901–923, 2003.

[62] V. Berardi and G. Zhang, "An empirical investigation of bias and variance in time series forecasting: Modeling considerations and error evaluation," *IEEE Transactions on Neural Networks,* vol. 14, no. 3, p. 668–679, 2003.

[63] M. Ghiassi and H. Saidane, "A dynamic architecture for artificial neural networks," *Neurocomputing,* vol. 63, p. 397–413, 2005.

[64] H. Hosseini, D. Luo and K. Reynolds, "The comparison of different feed forward neural network architectures for ECG signal diagnosis," *Medical Engineering and Physics,* vol. 28, p. 372–378, 2006.

[65] E. Pisoni, M. Farina, C. Carnevale and L. Piroddi, "Forecasting peak air pollution levels using NARX models," *Engineering Applications of Artificial Intelligence,* vol. 22, no. 4-5, p. 593–602, June 2009.

[66] C. Jiang and F. Song, "Forecasting chaotic time series of exchange rate based on non-linear autoregressive model," *2nd International Conference on Advanced Computer Control (ICACC),* vol. 5, pp. 238-241, 2010.

[67] J. M. P., J. Menezes and G. A. Barreto, "Long-term time series prediction with the NARX network:

An empirical evaluation," *Neurocomputing,* vol. 71, no. 16-18, pp. 3335-3343 , 2008.

[68] A. Andalib and F. Atry, "Multi-step ahead forecasts for electricity prices using NARX: A new approach, a critical analysis of one-step ahead forecasts," *Energy Conversion and Management* , vol. 50, no. 3, pp. 739-747, 2009.

[69] H. Xie, H. Tang and Y.-H. Liao, "Time series prediction based on narx neural networks: An advanced approach," *Proc. of the International Conference on Machine Learning and Cybernetics,* vol. 3, pp. 1275-1279, 2009.

[70] T. Thireou and M. Reczko, "Bidirectional Long Short-Term Memory Networks for Predicting the Subcellular Localization of Eukaryotic Proteins," *IEEE/ACM Transactions On Computational Biology And Bioinformatics,* vol. 4, no. 3, July-September 2007.

[71] J. Schmidhuber, D. Wierstra and F. Gomez, "Evolino: Hybrid Neuroevolution/Optimal Linear Search for Sequence Learning," *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence,* 2005.

[72] D. Eck and J. Schmidhuber, "A First Look at Music Composition using LSTM Recurrent Neural Networks," IDSIA/USI-SUPSI, Switzerland, 2002.

[73] F. Gers, J. Schmidhuber and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," IDSIA-01-99, Switzerland, 1999.

[74] M. Rabin, M. Hossain, M. Ahsan, M. Mollah and M. Rahman, "Sensitivity learning oriented nonmonotonic multi reservoir echo state network for short-term load forecasting," *International Conference on Informatics, Electronics & Vision (ICIEV),* pp. 1-6, 2013.

[75] L. Decai, H. Min and W. Jun, "Chaotic Time Series Prediction Based on a Novel Robust Echo State Network," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 23, no. 5, pp. 787-799, 2012.

[76] M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and ARIMA models for time series forecasting," *Applied Soft Computing,* vol. 11, p. 2664–2667, 2011.

[77] A. Mahdi, A. Hussain, P. Lisbo and D. Al-Jumeily, "The Application of the Neural Network Model Inspired by The Immune System In Financial Time Series Prediction," *Second International Conference on Developments in eSystems Engineering*, 2009.

[78] J. Luxhoj, J. Riis and B. Stensballe, "A hybrid econometric-neural network modeling approach for sales forecasting," *International Journal of Production Economics,* vol. 43, p. 1996, 175–192.

[79] G. Armano, M. Marchesi and A. Murru, "A hybrid genetic-neural architecture for stock indexes

forecasting," *Information Sciences,* vol. 170, p. 3–33, 2005.

