Integrated Multi-Model Framework for Adaptive Multiple Time-Series Analysis and Modelling

Harya Widiputra

A thesis submitted to Auckland University of Technology
in fulfilment of the requirement for
the degree of Doctor of Philosophy

November, 2011

School of Computing & Mathematical Sciences

Primary Supervisor: Professor Nikola Kasabov
Secondary Supervisor: Dr. Russel Pears

This thesis was proofread by Diana Kassabova

Contents

List of Figures

List of Tables

List of Abbreviations

kNN	k-Nearest Neighbour , page 50, 51, 52, 53, 128, 131
ADALINE	ADAptive LInear NEuron, a type of neural network introduced by Widrow & Hoff , page 30, 160, 162, 167
ANFIS	Adaptive-Network-Based Fuzzy Inference System , page 58
AR	Autoregressive model , page 18, 19
ARIMA	Autoregressive Integrated Moving Average model , page 21, 22
ARMA	Autoregressive Moving Average model also known as the Box-Jenkins model , page 20, 21, 22
CAST	Clustering Affinity Search Technique , page 97
DENFIS	Dyamic Evolving Neuro-Fuzzy Inference System , page 36, 37, 40, 85, 87, 126, 153
DIN	Dynamic Interaction Network , page 4, 7, 8, 25, 55, 56, 68, 72, 76, 81, 85, 120, 147, 160, 165, 167, 168, 170, 171, 177, 178, 181, 190, 194, 202, 203, 205, 206, 211, 213, 214, 216, 219–222, 224
ECM	Evolving Clustering Method , page 38, 39–42, 93, 99, 107, 133
ECMc	Off-line model of the ECM with constrained minimisation , page 41, 45
EFS	Evolving Neuro-Fuzzy System , page 46
EM	Expectation-Maximisation algorithm , page 7, 55, 68, 70, 73, 219

ESN	Echo State Network , page 58
eSNN	Evolving Spiking Neural Networks , page 226
eTS	EFS with Takagi-Sugeno inference system , page 47
FDL	Finite distributed lag model , page 17, 18
GA	Genetic Algorithm , page 225
GRN	Gene Regulatory Network , page 1
IMMF	Integrated Multi-Model Framework , page 5, 8, 158, 160, 165, 167, 168, 170, 171, 177, 178, 181, 202, 203, 205, 206, 210, 222, 225
LMS	Least Mean Square learning algorithm of ADALINE , page 162
LRP	Long-run propensity property of an FDL model , page 18
LTM	Localised Trend Model , page 4, 8, 93, 95, 115, 147, 160, 165, 167, 168, 170, 171, 177, 178, 181, 194, 202, 203, 205, 206, 213, 216, 220–222, 224, 225
MAE	Mean Absolute Error , page 168, 173
MLP	Multi-Layer Perceptron , page 2, 4, 14, 31, 32, 34, 85, 126, 131, 153, 184, 210, 222, 226
MLP-BP	Multi-Layer Perceptron with backpropagation learning rules , page 32
MLR	Multiple Linear Regression , page 4, 24, 85, 126, 131, 153, 184, 210, 222

mTNFI	Multiple Transductive Neuro-Fuzzy Inference System , page 4, 8, 130, 139, 143, 147, 148, 160, 165, 167, 168, 170, 171, 177, 178, 181, 198, 202, 203, 205, 206, 215, 221, 222, 224, 225
NFI	Neuro-Fuzzy Inference method for transductive reasoning , page 8, 130–132, 139, 221
OLS	Ordinary least square estimator , page 54, 106, 109
peSNN	Probabilistic Evolving Spiking Neural Networks , page 227
RBFN	Radial Basis Function Network , page 35
RMSE	Root Mean Square Error , page 84, 85, 121, 125, 147, 153, 168, 173, 179, 181, 190, 205, 210
RNOMC	Rooted Normalised One-Minus Correlation , page 95
SNN	Spiking Neural Networks , page 226
SVM	Support Vector Machines , page 2, 14, 58, 89, 157
VAR	Vector Autoregressive model , page 25
VARMA	Vector Autoregressive Moving Average model , page 25, 26
vQEA	Versatile Quantum Inspired Evolutionary Algorithm , page 225
WKNN	Weighted k-Nearest Neighbour , page 50, 52, 53, 128, 130
WLS	Weighted linear least-square estimator , page 38, 54, 16
WRLS	Weighted recursive linear least-square estimator , page 38, 39

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Auckland, November, 2011

Harya Widiputra

Acknowledgements

This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable knowledge, experience and assistance in the preparation and completion of this study.

First and foremost, my utmost gratitude goes to my primary supervisor, *Professor Nikola Kasabov*, for his ideas, guidance and support towards my PhD study. And also for the opportunity that allowed me to gain better insights and understanding in the area of knowledge engineering and discovery.

I would especially like to thank my secondary supervisor, *Dr. Russel Pears*, whose sincerity and encouragement I will never forget. He has always been available for both academic and personal discussions and has always encouraged and helped me with improvising the quality of my research work.

It was also a pleasure working with *Dr. Antoanetta Serguieva*, it was an inquisitive experience as she provided me with critical insight within my work pertaining to economics that enabled me to better infer some outcomes from the study.

In particular, I am also thankful to my friend and colleague, *Joyce D'Mello*, for all the great support she has offered throughout my PhD study. She has always supported me through all the hurdles and obstacles I encountered during the completion of this research work.

I would like to thank past and present members of the Knowledge Engineering and Discovery Research Institute (KEDRI), who have directly or indirectly given different perspectives on my work and shared their ideas freely. All the discussions with fellow researchers and students at KEDRI have been amusing and inspirational especially: *Dr. Qun Song, Dr. Peter Hwang, Dr. Stefan Schliebs, Dr. Raphael Hu, Kshitij Dhoble, Haza Hamed, Marin Karaivanov, Gary Chen* and *Lei Song*.

I am also grateful to the *School of Computing and Mathematical Sciences of Auckland University of Technology* for providing me with the scholarship, facilities and environment which proved to be helpful towards the completion of my research.

Last but not the least, *my family* and the one above all of us, *God Almighty*, for bequeathing me with strength and courage to stand up and walk again when I was down on my knees, thank you so much.

This work is dedicated to my beloved wife, ***Mira Purnama Dewi***, without whose caring support and help it would not have been possible, my son, ***Randitya Bumi Widiputra*** for all the inspiring smiles, laughs and cries, my daughter, ***Anindya Sasikirana Dewi*** for all the beams and giggles to come, and to my parents for their unconditional love and never ending prayers.

Abstract

The topic of time-series prediction has been very well researched in studies of dynamic systems. However, most studies in the field have focused more on predicting movement of a single time-series only, whilst prediction of multiple time-series based on the dynamics of interactions between variables has received little attention. This PhD study is concerned with advances in the analysis, modelling and prediction of dynamic systems with respect to multiple time-series.

The main objective of this thesis is to develop novel adaptive methods to discover and model dynamic pattern of interactions in multiple time-series to not only predict their future values but also to extract knowledge about their joint movement. Being able to adaptively model dynamic pattern of interactions between multiple variables is expected to lead to a better understanding of the dynamic system under evaluation. Additionally, as new patterns of interaction emerge intermittently, the models to be developed are also required to have the ability to evolve and learn incrementally.

To realise these objectives three distinct methods of multiple time-series analysis are developed based on different concepts of learning (inductive and transductive reasoning) which are capable of modelling the dynamics of interaction between variables in a specific setting. As each approach addresses the problem of multiple time-series analysis and modelling from a different perspective and since each has its own predictive power, an integrated multi-model framework that incorporates the different approaches through a dynamic contribution adjustment function is also proposed.

In this study, the proposed methods have been applied for the analysis, modelling and prediction of two real world case studies: (1) multiple interactive stock markets in the Asia Pacific region and (2) weather conditions at different locations in New Zealand. Results from the two case studies suggest that

(a) the proposed methods are capable of modelling dynamic pattern of interactions between variables and (b) that the idea of including the nature and strength of relationship between a collection of time sensitive variables (which are related to each other) into a prediction model appears to be beneficial to the problem of multiple time-series prediction.

The research is based on three main information processing principles proposed by Prof. Kasabov in the period of 1998-2011: (1) The principle of evolving, adaptive connectionist systems; (2) The principle of integrated global, local and personalised modelling; and (3) The principle of dynamic interaction network. Using these three principles 3 new computational methods are proposed and tested on both synthetic and real data for adaptive incremental learning and knowledge discovery from multiple time-series data. A software system was developed to implement the methods. Solutions to two specific case study problems are proposed and tested - financial time-series prediction; climate events prediction.

Chapter 1

Introduction

1.1 Motivation and Objective

Previous studies have revealed that dynamic relationships between series exist in multiple time-series data relating to real world phenomena in the Biological and Economic domains. It has also been established that being governed by these interrelationships multiple time-series move together through time. For instance, it is well known that the movement of a stock market index in a specific country is affected by the movements of other stock market indexes across the world or in that particular region (Antoniou, Pescetto, & Violaris, 2003; Chen & Poon, 2007; Chowdhury, 1994; Collins & Biekpe, 2003; Forbes & Rigobon, 1999; Masih & Masih, 2001). Likewise, in a Gene Regulatory Network (GRN) the expression level of a gene is determined by its time varying interactions with other genes (Davidson, 2006; Davidson & Erwin, 2006; Levine & Davidson, 2005; X. Li, Chen, Li, & Zhang, 2010).

However, even though time-series modelling and prediction have been extensively researched and some prominent methods have been developed in the machine learning and data mining arenas such as the multi-layer perceptron (MLP) (Azmy, El Gayar, Atiya, & El-Shishiny, 2009; El Dajani, Miquel, Maison-Blanche, & Rubel, 2003; Koskela, Lehtokangas, Saarinen, & Kaski, 1996; Shiblee, Kalra, & Chandra, 2009; Teo, Wang, & Lin, 2001) and the Support Vector Machines (SVM) (Camastra & Filippone, 2007; Müller et al., 1999; Sapankevych & Sankar, 2009; Thissen, Brakel, Weijer, Melssen, &

Buydens, 2003) little attention has been paid to the dynamics of interactions between multiple time-series. Furthermore, there has been no significant research so far on developing a method that can predict multiple time-series simultaneously by considering the existence of dynamic interactions between them.

Some of the studies that took into account multiple time-series variables were the following: Han, Fan, and Xi (2005) which adopted the Elman Recurrent Neural Network structure to predict levels of sunspots and run-off of the Yellow river in China; Zhanggui, Yau, and Fu (1999) performed a stock price prediction using a pattern classification method; Ankenbrand and Tomassini (1996) used neural networks to predict multivariate financial time-series; B. Liu and Liu (2002) proposed a multivariate time-series prediction with temporal classification; H. Yang, Chan, and King (2002) implemented the SVM for stock market prediction; T. Kim and Adali (2003) performed multivariate time-series approximation using MLP. However, these studies did not demonstrate an ability of their models to continuously adopt to changing dynamics of multiple time-series, to capture and model dynamic relationships between multiple variables and to predict their future values simultaneously.

Therefore, the task of extracting and modelling interdependencies or patterns of interaction between variables over time has become a challenge, in particular for research in information science. Being able to accomplish such missions is expected to lead us to the understanding of how observed variables in a specific environment move together, inhibit each other, are connected to each other and how their profile of relationship changes dynamically over time. Additionally, it will also be of help to identify important variables that have the most influencing power in governing the state of a system and predicting variables expression values at future moments.

In relation to this, the objectives of the work conducted in this PhD study are defined as follows:

- To develop novel methods for multiple time-series analysis and modelling in order to discover dynamic patterns of interaction between multiple variables that can be utilised to predict their future values simultaneously. The models are required to have the capability to adapt, evolve and learn as new observations or problems become available;

- To represent information about interactions between multiple variables in a form that can be easily understood and helpful to comprehend further the dynamics of relationship between variables of interest of the system being observed;

- To propose an integrated multi-model framework that incorporates different methods of multiple time-series analysis and modelling for multiple time-series prediction.