[80] "http://www.seattlerobotics.org/ fuzzy logic," [Online]. [Accessed 2014].

[81] O. Castillo and P. Melin, "Simulation and Forecasting Complex Economic Time Series using Neural Networks and Fuzzy Logic," *International Joint Conference on Neural Networks* , vol. 3, pp. 1805-1810, 2001.

[82] I. G. Damousis and P. Dokopoulos, "A fuzzy expert system for the forecasting of wind speed and power generation in wind farms," *22nd IEEE Power Engineering Society International Conference on Power Industry Computer Applications,* pp. 63 - 69, 2001.

[83] C. W. Potter and M. Negnevitsky, "ery Short-Term Wind Forecasting for Tasmanian Power Generation," *IEEE Transactions on Power Systems,* vol. 21, no. 2, pp. 965-972, 2006.

[84] N. K. Kasabov and Q. Song, "Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction," *IEEE Transactions on Fuzzy Systems,* vol. 10, no. 2, pp. 144-154, 2002.

[85] C. Wu, J. Ho and D. T. Lee, "Travel-time prediction with support vector regression," *IEEE Transaction on Intelligence Transport System,* vol. 5, no. 4, p. 276–281, 2004.

[86] C. Zhang and H. Hu, "Using PSO algorithm to evolve an optimum input subset for a SVM in time series forecasting," *IEEE Conference on Systems, Man, and Cybernetics,* vol. 4, p. 3793–3796, 2005.

[87] C.-C. Hsu, C.-H. Wu, S.-C. Chen and K.-L. Peng, "Dynamically optimizing parameters in support vector regression: An application of electricity load forecasting," *39th Annual Hawaii International Conference on System Sciences,* vol. 2, pp. 1-8, January 2006.

[88] B. Dong, C. Cao and S. E. Lee, "Applying support vector machines to predict building energy consumption in tropical region," *Energy Buildings,* vol. 37, no. 5, p. 545–553, 2005.

[89] P. Pai and C. Lin, "A hybrid ARIMA and support vector machines model in stock price forecasting," *Omega 33,* p. 497–505, 2005.

[90] N. I. Sapankevych and R. Sankar, "Time series predictions using support vector machine: A survey," *IEEE Computational Intelligence Magazine,* vol. 4, no. 2, p. 24 – 38, 2009.

[91] G. C. Tiao and R. S. Tsay, "Some advances in non-linear and adaptive modelling in time-series.," *Journal of Forecasting,* vol. 13, no. 2, pp. 109-131, 1994.

[92] N. A. Gershenfeld and A. S. Weigend, Time series prediction: Forecasting the future and

understanding the past, Addison-Wesley, 1994.

[93] K. Bache and M. Lichman, "UCI Machine Learning Repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Bach+Chorales.

[94] "NASDAQ Stock Market," 2013. [Online]. Available: http://www.nasdaq.com/quotes/.

[95] "MIT-BIH Arrhythmia Database Directory," [Online]. Available: http://www.physionet.org/physiobank/database/html/mitdbdir/mitdbdir.htm.

[96] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark and H. E. . . . Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals.," *Circulation,* vol. 101, no. 23, pp. e215-e220, 2000.

[97] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH Arrhythmia Database," *IEEE Engineering in Medicine and Biology Magazine,* vol. 20, no. 3, pp. 45-50, 2001.

[98] L. Citi and R. Barbieri, "A Real-Time Automated Point-Process Method for the Detection and Correction of Erroneous and Ectopic Heartbeats," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING,* vol. 59, no. 10, pp. 2828-2830, 2012.

[99] H. Jaeger, "Simple and very simple Matlab toolbox for Echo State Networks," http://reservoir-computing.org/, 2009.

[100] R. Kohavi and F. Provost, "Special Issue on Applications of Machine Learning and the Knowledge Discovery Process," *Machine Learning,* vol. 30, pp. 271-274, 1998.