- To assess the value of extracting and exploiting relationships between multiple variables in prediction when the variables concerned are influencing each other in a dynamic fashion.

- To apply the proposed methods and integrated multi-model framework to two different case studies of real world phenomena that require these models. The first case study is the analysis, modelling and prediction of selected stock markets in the Asia Pacific region, whilst the second case study is the analysis, modelling and prediction of weather conditions at different locations in New Zealand.

To realise the objectives outlined above, three distinct models of multiple time-series analysis are proposed in this thesis. The models are developed based on different concepts of learning, i.e. the global, local and instance-based reasoning and are capable of modelling the dynamics of interaction between variables in a specific setting such as in Finance, Biology, Weather, etc. Utilising the three distinct models as main building blocks, an integrated

multi-model framework is proposed and assessed in this study with synthetic data and two case studies of real world data. The proposed framework integrates different types of knowledge and predictive power of each model.

Additionally, to evaluate the performance of the proposed methods an assessment is to be done by conducting a comparative analysis between the proposed methods and other methods which have been widely applied on single time-series prediction, i.e. multiple linear regression (MLR), MLP, random walk models, etc., on a pre-generated synthetic data and two different case studies. Both MLR and MLP offer acceptable degree of time-series prediction accuracy and therefore they are still being widely used and trusted to solve many real world problems. This actuality serves as the main reason of choosing these models as the main competitors to the proposed methods.

1.2 Main Contributions

Throughout the completion of the PhD study, six main contributions have been made:

1. Development of a global model for multiple time-series analysis and modelling, called here Dynamic Interaction Network (DIN), which is capable of extracting the dynamics of interaction between variables over time.

2. Development of a local model, called here Localised Trend Model (LTM), which constructs a repository of profiles of relationships and recurring trends from a set of multiple time-series data whose structure will dynamically evolve over time, for the analysis and modelling of multiple time-series.

3. Development of a transductive/nearest neighbour learning model named Multivariate Transductive Neuro-Fuzzy Inference System (mTNFI) which

dynamically creates a new model for multiple time-series prediction whenever a new prediction is required to be made.

4. Development of a multi-model framework named the Integrated Multi-Model Framework (IMMF) that integrates different types and levels of knowledge from the proposed global, local and transductive models for multiple time-series prediction.

5. Implementation of the proposed global, local, and transductive models and the integrated multi-model framework for the analysis, modelling and prediction of multiple interactive stock markets in the Asia Pacific region.

6. Implementation of the proposed global, local, transductive model and integrated multi-model framework for the analysis, modelling and prediction of weather condition on different sites in New Zealand.

In addition a number of publications including some book chapters, journal papers and conference papers have also been produced and are listed below:

Book Chapters

1. Widiputra, H., Pears R., Kasabov, N. Dynamic Learning of Multiple Time-Series in a Non-Stationary Environment in Learning in Non-Stationary Environments: Methods and Applications, edited by Moamar Sayed-Mouchaweh and Edwin Lughofer. Springer, in press.

2. Widiputra, H., Pears R., Kasabov, N. Kalman Filter to Estimate Dynamic and Important Patterns of Interaction between Multiple Variables in Kalman Filtering, edited by Joaquín M. Gomez (2011), pp. 289-320. Nova Science, New York.

Journal Papers

1. Widiputra, H., Pears R., Kasabov, N. Dynamic Interaction Networks versus Localized Trends Model for Multiple Time-Series Prediction. Cybernetics and Systems 42 (2011), pp. 1-24.

2. Widiputra, H., Pears, R., Serguieva, A., and Kasabov, N. Dynamic Interaction Networks in Modelling and Predicting the Behaviour of Multiple Interactive Stock Markets. Journal of Intelligent Systems in Accounting, Finance, and Management, 16 (2009), pp. 189-205.

Conference Papers

1. Widiputra, H., Pears R., Kasabov, N. Multiple Time-series Prediction Through Multiple Time-Series Relationships Profiling and Clustered Recurring Trends. Lecture Notes in Computer Science 6635 (2011), pp. 161-172. Springer Verlaag.

2. Widiputra, H. Building an Integrated Multi-model Framework for Multiple Time-series Prediction. Proceeding of the 8th New Zealand Computer Science Research Student Conference (NZCSRSC). Victoria University of Wellington, New Zealand (2010).

3. Lukmanto, L., Widiputra, H., and Lukas. Dynamic Interaction Networks to Model Interactive Patterns of International Stock Markets. World Academy of Science, Engineering and Technology 59 (2009), pp. 257-261.

4. Widiputra, H., Kho, H., Lukas, Pears, R., and Kasabov, N. A Novel Evolving Clustering Algorithm with Polynomial Regression for Chaotic Time-Series Prediction. Lecture Notes in Computer Science 5864 (2009), pp. 114-121. Springer Verlaag.

5. Widiputra, H., Pears, R., Kasabov, N. Personalised Modelling for Multiple Time-Series Data Prediction: A Preliminary Investigation in Asia Pacific Stock Market Indexes Movement. Lecture Notes in Computer Science 5506-Part 1 (2009), pp. 1231-1238. Springer Verlaag.

1.3 Thesis Structure

The thesis is structured as follows:

- CHAPTER 1 presents an introduction to the PhD study and a brief description of multiple time-series problem of modelling and predicting simultaneous movements of a collection of time sensitive variables that are related to each other.

- CHAPTER 2 presents a review of the fundamentals in univariate and multiple time-series analysis, methods of time-series analysis from the machine learning and data mining arena, and types of learning process that are used in time-series analysis and modelling. It also reviews a set of existing methods that are adopted as part of the proposed models or which inspired the development of the proposed models.

- CHAPTER 3 presents a global modelling approach of multiple time-series analysis named the Dynamic Interaction Network and denoted as DIN. It employs the Kalman Filter and the Expectation-Maximisation (EM) algorithm to capture and model the dynamics of interaction between variables. The extracted interaction model is then put in place for simultaneous multiple time-series prediction. A synthetic data set that consists of four interrelated time-series was generated and used to evaluate the performance of the proposed method.

- CHAPTER 4 presents a local modelling approach of multiple time-series analysis named the Localised Trend Model and denoted as LTM. The synthetic data introduced in Chapter 3 is again being utilised in this chapter to evaluate LTM's capability to construct and dynamically maintain a knowledge repository of profiles of relationships and recurring trends between variables for multiple time-series prediction.

- CHAPTER 5 presents nearest neighbour learning model for multiple time-series analysis named the Multivariate Transductive Neuro-Fuzzy Inference System and denoted as mTNFI. The mTNFI was inspired by and developed further the Neuro-Fuzzy Inference method for transductive reasoning denoted as NFI proposed by Q. Song and Kasabov (2005). It was originally designed to work only for single time-series prediction or to solve classification problems.

- CHAPTER 6 presents an Integrated Multi-Model Framework denoted as IMMF, for multiple time-series prediction that implements a modified version of the multi-model framework proposed by Kasabov (2007b). The integrated framework utilises the proposed global (Chapter 3), local (Chapter 4) and transductive model (Chapter 5) of multiple time-series analysis to allow contrasting views on a single problem and the assimilation of different types and levels of knowledge for multiple time-series prediction.

- CHAPTER 7 presents the implementation of the global (DIN), local (LTM), transductive model (mTNFI) and the integrated framework (IMMF), as proposed in Chapters 3, 4, 5 and 6 respectively, for the analysis and modelling of multiple interactive stock markets in the Asia Pacific region.

- CHAPTER 8 presents the implementation of the global (DIN), local

(LTM), transductive model (mTNFI) and the integrated framework (IMMF), as proposed in Chapters 3, 4, 5 and 6 respectively, for the analysis and modelling of weather conditions in New Zealand.

- CHAPTER 9 presents the discussion and conclusion of the study and suggests future work.

Chapter 2

Fundamentals of Time-Series Analysis and Modelling

2.1 What is Time-Series Data?

Time-series data is a train of numerical data points in sequential order, usually recorded in uniform intervals. Thus, a time-series consists of a sequence of numbers collected at regular intervals over a period of time. In statistics, signal processing, econometrics and mathematical finance, a time-series is described as a sequence of data points, measured typically at successive times spaced at uniform time intervals. Time-series data often arise when monitoring any real world phenomenon, e.g. the global economy conditions, global weather system, ecological system, etc. Some common examples of time-series are the daily closing value of an equity market, e.g. the Dow Jones index or the annual flow volume of the Nile River in Egypt.

An obvious characteristic of time-series data that distinguishes it from cross-sectional data is temporal ordering (Wooldridge, 2006). For example, given a time-series data set on employment, the minimum wage, and other economic variables for a certain country, it is possible to learn that the data for year 1970 immediately precedes the data for 1971. A basic concept in analysing time-series data for a real world phenomenon is then to recognise that the past can affect the future, but not vice versa. Emphasising the ordering properties of time-series data, Table 2.1 gives a sample of time-series

Table 2.1

Partial Listing of Data on U.S. Inflation and Unemployment Rates, 1948-2003 (The Federal Reserve Archival System for Economic Research, 2011)

Year	Inflation	Unemployment
1948	8.1	3.8
1949	−1.2	5.9
1950	1.3	5.3
1951	7.9	3.3
⋮	⋮	⋮
1998	1.6	4.5
1999	2.2	4.2
2000	3.4	4.0
2001	2.8	4.7
2002	1.6	5.8
2003	2.3	6.0

data of inflation and unemployment rates from the United States of America.

Another difference between cross-sectional and time-series data is more subtle. The cross-sectional data is viewed as random outcomes, which is fairly straightforward such as: a different sample drawn from the population will generally yield different values of the independent and dependent variables. Yet, how should randomness in time-series data be considered?

Certainly, observations taken from the field of economics meet the basic requirement of being random variables. For instance, today's closing value of an equity market is not known until the end of the trading day. On the other hand, time-series data of weather conditions also satisfy this intuitive requirement, as the level of air pressure, wind speed, air humidity, etc. in a

certain place at 6.00 am tomorrow is not yet known today. Since the outcomes of these variables are not foreknown, they should clearly be viewed as random variables.

Formally, a sequence of random variables indexed by time is called a stochastic process or a time-series process (Wooldridge, 2006; Kirchgässner & Wolters, 2007). When time-series data is collected, one possible outcome (or realisation) of the stochastic process is obtained. Only a single realisation can be observed, since it is not possible to go back in time and start the process over again. However, if certain conditions in history had been different, generally a different realisation for the stochastic process would have been obtained, and this is why a time-series data set is considered as the collection of the outcomes of a set of random variables.

2.2 Time-Series Analysis and Modelling

Time-series analysis comprises methods for analysing time-series data in order to extract meaningful statistics and other characteristics of the data. There are two main goals of time-series analysis: (a) identifying the nature of the phenomenon represented by the sequence of observations, and (b) forecasting (predicting future values of the time-series variable). Both of these goals require that the pattern of observed time-series data is identified and formally described. Once the pattern is established, it can be interpreted and integrated with other data.

Regardless of the depth of the understanding and validity of the interpretation of the phenomenon, the identified pattern can be extrapolated to predict future events. Time-series forecasting is the use of a model to foresee future events based on known past events, that is to predict data points before they are measured. An example of time-series forecasting in econometrics is predicting the opening price of a stock based on its past performance.

The essential characteristic of data modelling using methods for time-series analysis is the following: time-series analysis considers the fact that data points recorded over time at uniform intervals may have an internal structure such as autocorrelation, trend or seasonal variation that should be accounted for in the model (Wooldridge, 2006). In addition, the actuality that time-series data have a natural temporal ordering makes time-series analysis distinct from other common data analysis problems, in which there is no natural ordering of the observations, e.g. explaining people's wages by reference to their educational level, where the individuals' data could be entered in any order. Furthermore, time-series analysis is also distinct from spatial data analysis where the observations typically relate to geographical locations, e.g. accounting for house prices by suburb.

A time-series model will generally reflect the fact that observations closer together in time will be more closely related than observations further apart. Therefore, it is essential in time-series analysis to be able to build a model that can dynamically evolve its structure in relation to current behaviour of the system. This chapter of the thesis gives a brief overview of some of the more widely used techniques in the rich and rapidly growing field of time-series analysis and modelling.

2.3 Univariate Time-Series Analysis

To make a choice from a set of alternatives, decision makers at all structural levels in different fields often need predictions of variables involved in the problem. If time-series observations are available for a variable of interest and the data from the past contain information about the future development of a variable, it is then plausible to develop a model that can be used to forecast the future values of the variable and to assist the decision making process.

For instance, being able to forecast the monthly unemployment rate from

past experience, helps an economy analyst to understand that in some country or region a high unemployment rate in one month tends to be followed by a high rate in the next month. Assuming that the tendency prevails in future periods, prediction models can then be made based on current and past data. This approach to forecasting may be expressed in a mathematical model as follows: Let x_t denote the value of the variable of interest in period t, then a prediction for period $t + h$, made at the end of period t, may have the form:

$$\hat{x}_{t+h} = f(x_t, x_{t-1}, ...), \tag{2.1}$$

where $f(.)$ denotes some suitable functions of the past observations $x_t, x_{t-1}, ...$ Equation 2.1 forms the general model of *univariate time-series analysis*, in which the upcoming value of a variable of interest is explained only by previous observations of the variable itself.

One major objective of univariate time-series analysis (Wei, 2005) is to specify sensible forms of functions $f(.)$. In many applications, linear functions have been used so that, for example, $\hat{x}_{t+h} = v + \alpha_1 x_t + \alpha_2 x_{t-1} + ...$, where v is a constant value, while $\alpha_1, \alpha_2, ...$ are the coefficients that give weight to current and past observations.

However, in dealing with data from real world phenomenon, often a value of a variable of interest is being influenced not only by its current and past values but also by current and past values of other variables. If this is the case, then the model in Equation 2.1 needs to be generalised to a multivariate model. This type of modelling is known as the *multivariate time-series* analysis and will be outlined in the next section.

2.3.1 Static Models

As in the general context of regression, in a time-series regression model, time-series analysts deal with stochastic relationships, in which error terms are included in the specification of the model (Wooldridge, 2006). The simplest

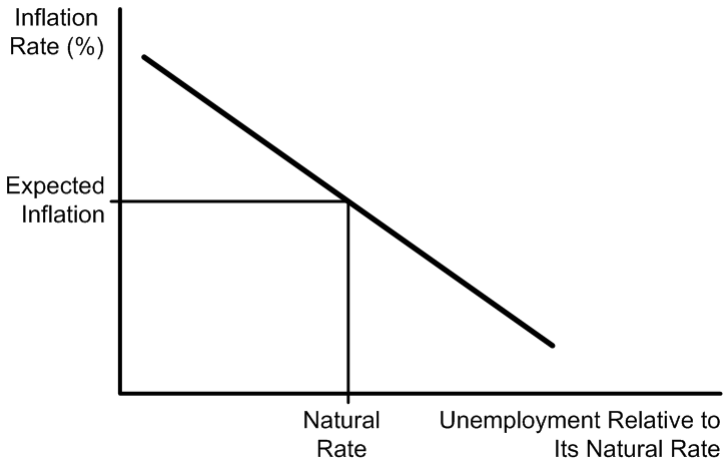


Figure 2.1. Static Phillips curve (DeLong, 2002).

form of a time-series regression model that explains the relationship between two variables of time-series data, y_t and x_t, is called *static model*:

$$y_t = \beta_0 + \beta_1 x_t + \varepsilon_t \tag{2.2}$$

A static model is a simple linear regression model where relationship between two variables x and y is being modelled simultaneously. Usually, a static model is postulated when one-unit increase in x at time t is believed to have immediate effect on y with a size of β_1. A static model cannot be written with lag operators since there are obviously no lagged terms.

An example of a static model is the *static Phillips curve* (Mankiw, 2006), given by $inf_t = \beta_0 + \beta_1 unem_t + \varepsilon_t$, where inf_t is the annual inflation rate and $unem_t$ is the unemployment rate. The Phillips curve assumes a constant natural rate of unemployment and constant inflation expectations, and it can be used to study the contemporaneous trade-off between inflation and unemployment. Figure 2.1 illustrates the static Phillips curve.

2.3.2 Finite Distributed Lag Models

A distributed lag model is a model for time-series analysis in which a regression equation is used to predict current values of a dependent variable based on both the current values and the lagged (past period) values of an explanatory variable (Borghers & Wessa, 2011; Wooldridge, 2006). Therefore, unlike in a static model, in a *finite distributed lag* (FDL) model one or more variables are considered to affect y with a certain number of lag as described by this equation:

$$y_t = \alpha_0 + \delta_0 x_t + \delta_1 x_{t-1} + \delta_2 x_{t-2} + \varepsilon_t \tag{2.3}$$

This is an FDL of order two. More generally, an FDL model of order q will include q lags of x as described by this equation:

$$y_t = \alpha_0 + \delta_0 x_t + \delta_1 x_{t-1} + ... + \delta_q x_{t-q} + \varepsilon_t, \tag{2.4}$$

where δ_0 is the immediate change in y due to the one-unit increase in x at time t, δ_0 is usually called the *impact propensity* or *impact multiplier*. For a temporary, one-period change, y returns to its original level in period $q + 1$. Equations 2.3 and 2.4 show that just like the static model, FDL also falls under the category of univariate time-series regression model, in which error terms are included in the specification of the model.

In FDL models, the sum of the coefficients on current and lagged x, $\delta_0 + \delta_1 + ... + \delta_q$, is the long-run change in y given a constant increase in x. This property of the FDL model is called the *long-run propensity* (LRP) or *long-run multiplier* and is often of interest in distributed lag models. An example of an FDL model for annual data of a real world scenario is: $int_t = 1.6 + 0.48inf_t - 0.15inf_{t-1} + 0.32inf_{t-2} + \varepsilon_t$ where int is an interest rate and inf is the inflation rate. The impact propensity is 0.48, while the long-run propensity is $0.48 - 0.15 + 0.32 = 0.65$.

2.3.3 Autoregressive Model: AR

A common approach for modelling univariate time-series is the autoregressive (AR) model (Kirchgässner & Wolters, 2007; Wooldridge, 2006). The AR model is a special case of FDL model, in which the dependent variable is being explained by the lagged values of itself. The mathematical model of AR is described as follows:

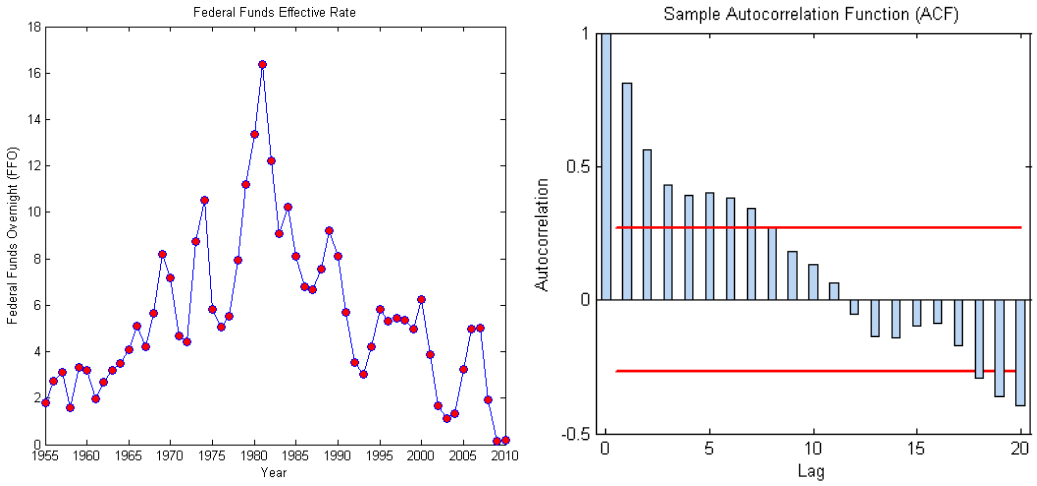
$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \varepsilon_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + ... + \phi_p x_{t-p} + \varepsilon_t \tag{2.5}$$

where ϕ_i represents the autoregression coefficients, x_t is the series under investigation, and p is the order (length) of the model which is generally much less than the length of the complete observations. The noise term or residue, ε_t in the above, is almost always assumed to be Gaussian white noise. The AR model suggests that the current term of the series can be estimated by a linear weighted sum of previous terms in the series. Accordingly, the weights are the autoregression coefficients represented by ϕ.

An AR(p) model is actually simply a linear regression of the current value of the series against one or more prior values of the series. The key challenge in AR analysis is to derive the "best" values for ϕ_i given a series x_t. To deal with this issue various methods of parameter estimation, including the ordinary least square, can be put into place for the analysis of AR models. Additionally, AR models give a straightforward interpretation. For example, an AR(1) process is a first-order one process, meaning that only the immediately previous value has a direct effect on the current value as described by this equation below:

$$x_t = \phi_1 x_{t-1} + \varepsilon_t \tag{2.6}$$

To identify whether an AR model is the appropriate approach to model the time-series under examination, the *autocorrelation* plot is put in place. Autocorrelation plots are a commonly-used tool for checking randomness in a



(a) US Federal fund effective rate 1955-2010 (Federal Reserve Statistical Release, 2011)

(b) Autocorrelation plot

Figure 2.2. Plot of the United State of America Federal fund effective rate from 1955 to 2010 and its autocorrelation function (ACF).

data set. This randomness is ascertained by computing autocorrelations for data values at different time lags. If the time-series is indeed random, such autocorrelations should be near zero for any and all time-lag separations. If it is non-random, then one or more of the autocorrelations will be significantly non-zero. When a strong positive autocorrelation is detected in a time-series data, then it indicates that the data comes from an underlying autoregressive model, therefore it is assumed that an AR model is the appropriate analysis tool. An example of data with strong positive autocorrelation is illustrated in Figure 2.2b.

In addition, *partial autocorrelation* plot is also a commonly used tool for model identification in AR models. The partial autocorrelation at lag k is the autocorrelation between x_t and x_{t-k} that is not accounted for by lags 1 through $k-1$. Specifically, partial autocorrelation plots are useful to identify the order of an AR model, p.

The partial autocorrelation of an AR(p) process is zero at lag $p+1, p+$

2, ... If the sample autocorrelation plot indicates that an AR model may be appropriate, then the sample partial autocorrelation plot is examined to help identify the order of the AR model, by locating a point on the plot where the partial autocorrelations essentially become zero.

2.3.4 Box-Jenkins Model: ARMA

The Box-Jenkins model refers to the model with p autoregressive terms and q moving average terms (Box & Jenkins, 1970; Hamilton, 1994; Kirchgässner & Wolters, 2007; Pankratz, 1983). This model is a combination of the AR(p) and MA(q) models and therefore is also known as the ARMA(p, q) model described by this equation below:

$$\begin{aligned} x_t &= \phi_1 x_{t-1} + ... + \phi_p x_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} \\ &= \varepsilon_t + \textstyle\sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \end{aligned} \tag{2.7}$$

where ε_t are white noise, (p, q) is the order of the model, ϕ_i are the parameters or coefficients of the autoregressive part of the model, and θ_i are the coefficients of the moving average part.

Different representation of the autoregression coefficients that allows all the polynomials involving the lag operator to appear in a similar form was used in Box, Jenkins, and Reinsel (2008). Using this representation, the ARMA model then can be written in the following form:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) x_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t \tag{2.8}$$

where L is the lag operator (see Appendix C).

After the order of the model (p, q) is chosen, the coefficient of both the autoregressive and moving average part can be estimated using any method of parameter estimation that minimises the error term, such as the ordinary least square. Finding appropriate values of p and q in the ARMA(p, q) model can

be facilitated by plotting the partial autocorrelation functions for an estimate of p, and likewise using the autocorrelation functions for an estimate of q.

The ARMA model is appropriate for modelling a system whose behaviour is a function of itself and a series of unobserved shocks. The autoregressive part (AR) will represent the system own behaviour, whilst the moving average part (MA) models the influencing unobserved shocks. An example of such system is illustrated by movement of stock prices, as they exhibit technical trending and may experience shocks by fundamental information. Nevertheless, when looking at long term data, econometricians or financial analysts tend to opt for an AR(p) model for simplicity.

2.3.5 ARIMA Model

An autoregressive integrated moving average (ARIMA) model (Borghers & Wessa, 2011; Helfenstein, 2005) is a generalisation of an ARMA model. ARIMA model is applied to time-series data either to better understand the nature of the series under observation or to predict its future values. In particular, the ARIMA model is applicable to some cases in which the data shows evidence on *non-stationarity*, where an initial differencing step is applied to remove the non-stationarity. This process is corresponding to the integrated part (I) of the model, where the AR and MA part are the autoregressive and moving average part respectively.

The model is generally referred to as an ARIMA(p, d, q) model where p, d, and q are integers greater than or equal to zero and refer to the order of the autoregressive, integrated, and moving average parts of the model respectively. Additionally, an important part of the Box-Jenkins approach (ARMA) to time-series modelling is derived from the ARIMA model. Assume that the polynomial $(1 - \sum_{i=1}^{p} \phi_i L^i)$ of the ARMA model in Equation 2.8 has a unitary

root of multiplicity d, then the AR part can be rewritten as follows:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) = \left(1 - \sum_{i=1}^{p-d} \phi_i L^i\right)(1 - L)^d \tag{2.9}$$

Accordingly, an ARIMA(p, d, q) process then expresses this polynomial factorisation property as follows:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right)(1 - L)^d x_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\varepsilon_t \tag{2.10}$$

and thus can be thought of as a particular case of an ARMA$(p+d, q)$ process having the autoregressive polynomial with some roots in the unity. For this reason any ARIMA model with $d > 0$ is not "wide sense stationary".

Theoretically, ARIMA model is the most general class of models for time-series prediction which can be stationarised by transformations such as differencing and lagging. In fact, ARIMA model can be seen as the fine-tuned version of the random walk model, in which lags of differenced series and/or lags of the prediction errors were added to the prediction model to remove any last traces of autocorrelation from the prediction errors. A special case of ARIMA model is an ARIMA $(0, 1, 0)$ model given by:

$$x_t = x_{t-1} + \varepsilon_t \tag{2.11}$$

which is simply a random walk model.

2.4 Multiple Time-Series Analysis

When dealing with variables from a real world phenomenon such as economic, weather, ecological, etc., often the value of one variable is not only related to its prior values but depends on past values of other variables as well. For instance, consumption expenditures of a household may depend on variables such as income, interest rates, and investment expenditures. If all of these

variables are related to the consumption expenditures, it is then logical to consider their conditions in predicting consumption expenditures. In other words, denoting the related variables by $x_{1,t}, x_{2,t}, ..., x_{k,t}$, prediction of $x_{1,t+h}$ at the end of period t may be represented by the following form:

$$\hat{x}_{1,t+h} = f_1(x_{1,t}, x_{2,t}, ..., x_{k,t}, x_{1,t-1}, x_{2,t-1}, ..., x_{k,t-1}, x_{1,t-2}, ...). \tag{2.12}$$

Similarly, a prediction for the second variable may be based on past values of all variables in the system. More generally, a prediction of the k^{th} variable may be expressed by this equation:

$$\hat{x}_{k,t+h} = f_k(x_{1,t}, ..., x_{k,t}, x_{1,t-1}, ..., x_{k,t-1}, ...) \tag{2.13}$$

A set of time-series $x_{k,t},\ k = 1, ..., k;\ t = 1, ..., t$, is called a multiple time-series and Equation 2.13 expresses the prediction of $x_{k,t+h}$ as a function of a multiple time-series (Wei, 2005). In analogy with the univariate time-series analysis, one major objective of multiple time-series analysis is to determine suitable functions $f_1, ..., f_q$ that may be used to predict future values of the variables with good properties.

Additionally, it is also often of interest to learn about the inter-relationships between a number of variables. For instance, in a system consisting of investment, income, and consumption one may want to know about the likely impact of a change in income. What will be the present and future implications of such an event for consumption and for investment? Under what conditions can the effect of an increase in income be isolated and traced through the system? Alternatively, given a particular subject matter theory, is it consistent with the relations implied by a multiple time-series model which is developed with the help of statistical tools?

These and other questions regarding the structure of the relationships between the variables involved are occasionally investigated in the context of multiple time-series analysis. Thus, obtaining insight into the dynamic

structure of a system is a further objective of multiple time-series analysis conducted in this PhD study.

2.4.1 Multiple Linear Regression

A linear regression process within the time-series context assumes that a dependent time-series, y_t, is being influenced by a collection of explanatory or independent series, $x_{1,t}, x_{2,t}, ..., x_{k,t}$. This relationship between the dependent and independent variables can be expressed through the multiple linear regression model as follows:

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + ... + \beta_k x_{k,t} + \varepsilon_t \tag{2.14}$$

where β_0, being the intercept of the regression, is a constant, $\beta_1, \beta_2, ..., \beta_q$ are the regression coefficients and ε_t is a random error or white noise, ordinarily assumed to be white with mean zero and variance σ_ε^2.

The linear model described in Equation 2.14 can be written in a more general form by defining the column vector $\mathbf{x}_t = (1, x_{1,t}, x_{2,t}, ..., x_{k,t})'$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, ..., \beta_q)$, resulting in this equation:

$$y_t = \boldsymbol{\beta}\mathbf{x}_t + \varepsilon_t \tag{2.15}$$

In this study, a comparative analysis is conducted of the proposed methods of multiple time-series analysis and modelling against multiple linear regression (MLR) as it has been widely used to model and predict real world data.

2.4.2 Vector AR Model: VAR

The vector autoregressive (VAR) model (Kirchgässner & Wolters, 2007; Wooldridge, 2006; Zivot & Wang, 2006) is one of the most successful, flexible, and easy to use model from the statistics domain for multiple time-series analysis. It is a natural extension of the univariate autoregressive (AR) model to

multiple time-series analysis in the sense that one single dependent variable in AR x_t is now replaced by a vector of dependent variables. The VAR model has been widely used to describe the dynamic behaviour of economic and financial time-series (Zivot & Wang, 2006). In addition, it often provides predictions that are better than those obtained from univariate time-series models and elaborated theory-based simultaneous equations models. Forecasts from VAR models are quite flexible because they can be made conditional on the potential future paths of specified variables in the model.

The VAR model of multiple time-series is described as follows: Let $\mathbf{x}_t = (x_{1,t}, x_{2,t}, ..., x_{k,t})'$ denote an $(k \times 1)$ vector of time series variables, then the basic p-lag vector autoregressive (VAR(p)) model has the form:

$$\mathbf{x}_t = \mathbf{A}_1\mathbf{x}_{t-1} + \mathbf{A}_2\mathbf{x}_{t-2} + ... + \mathbf{A}_p\mathbf{x}_{t-p} + \boldsymbol{\varepsilon}_t \tag{2.16}$$

where $\mathbf{A}_i$ $(i = 1, ..., p)$ are $(k \times k)$ coefficient matrices and $\boldsymbol{\varepsilon}_t$ is an $(k \times 1)$ unobservable zero mean white noise vector process (serially uncorrelated or independent).

One of the computational approaches to multiple time-series analysis and modelling proposed in this thesis, the DIN, extends further the basic concept of VAR model by enabling the on-line learning property in the model.

2.4.3 Vector ARMA Model: VARMA

The VARMA model (Box et al., 2008; Reinsel, 1997) extends the standard finite order VAR model by allowing the error terms, $\boldsymbol{\varepsilon}_t$, to be autocorrelated rather than the white noise. The autocorrelation structure is assumed to be of a relatively simple type so that $\boldsymbol{\varepsilon}_t$ has a finite order moving average (MA) representation

$$\boldsymbol{\varepsilon}_t = \mathbf{u}_t + \mathbf{M}_1\mathbf{u}_{t-1} + ... + \mathbf{M}_q\mathbf{u}_{t-q} \tag{2.17}$$

where, as usual, $\mathbf{u}_t$ is zero mean white noise. A finite order VAR process with finite order MA error term is called a VARMA (vector autoregressive moving

average) process.

Allowing finite order VAR processes to have finite order MA instead of white noise error terms, results in the broad and flexible class of VARMA processes. The general form of a process from this class with VAR order p and MA order q is

$$\mathbf{x}_t = \mathbf{A}_1\mathbf{x}_{t-1} + ... + \mathbf{A}_p\mathbf{x}_{t-p} + \mathbf{u}_t + \mathbf{M}_1\mathbf{u}_{t-1} + ... + \mathbf{M}_q\mathbf{u}_{t-q} \tag{2.18}$$

Such a process is called a VARMA(p, q) process. As before, $\mathbf{u}_t$ is zero mean white noise.

2.5 Machine Learning Methods for Time-Series Analysis

Machine learning is a scientific discipline that is concerned with the design and development of algorithms that allow computers to evolve behaviours based on empirical data, such as sensor data, observation data or databases. A learner can take advantage of examples (data) to capture characteristics of interest of their unknown underlying probability distribution or behaviour. Data can be seen as examples that illustrate relations between observed variables. A major focus of machine learning research is to automatically learn to recognise complex patterns and make intelligent decisions based on data.

The following subsections outline some state-of-the-art machine learning algorithms that have been used for or closely related to time-series analysis.

2.5.1 Artificial Neural Networks

An *artificial neural network* simply known as a *neural network* or *connectionist model* is a biologically inspired computational model that consists of processing elements (neurons) and connections (weights) which constitute the

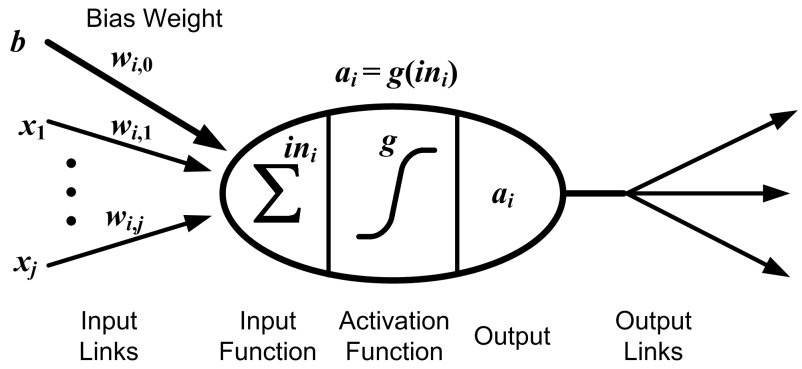


Figure 2.3. A simple mathematical model for a neuron devised by McCulloch and Pitts in 1943 (Russell & Norvig, 1995).

neuronal structure with training and recall algorithm attached to it as stated by Kasabov (1996).

The first mathematical model of a neuron was proposed by McCulloch and Pitts (1943) and is illustrated in Figure 2.3. The figure depicts that the neuron model will "fire" when a linear combination of inputs exceeds some threshold. Based on this initial model, much more detailed and realistic models have been developed over the years, both for neurons and for larger systems in the brain, leading to the modern field of computational neuroscience.

The usefulness of neural network models lies in the fact that they can be used to deduce a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical. The tasks neural networks are applied to tend to fall within these domains of study:

- Function approximation or regression analysis, including time series prediction (El Dajani et al., 2003; T. Kim & Adali, 2003; Koskela et al., 1996; Shiblee et al., 2009).

- Classification, including pattern and sequence recognition (Calcagno et al., 2010; Costa, Filippi, & Pasero, 2005; Isa & Mamat, 2011; Rossi &

Conan-Guez, 2005).

- Data processing, including filtering and clustering (Charalampidis & Muldrey, 2009; Kahla, Faraj, Castanie, & Hoffmann, 1994; H. Lee & Tsoi, 1995).

- Robotics, including directing manipulators and computer digital control (Arroyo, Gonzalo, & Moreno, 1991; Sartori, Passino, & Antsaklis, 1992; Teixeira, Braga, & Menezes, 2000).

There are two main categories of neural network structures: feed-forward networks and recurrent networks. A feed-forward network represents a function of its current input; thus, it has no internal state other than weights themselves.

A recurrent network, on the other hand, feeds its outputs back into its own inputs. This means that the activation levels of the network form a dynamical system that may reach a stable state or exhibit oscillations or even chaotic behaviour. Moreover, the response of a network to a given input depends on its initial state, that may depend on previous inputs. Hence, recurrent networks (unlike feed-forward networks) can support short-term memory (Kasabov, 1996);(Russell & Norvig, 1995). This makes them more interesting as models of the brain, but also more difficult to understand.

2.5.1.1 The Perceptron

One of the first models that was developed based on the McCulloch and Pitts (1943) neuron model was a neural network called the *perceptron* (Kasabov, 1996; Rosenblatt, 1958; Russell & Norvig, 1995). Originally, the neurons in the perceptron have a simple summation input function and a *hardlim* activation function or linear threshold activation function. Additionally, the inputs are real numbers while outputs are binary. Therefore, the most common

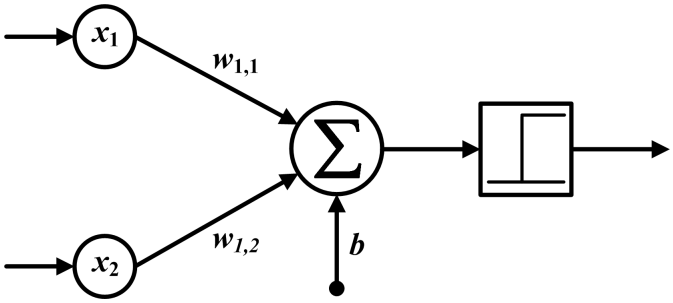


Figure 2.4. Illustration of 2-input perceptron.

application of the perceptron is as a binary classifier. A simple structure of the perceptron is illustrated in Figure 2.4.

In the training phase, a perceptron learns only when it misclassified an input vector from the training examples. When a misclassification is recognised, the perceptron changes the connection weights in such a way that if the desired output is 1, while the produced output is 0, the connection weights of the output neuron is increased and vice versa. A learning algorithm for the perceptron is given below (Kasabov, 1996):

- Step 1: initialise all connection weights w_{ij} where $i = 0, 1, 2, ..., n; j = 1, 2, ..., m$; and n is the size of input variables and m is the number of output neuron, to zero or a small random value;
- Step 2: define a learning rate α, where $0 < \alpha < 1$;
- Step 3: for input vector $\mathbf{x}$ calculate the net input signal u_j to each output neuron j using this equation:

$$u_j = \sum x_i w_{ij},$$

  where $x_0 = 1$ is the bias;
- Step 4: apply a hardlim activation function to the net input signal

defined as follows:

$$o_j = \begin{cases} 1 & \text{if } u_j > threshold \\ 0 & \text{otherwise} \end{cases}$$

Another possibility is to apply a linear thresholding function;

- Step 5: calculate the error for each output neuron as follows:

$$Err_j = y_j - o_j,$$

where y_j is the desired output and o_j is the output produced by neuron j;

- Step 6: adjust each connection weight w_{ij} using this formula:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha x_i . Err_j$$

- Step 7: repeat Step 3 to 6 until the error vector **Err** is sufficiently low, in other words until the perceptron converges.

In 1960, Widrow and Hoff proposed a different formula to calculate the output error during training (Widrow & Hoff, 1960). This learning algorithm was then applied to a neural machine called the ADALINE (ADAptive LInear NEuron).

Using the learning algorithm outlined above, if there are sufficient examples, the perceptron can learn to approximate the training examples and converge after a number of training iterations (*epochs*). However, Minsky and Papert (1969) discovered an important limitation of the perceptron, that is it can only be used as classifier to problems with linearly separable classes. In the case where the problems are not linearly separable, e.g. the XOR problem, the perceptron fails to converge.

Despite this limitation, perceptrons are still used for solving problems due to their simple architecture and the unconditional convergence (when dealing with linearly separable classes).

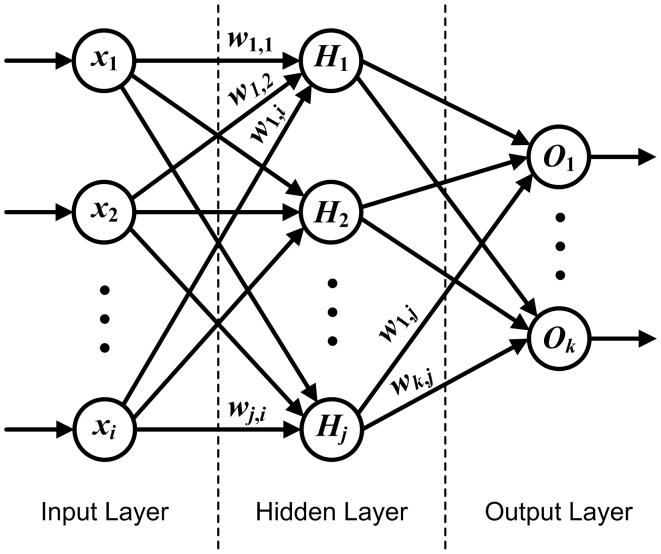


Figure 2.5. Illustration of multi-layer perceptron with a single hidden layer. When applied to perform time-series prediction, $x_1, x_2, ..., x_i$ represent input variables $x_t, x_{t-1}, ..., xt - n$ while $o_1, ..., o_k$ represent the predictions i.e. $x_{t+1}, ..., x_{t+m}$.

2.5.1.2 Multi-Layer Perceptron

To deal with the linear separability limitation of the perceptron, the multi-layer perceptron (MLP) was introduced. The MLP basic structure consists of an input layer, at least one hidden (intermediate) layer and an output layer, in which each neuron from one layer is connected to all neurons in the next layer. Figure 2.5 illustrates the structure of an MLP with a single hidden layer.

The MLP (Hornik, Stinchcombe, & White, 1989) is a feed forward neural network model that is capable of learning the relationships between input and output of a data set by adjusting the connection weights through layers of perceptrons. The MLP was only put into practice when learning algorithms were developed. One of them is the well-known *backpropagation algorithm* whose full name is the *error backpropagation algorithm* (Amari, 1990; Rumelhart, Hinton, & Williams, 1986; Werbos, 1974). When utilising the error backpropagation algorithm which minimises the difference between the desired output

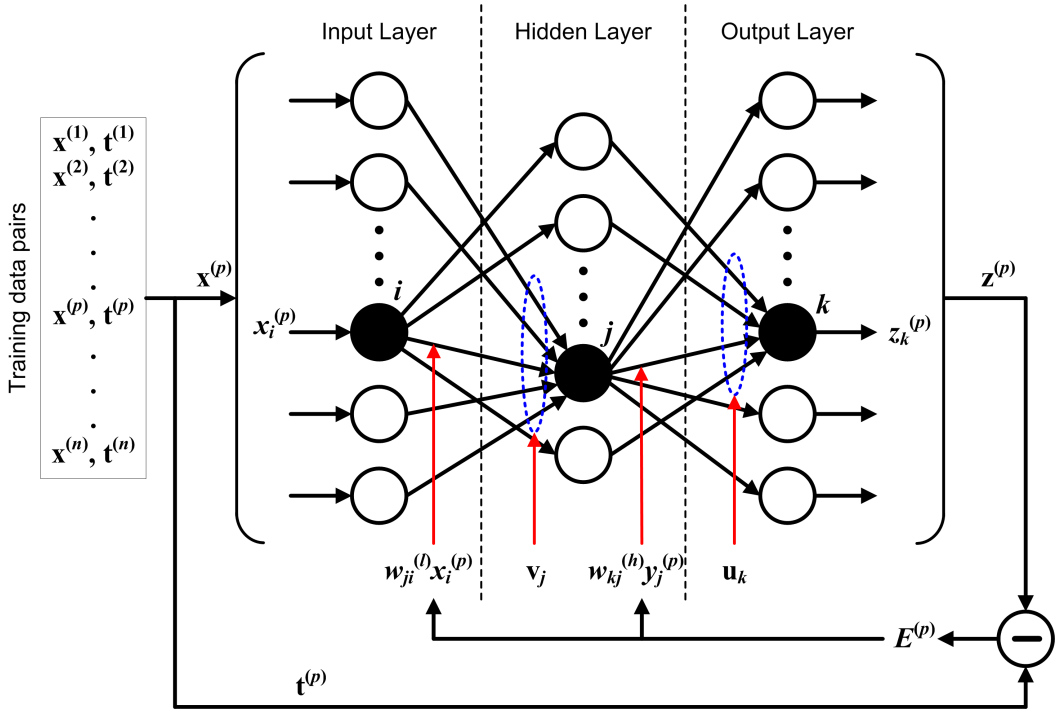


Figure 2.6. Multi-layer perceptron with backpropagation.

and the predicted one, the connection weights in all layers of the network are adjusted.

In the MLP the sigmoid function is commonly used as the activation function for neurons in the hidden layer, while at the output layer either sigmoid or sum function can be used depending on the type of the problem. When the network is first constructed, the connection weights are initially randomised and then adjusted by the error backpropagation algorithm.

Generally, the MLP works by feeding forward the input data from the input layer through the hidden layer and output layer, before finally the predicted output is derived. If the result differs from the desired output, the error is then propagated back through the layers. The connection weights and the activation functions are then adjusted based on a pre-defined learning rate to minimise the error. A number of iterations of this process on whole training samples is performed until the network converges.

The MLP with backpropagation algorithm (MLP-BP) is one of the most widely used models for classification or prediction. A detailed learning algorithm for MLP-BP in relation to Figure 2.6 is described below:

- Step 1: initialise the error threshold err_{max}, the maximum number of training epochs ep_{max}, the learning rate α where $0 < \alpha < 1$ and the connection weights to zero or randomise small value;

- Step 2: select a data pair $(\mathbf{x}^{(p)}, \mathbf{t}^{(p)})$ from the training samples where $x^{(p)}$ is the p^{th} input vector and $\mathbf{t}^{(p)}$ is the p^{th} target value, set the training epochs: $ep \leftarrow ep + 1$ and calculate output of the j^{th} hidden node $y_j^{(p)}$ using the sigmoid function as follows:

$$y_j^{(p)} = f(v_j) = \frac{1}{1 + \exp(-v_j)}$$

and calculate output of the k^{th} output node $z_k^{(p)}$ as follows:

$$z_k^{(p)} = f(u_k) = \frac{1}{1 + \exp(-u_k)}$$

where

$$v_j = \sum_i w_{ji}^{(l)} x_i^{(p)} \quad \text{and} \quad u_k = \sum_j w_{kj}^{(h)} y_j^{(p)}$$

- Step 3: calculate the error for each output neuron as follows:

$$err_k^{(p)} = t_k^{(p)} - z_k^{(p)},$$

adjust the connection weights between the hidden layer and the output layer $\mathbf{w}^{(h)}$,

$$\begin{aligned} \delta_k^{(h)} &= err_k^{(p)} z_k^{(p)} \left(1 - z_k^{(p)}\right) \\ \Delta w_{kj}^{(h)} &= \alpha \delta_k^{(h)} y_j^p \\ w_{kj}^{(h)} &\leftarrow w_{kj}^{(h)} + \Delta w_{kj}^{(h)} \end{aligned} \tag{2.19}$$

afterwards adjust the connection weights between the input layer and the hidden layer $\mathbf{w}^{(l)}$ based on the new $\mathbf{w}^{(h)}$,

$$\begin{aligned} \delta_j^{(l)} &= \left(\sum_1^k \delta_k^{(h)} w_{kj}^{(h)}\right) y_j^{(p)} \left(1 - y_j^{(p)}\right) \\ \Delta w_{ji}^{(l)} &= \alpha \delta_j^{(l)} x_i^p \\ w_{ji}^{(l)} &\leftarrow w_{ji}^{(l)} + \Delta w_{ji}^{(l)} \end{aligned} \tag{2.20}$$

This process of connection weights optimisation will go further backward if any additional layer exists in the network structure;

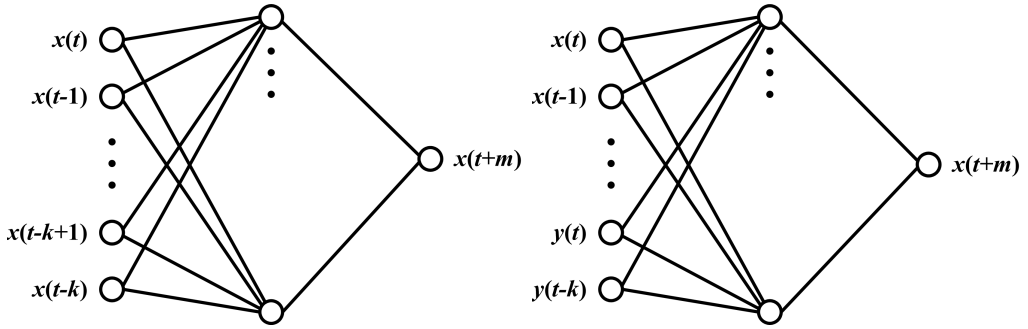
- Step 4: if $err < err_{max}$ or $ep > ep_{max}$ then the algorithm terminates, otherwise repeat Step 2 to 4.

In the listed learning algorithm above, i is the number of input variables, j is the number of hidden nodes and k is the number of output target and therefore also the number of output neurons.

One of the main features of MLP is that it can be used as a universal approximator as stated by (Cybenko, 1989; Hornik et al., 1989): An MLP with one hidden layer can approximate any continuous functions to any desired accuracy, subject to sufficient number of hidden nodes.

Application of the MLP for univariate time-series prediction and multivariate time-series prediction is illustrated in Figure 2.7a and 2.7b. Here, the predicted value of variable x at future time is based on k previous values of either only variable x in a case of univariate prediction or both x and y for multivariate prediction (Kasabov, 1996).

The MLP has been widely used to solve the problem of time-series prediction in real world data. In relation to this, in order to assess the performance of our proposed methods a comparative analysis of prediction accuracy against the MLP is also conducted in this study.



(a) MLP for univariate time-series prediction (b) MLP for multivariate time-series prediction

Figure 2.7. Illustration of implementation of MLP for time-series prediction.

2.5.1.3 Radial Basis Function Network

Radial Basis Function Network (RBFN) is a two-layer feed forward neural network that employs Gaussian function as the activation function in the hidden nodes. In the output node, for prediction problems a weighted sum function is used to aggregate the output from the hidden nodes, whilst for classification problems a sigmoid function can used. RBFN has been proposed and used by a number of studies (Lucks & Oki, 1999; Marinaro & Scarpetta, 2000; Poggio, 1994). A general architecture of the RBFN is given in Figure 2.8.

In RBFN's architecture (as illustrated in Figure 2.8) w_i is the activation level of the i^{th} hidden node defined by

$$c_i = R_i(\mathbf{x}) = \exp\left(-\frac{0.5\left\|\mathbf{x}-\mathbf{m}_i\right\|^2}{\sigma_i^2}\right), \tag{2.21}$$

where $\mathbf{x}$ is the input vector, $\mathbf{m}_i$ is the centre of a Gaussian function and σ_i is the bandwidth of the i^{th} hidden node. Different to MLP, in RBFN there is no connection weights between the input layer and the hidden layer. The output of an RBFN is the weighted sum of the output values from all hidden nodes as given below:

$$o(\mathbf{x}) = \sum_{i=1}^{n} w_i c_i \tag{2.22}$$

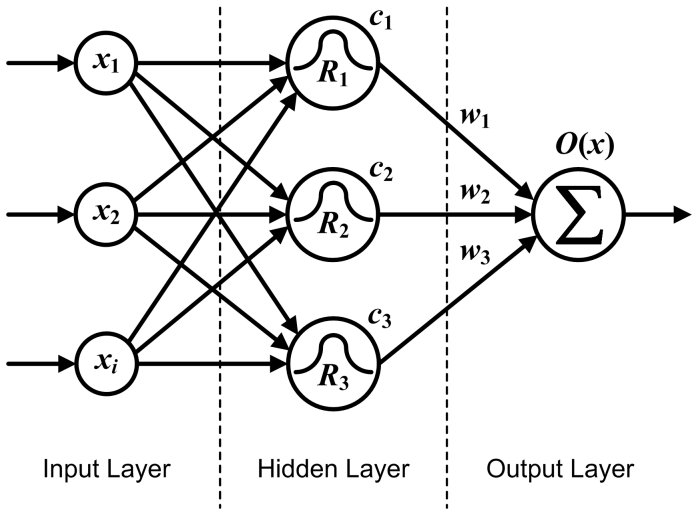


Figure 2.8. Illustration of Radial Basis Function Network structure.

where w_i is the output value associated with the i^{th} hidden node. It can also be seen as the connection weights between the i^{th} hidden node and the output node.

When being applied for time-series modelling and prediction, the structures illustrated in Figure 2.7 can also be implemented by the RBFN. In these structures the input variables are current or past observations of a time-series whilst the outputs are predicted values in the upcoming time-points.

2.5.2 Dynamic Evolving Neuro-Fuzzy Inference System: DENFIS

The Dynamic Evolving Neuro-Fuzzy Inference System denoted as DENFIS was proposed by Kasabov and Song (2002). DENFIS is a fuzzy inference systems for adaptive on-line learning and dynamic single time-series analysis and prediction. DENFIS evolves through incremental hybrid (supervised/unsupervised) learning and accommodates new input data, including new features, new classes, etc. through local element tuning. New fuzzy rules are created and updated during the operation of the system. At each time

moment the output of DENFIS is calculated through a fuzzy inference system based on m-most activated fuzzy rules that are dynamically chosen from a fuzzy rule set.

DENFIS uses Takagi-Sugeno type fuzzy inference engine (Takagi & Sugeno, 1985). The inference engine used in DENFIS is composed of m fuzzy rules indicated as follows:

$$\begin{cases} \text{if } x_1 \text{ is } R_{11} \text{ and } x_2 \text{ is } R_{12} \text{ and ... and } x_q \text{ is } R_{1q}, \text{ then } y \text{ is } f_1(x_1, x_2, ..., x_q) \\ \text{if } x_1 \text{ is } R_{21} \text{ and } x_2 \text{ is } R_{22} \text{ and ... and } x_q \text{ is } R_{2q}, \text{ then } y \text{ is } f_2(x_1, x_2, ..., x_q) \\ ... \\ \text{if } x_1 \text{ is } R_{m1} \text{ and } x_2 \text{ is } R_{m2} \text{ and ... and } x_q \text{ is } R_{mq}, \text{ then } y \text{ is } f_m(x_1, x_2, ..., x_q) \end{cases}$$

where "x_j is R_{ij}", $i = 1, 2, ..., m;\ j = 1, 2, ..., q$, are $m \times q$ fuzzy propositions as m antecedents from m fuzzy rules respectively; $x_j,\ j = 1, 2, ..., q$, are antecedent variables defined over universes of discourse $X_j,\ j = 1, 2, ..., q$, and $R_{ij},\ i = 1, 2, ..., m;\ j = 1, 2, ..., q$, are fuzzy sets defined by their fuzzy membership functions $\mu R_{ij} : X_j \to [0, 1],\ i = 1, 2, ..., m;\ j = 1, 2, ..., q$. In the consequent parts, y is a consequent variable, and linear or polynomial functions $f_i, i = 1, 2, ..., m$, are employed.

In DENFIS all fuzzy membership functions are triangular type functions that depend on three parameters as given by the following equation:

$$\mu(x) = f(x, a, b, c) = \begin{cases} 0, & c \le x \le a \\ \dfrac{x - a}{b - 1}, & a \le x \le b \\ \dfrac{c - x}{c - b}, & b \le x \le c \end{cases} \tag{2.23}$$

where b is the value of the cluster centre on the x dimension; $a = b - d \times Dthr$ and $c = b + d \times Dthr, d = 1 \sim 2$; the threshold value, $Dthr$ is a clustering parameter.

If the consequent functions are crisp constants, i.e. $f_i(x_1, x_2, ..., x_q) = C_i,\ i = 1, 2, ..., m$, such a system is called a zero-order Takagi-Sugeno type

fuzzy inference system. The system is called a first-order Takagi-Sugeno type fuzzy inference system if $f_i(x_1, x_2, ..., x_q),\ i = 1, 2, ..., m$, are linear functions (Kasabov & Song, 2002). If these functions are non-linear functions, it is called high-order Takagi-Sugeno fuzzy inference system (Kasabov & Song, 2002).

For an input vector $x^0 = [x_1^0, x_2^0, ..., x_q^0]$, the result of inference y^0 (the output of the system) is the weighted average of each rule's output indicated as follows:

$$y^0 = \frac{\sum_{i=1}^{m} w_i f_i(x_1^0, x_2^0, ..., x_q^0)}{\sum_{i=1}^{m} w_i} \tag{2.24}$$

where

$$w_i = \prod_{j=1}^{q} \mu R_{ij}(x_j^0);\ i = 1, 2, ..., m;\ j = 1, 2, ..., q.$$

2.5.2.1 Learning Processes in DENFIS

In DENFIS, the rules are created and updated at the same time with the input space partitioning using on-line Evolving Clustering Method (ECM) that was designed specifically for DENFIS (Q. Song & Kasabov, 2001). Here, the first-order Takagi-Sugeno type fuzzy rules are employed and the linear functions in the consequences are created using the *weighted linear least-square estimator* denoted as WLS (see Appendix A.2) and updated by the *weighted recursive linear least-square estimator* denoted as WRLS (see Appendix A.4) with learning data. Each of the linear functions can be expressed as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_q x_q. \tag{2.25}$$

The creation of the first m fuzzy rules in DENFIS is described as follow:

- Step 1: take the first n_0 learning data samples from the learning data set;

- Step 2: implement clustering using ECM to these n_0 data to obtain m cluster centres;
- Step 3: for every cluster centre C_i, find p_i data samples from the learning data set whose positions in the input space are closest to the centre, $i = 1, 2, ..., m$;
- Step 4: to obtain a fuzzy rule corresponding to a cluster centre, create the antecedents of the fuzzy rule using the position of the cluster centre and Equation 2.23. Using the weighted linear least-square estimator (see Appendix A.2) on p_i data samples calculate the coefficients of the consequent function. The distances between p_i data samples and the cluster centre are taken as the weights.

In the above steps, m, n_0 and p are the parameters of DENFIS learning model, and the value of p_i should be greater than the number of input variables, q.

As new data samples are presented to the system, new fuzzy rules may be created and some existing rules are updated. A new fuzzy rule is created if a new cluster centre is found by the ECM. The antecedent of the new fuzzy rule is formed using Equation 2.23 with the position of the cluster centre (as a rule node). However, to construct the consequence function of this new fuzzy rule a minimum number of samples that belong to this new cluster is required. The size of minimum number of samples itself relates to dimensionality of the data set. Yet, as this new cluster will only have one single sample, it is then problematical to construct the consequence function. To overcome this issue, an existing fuzzy rule about which rule node is the closest to this new rule node is then found; the consequence function of this rule is then taken as the consequence function for this new fuzzy rule. Nevertheless, as more data samples are presented, and more samples might join this new cluster, both the antecedent and consequence parts of the fuzzy rule related to this new cluster will then again be updated.

For every data sample, several existing fuzzy rules are updated using WRLS (see Appendix A.4) if their rule nodes have distances to the data point in the input space that are not greater than $2 \times Dthr$ (the threshold value, a clustering parameter). The distances between these rule nodes and the data sample in the input space are taken as the weights. In addition to this, one of these rules may also be updated through changing its antecedent so that, if its rule node position (cluster centre) is changed by the ECM, the fuzzy rule will have a new antecedent.

2.5.2.2 Takagi-Sugeno Fuzzy Inference in DENFIS

DENFIS offers the ability to extract a number of rules that can be easily understood and used to help predicting the upcoming values of a time-series. As it was outlined in Section 2.5.2, DENFIS employs the Takagi-Sugeno fuzzy inference system. However, the Takagi-Sugeno fuzzy inference system utilised in DENFIS is applied as a dynamic inference system, in which existing rules could be updated and new rules might be created as new observations are added to the system.

Additionally, DENFIS dynamically creates specific fuzzy inference system for each input vector. In DENFIS when a new prediction needs to be made for an input vector then instead of using all existing rules to construct a Takagi-Sugeno inference system only m most relevant rules will be activated as used to create the inference system. The rules are chosen based on the position of the input vector. Since in DENFIS the rules are updated continuously, two input vectors with the same values at different time points may have different inferences as the fuzzy rules may have been updated before the second input vector entered the system.

An example of a set of 3 activated rules chosen to make a prediction for an input vector $\mathbf{x}$ when DENFIS is applied to the Mackey-Glass data set is

Rule1
If x(t-6) is GaussianMF (0.50 0.59)
x(t) is GaussianMF (0.52 0.73)
Then x(t+6) = 1.40 - 0.06*x(t-6) + 0.56*x(t)

Rule2
If x(t-6) is GaussianMF (0.49 0.73)
x(t) is GaussianMF (0.49 0.77)
Then x(t+6) = 1.36 - 0.96*x(t-6) + 1.39*x(t)

Rule3
If x(t-6) is GaussianMF (0.49 0.78)
x(t) is GaussianMF (0.50 0.59)
Then x(t+6) = 1.22 - 0.92*x(t-6) + 1.53*x(t)

Figure 2.9. 3 activated fuzzy rules created and chosen by DENFIS to construct a Takagi-Sugeno inference system when being applied for prediction of the Mackey-Glass data set. Rules are extracted from DENFIS available in the NeuCom (`http://www.theneucom.com`).

presented in Figure 2.9.

2.5.2.3 Evolving Clustering Method

The Evolving Clustering Method denoted as ECM is an evolving, on-line, maximum distance-based clustering method developed by Q. Song and Kasabov (2001) to implement a scatter partitioning of the input space for the purpose of creating fuzzy inference rules.

There are two possible modes of this method: the first one is usually applied to on-line learning systems i.e. DENFIS, and the second one is more suitable for off-line learning systems. The off-line mode of ECM with constrained minimisation (ECMc) is an extension of the on-line mode. It takes the result from the on-line mode as initial values, and afterwards an optimisation is applied that makes a pre-defined objective function based on a distance measure to reach a minimum value subject to given constraints.

On-Line Evolving Clustering Method: ECM

Without any optimisation, the on-line ECM (hereafter is denoted as ECM) is a fast, one-pass algorithm for a dynamic estimation of the number of clusters in a set of data samples, and for finding their current centres in the input data space. It is a distance-based clustering method where the cluster centres are represented by evolved nodes in an on-line mode. In any cluster, the maximum distance, $MaxDist$, between a data sample and the cluster centre, is less than a threshold value, $Dthr$, that has been set as a clustering parameter. This parameter would affect the number of clusters to be created.

In the clustering process, the data samples come from a data stream and this process starts with an empty set of clusters. When a new cluster is created, its cluster centre, Cc, is located and its cluster radius, Ru, is initially set with a value 0. As new samples are presented one after another, new clusters may be created or some already created clusters will be updated through changing their centres' positions and increasing their cluster radii. Which cluster should be updated and how it should be changed depends on the position of the current data sample. A cluster will not be updated any more when its cluster radius, Ru, has reached the special value that is, usually, equal to the threshold value $Dthr$.

Figure 2.10 illustrates a brief ECM clustering process in a 2-D space and details of the ECM algorithm are given below:

- Step 1: create the first cluster C_1 by simply using the first sample from the input data stream and taking its position as the first cluster centre Cc_1, and initially setting the cluster radius Ru_1 to a value 0 (Figure 2.10a);

- Step 2: if all samples from the data stream have been presented, the clustering process finishes. Else, the current input sample x_i, is taken

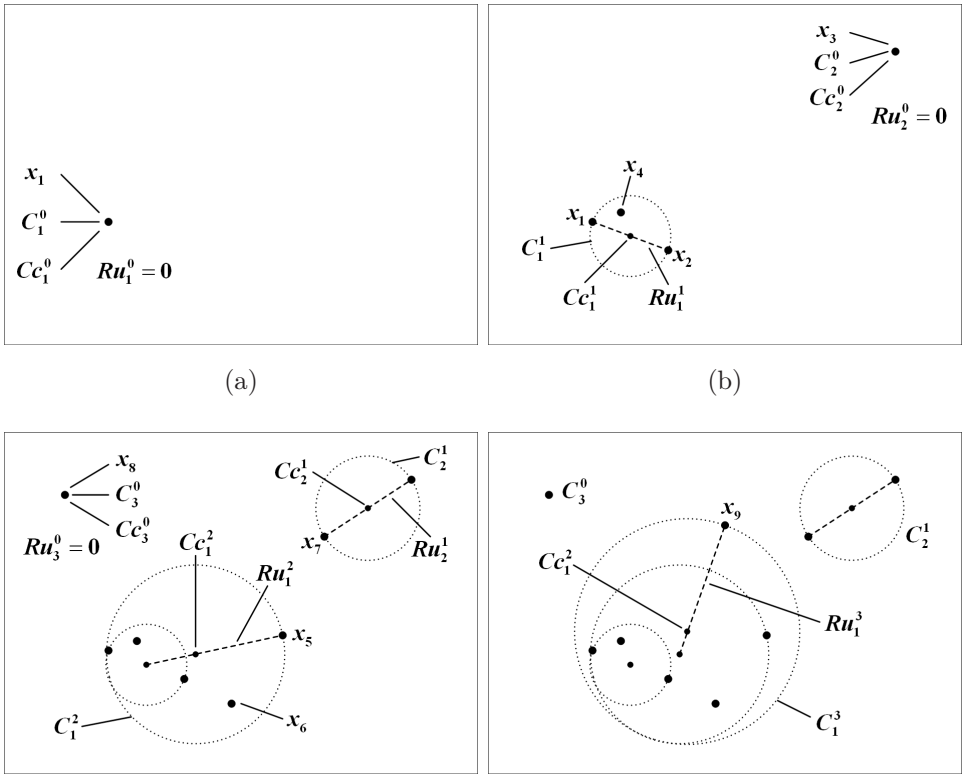


Figure 2.10. Illustration of ECM clustering process in a 2-D space (Q. Song & Kasabov, 2001).

and the normalised Euclidean distances $d(x_i, x_j)$, between this sample point and all n already created cluster centres Cc_j,

$$d(x_i, Cc_j) = \|x_i - Cc_j\|, \ j = 1, 2, ..., n \tag{2.26}$$

are calculated. A normalised Euclidean distance between two q-element vectors x and y is defined as follows:

$$\|x - y\| = \left(\frac{1}{q} \sum_{i=1}^{q} |x_i - y_i|^2 \right)^{1/2} \tag{2.27}$$

where $x, y \in R^q$;

- Step 3: if there is a cluster C_m with its centre Cc_m; cluster radius Ru_m; and a distance value $d(x_i, Cc_m)$, which is between the cluster centre Cc_m

and the sample x_i and defined as follows:

$$\begin{aligned} d(x_i, Cc_m) &= \min_j d(x_i, Cc_j) \\ &= \min_j(\|x_i - Cc_j\|),\ j = 1, 2, ..., n \end{aligned} \tag{2.28}$$

and, $d(x_i, Cc_m) \leq Ru_m$, it is regarded that the current sample x_i belongs to the cluster C_m. In this case neither a new cluster is created, nor is any existing cluster updated (e.g. data vectors x_4 and x_6 in Figure 2.10b). The algorithm then returns to Step 2;

- Step 4: else not the case of Step 3, then find a cluster C_a, with its centre Cc_a; cluster radius Ru_a and the distance value $d(x_i, C_a)$ from all n existing clusters through calculating the extended distance values:

$$s(x_i, Cc_j) = d(x_i, Cc_j) + Ru_j,\ j = 1, 2, ..., n, \tag{2.29}$$

and then selecting the cluster C_a with the minimum value $s(x_i, Cc_a)$:

$$\begin{aligned} s(x_i, Cc_a) &= d(x_i, Cc_a) + Ru_a \\ &= \min_j s(x_i, Cc_j),\ j = 1, 2, ..., n. \end{aligned} \tag{2.30}$$

- Step 5: if $s(x_i, Cc_a) > 2 \times Dthr$, the sample x_i does not belong to any existing clusters. A new cluster is then created in the same way as described in Step 1 (e.g. input data vectors x_3 and x_8 in Figure 2.10c). The algorithm then returns to Step 2;

- Step 6: if $s(x_i, Cc_a) \leq 2 \times Dthr$, the cluster C_a is updated by moving its centre, Cc_a, and increasing its radius value, Ru_a. The updated radius Ru_anew is set to be equal to $s(x_i, Cc_a)/2$ and the new centre Cc_anew is located on the line connecting the sample x_i to the old cluster centre Cc_a, so that the distance from the new centre Cc_anew to the sample x_i is equal to Ru_anew (e.g. input data points x_2, x_5, x_7 and x_9 in Figure 2.10d). The algorithm then returns to Step 2.

Through this clustering process, the maximum distance from any cluster centre to the farthest sample that belongs to this cluster is kept below the threshold value $Dthr$, although the algorithm does not keep any information of passed samples.

Constrained Optimisation and Off-Line Evolving Clustering: ECMc

Ideally a cluster centre should be positioned at the centre of the gravity among all samples that belong to the cluster, as this is what the term "cluster centre" means. However, in ECM the cluster centre does not necessarily coincide with the centre of gravity of a cluster. To address this issue, an off-line version of ECM was proposed (Q. Song & Kasabov, 2001).

The off-line Evolving Clustering Method, called ECMc applies an optimisation procedure to the resulted cluster centres after the application of ECM in a way that it moves the resulted cluster centres to the centre of gravity.

The ECMc partitions a data set including p vector $x_i,\ i = 1, 2, ..., p$, into n clusters $C_j,\ j = 1, 2, ..., n$ (using the ECM algorithm), and finds a cluster centre in each cluster in order to minimise an objective function based on a distance measure subject to given constraints. Taking the normalised Euclidean distance (Equation 2.27) between the sample vector x_k, belonging to a cluster C_j, and the corresponding cluster centre Cc_j, as the measure, the objection function is defined as follows:

$$\begin{aligned} J &= \sum_{j=1}^{n} J_j \\ &= \sum_{j=1}^{n} \sum_{x_k \in C_j} \|x_k - Cc_j\| \end{aligned} \tag{2.31}$$

where $J_j = \sum_{x_k \in C_j} \|x_k - Cc_j\|$ is the sub-objection function within cluster $C_j,\ j = 1, 2, ..., n$, and the constraints are defined as follows:

$$\|x_k - Cc_j\| \leq Dthr, \tag{2.32}$$

where $x_k \in C_j$ and $j = 1, 2, ..., n$.

The clusters are typically defined as a $p \times n$ binary membership matrix U, where the element u_{ij} is set to 1 if the i^{th} data point x_i belongs to the j^{th} cluster C_j and to 0 if otherwise. Once the cluster centres Cc_j are defined, the values u_{ij} are derived as follows:

$$u_{ij} = \begin{cases} 1, & \text{if } \|x_i - Cc_j\| \leq \|x_i - Cc_k\|, \text{for each } j \neq k \\ 0, & \text{if } \|x_i - Cc_j\| > \|x_i - Cc_k\|, \text{for each } j \neq k \end{cases} \tag{2.33}$$

The ECMc minimising algorithm works in an off-line, iterative mode on a batch of data repeating the following steps:

- Step 1: initialise the cluster centres $Cc_j,\ j = 1, 2, ..., n$, which are produced by the ECM using p vector $x_i,\ i = 1, 2, ..., p$;
- Step 2: determine the membership matrix U using Equation 2.33;
- Step 3: employ the constrained minimisation method with Equations 2.31 and 2.32 to obtain new cluster centres;
- Step 4: calculate the objective function J according to Equation 2.31. Stop if the result is below a certain tolerance value, or if its improvement over previous iteration is below a certain threshold, or if the iteration number of optimising operation is greater than a certain value. Else, the algorithm returns to Step 2.

2.5.3 Evolving Neuro-Fuzzy System: EFS

In Angelov (2002) an evolving neuro-fuzzy system framework, namely the *evolving fuzzy systems* and denoted as EFS, is introduced. EFS is defined as a self-developing, self-learning neuro-fuzzy system where both parameters and structure are self-adapting on-line. The framework can also be represented as a connectionist architecture illustrated in Figure 2.11 (Kasabov, 2007a).

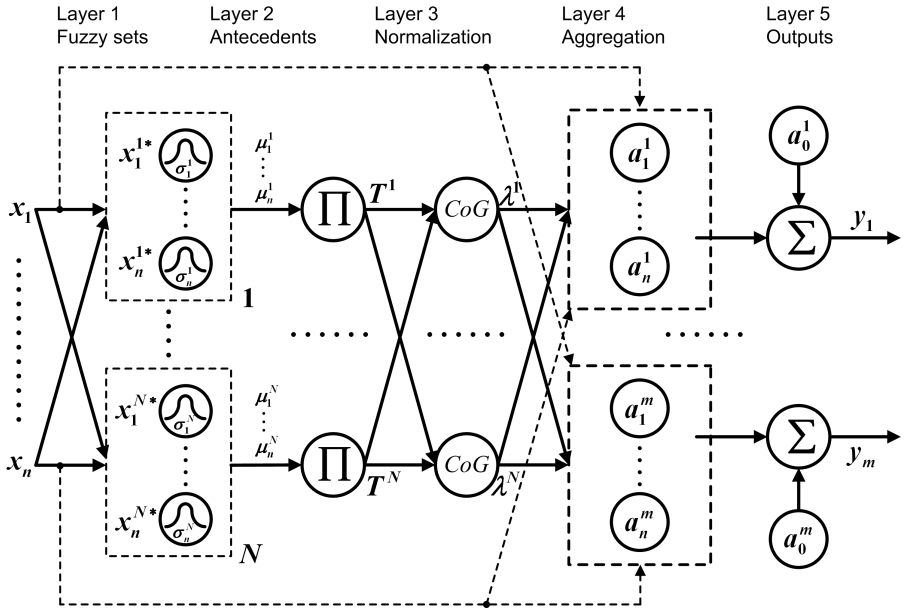


Figure 2.11. Neuro-fuzzy interpretation of the evolving fuzzy system. The structure is not predefined and fixed; rather it evolves 'from scratch' by learning from data simultaneously with the parameter adjustment/adaption (Angelov, 2006).

The EFS framework consists of five layers. The first (input) layer serves to pass the input signal to the respective neurons of each rule corresponding to the membership functions of fuzzy sets. The second layer represents the antecedent parts of the fuzzy rules. Inputs to each neuron are equal to the degree of membership to the respective fuzzy set of every input signal. This layer produces as output the firing level of the i-th rule. The third layer of the network takes as inputs the firing levels of the respective rule and gives as output the normalised firing level. The fourth layer aggregates the antecedent and consequent parts of the rules that represent the local subsystems. Finally, the last, fifth layer forms the total output of the evolving fuzzy system performing a weighted summation of local sub-models (Angelov, 2002; Angelov & Filev, 2004c; Kasabov, 2007a).

EFS with Takagi-Sugeno inference system is called the *evolving Takagi-Sugeno* model, denoted as eTS (Angelov & Filev, 2004b). The eTS applies

a learning algorithm that combines unsupervised learning with respect to the antecedent part of the model and supervised learning in terms of the consequent parameters. This concept of learning is similar to the DENFIS training algorithm presented in Section 2.5.2 (Kasabov & Song, 2002).

The eTS employs an unsupervised clustering algorithm which continuously analyses the input-output data streams and identifies emerging new data structures (Angelov & Buswell, 2002; Angelov & Filev, 2004b, 2004c). The algorithm clusters the input-output space into N fuzzy regions which represent the fuzzy rules. The cluster centres are then projected as the antecedent part of each fuzzy region; the algorithm also assigns a linear subsystem to each of the clusters. The learning rules of the eTS to define the antecedent part of the fuzzy rules apply an evolving on-line clustering approach the eClustering (Angelov & Filev, 2004a).

The on-line eTS model implements an on-line clustering procedure (Angelov & Buswell, 2002; Angelov & Filev, 2004b, 2004c) which starts with the first data point established as the centre of the first cluster. This cluster centre is then used to form the antecedent part of the first fuzzy rule. Consequently, the potential of this first data point being the first cluster centre is assumed equal to 1.

As new data becomes available the potential of each new data point is calculated recursively by:

$$P_k(z_k) = \frac{k-1}{(k-1)(\vartheta_k + 1) + \sigma_k - 2\upsilon_k} \tag{2.34}$$

where

$$\begin{aligned} \vartheta_k &= \sum_{j=1}^{n+1} \left(z_k^j\right)^2, \\ \sigma_k &= \sum_{l=1}^{k-1} \sum_{j=1}^{n+1} \left(z_l^j\right)^2, \\ \upsilon_k &= \sum_{j=1}^{n+1} z_k^j \beta_k^j; \ \beta_k^j = \sum_{l=1}^{k-1} z_l^j. \end{aligned}$$

When new data are collected in on-line mode, they influence the potentials of the existing cluster centres that are respective to the focal points of existing rules since by definition the potentials depend on the distance to all data points, including the new ones. In relation to this, the potential of existing cluster centres needs to be updated as well. The recursive equation to update the potential of existing cluster centres is given by:

$$P_k(z_l^*) = \frac{(k-1)P_{k-1}(z_l^*)}{k-2+P_{k-1}(z_l^*)+P_{k-1}(z_l^*)\sum_{j=1}^{n+1}\left(d_{k(k-1)}^j\right)^2} \tag{2.35}$$

where $P_k(z_l^*)$ is the potential at time k of the cluster centre, which is a prototype of the l^{th} rule; $d_{k(k-1)}^j = z_k^j - z_{k-1}^j$ denotes projection of the distance between two data points (z_k^j and z_{k-1}^j) on the axis z^j.

Afterwards, potentials of the new data points are compared to the updated potential of existing cluster centres. If the potential of the new data point is higher than the potential of the existing centres then the new data point is accepted as a new centre, that is a new cluster is formed and a new rule with a focal point based on the projection of the new cluster centre is created. However, if in addition to a prior condition the new data point is within a close range to an existing cluster centre defined as follows:

$$\frac{P_k(Z_k)}{\max_{l=1}^{R} P_k(z_l^*)} - \frac{\delta_{\min}}{r} \geq 1 \tag{2.36}$$

then the new data point is set as the new cluster centre replacing the old centre.

In the on-line eTS model estimation of the parameters of the consequent linear model of each rule is recursively calculated through the modified recursive least square algorithm or weighted recursive least square algorithm as introduced by Angelov and Filev (2004b). Generally, the recursive procedure for on-line learning of the eTS model is defined as follows (Angelov & Filev, 2004b):

- Step 1: initialisation of the rule-base structure (antecedent part of rules);
- Step 2: at the next time step reading the next data sample;
- Step 3: recursive calculation of the potential of each new data sample to influence the structure of the rule-base;
- Step 4: recursive update of the potentials of old centres taking into account the influence of the new data sample;
- Step 5: possible modification or upgrade of the rule-base structure based on the potential of the new data sample in comparison to the potential of the existing rules' centres (focal points);
- Step 6: recursive calculation of the consequent parameters;
- Step 7: prediction of the output for the next time step by the eTS model.

In (Angelov & Filev, 2004b) the algorithm was tested and applied for the prediction of both the Mackey-Glass data set (a benchmark data set) and data from a fan-coil sub-system of an air-conditioning system serving a real building. Results of conducted experiments in (Angelov & Filev, 2004b) suggest that eTS is able to develop/evolve an existing model when the data pattern changes and is a reliable method for time-series analysis and prediction.

2.5.4 Instance-Based Learning

In this section some commonly used algorithms of instance-based learning, namely the k-Nearest Neighbour (kNN), the Weighted k-Nearest Neighbour (WKNN) and the *locally weighted regressions* are outlined and explained. The transductive model of multiple time-series analysis proposed in this thesis implements and extends further the general principles of the methods explained here.

2.5.4.1 k-Nearest Neighbour: kNN

The most basic instance-based method is the kNN algorithm (Govindarajan & Chandrasekaran, 2010; Soucy & Mineau, 2001). This algorithm assumes all instances (data samples) correspond to points in the q-dimensional space $\Re^q$. The nearest neighbours of an input vector are defined in terms of the standard Euclidean distance (Mitchell, 1997). More precisely, let an arbitrary input vector x be described by the feature vector:

$$\langle a_1(x), a_2(x), ..., a_q(x) \rangle$$

where $a_r(x)$ denotes the value of r^{th} attribute of input vector x. Then the distance between two instances x_i and x_j is defined to be $d(x_i, x_j)$, where

$$d(x_i, x_j) = \left(\sum_{r=1}^{q} \left(a_r(x_i) - a_r(x_j) \right)^2 \right)^{1/2} \tag{2.37}$$

In nearest neighbour learning the target function may be either discrete-valued or real-valued. Let the target function be a discrete-valued of the form $f : \Re^q \rightarrow V$, where V is the finite set $\{v_1, v_2, ..., v_s\}$. Then, the kNN equation for approximating a discrete-valued target function is given by:

$$\hat{f}(x_i) = \arg\max_{v \in V} \sum_{j=1}^{k} \delta(v, f(x_j)) \tag{2.38}$$

where

$$\delta(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

and x_i is a new input vector to be classified; $x_1, x_2, ..., x_k$ denote k instances from training samples that are nearest to x_i.

As it can be seen, the value $\hat{f}(x_i)$ given by Equation 2.38 as its estimate of $f(x_i)$ is just the most common value of f among the k nearest neighbours of x_i. If $k = 1$, then the 1-Nearest Neighbour algorithm assigns to $\hat{f}(x_i)$ the value of $f(x_j)$ where x_j is a single training sample nearest to x_i. For larger values of k,

the algorithm assigns the most common value between the k nearest training samples.

The kNN algorithm is easily adapted to approximating continuous-valued target functions. To accomplish this, the algorithm calculates the mean value of the k nearest neighbour rather than calculate their most common value. More precisely, to estimate a real-valued target function $f : \Re^q \rightarrow \Re$, $\hat{f}(x_i)$ is now defined by:

$$\hat{f}(x_i) = \frac{1}{k} \sum_{j=1}^{k} f(x_j) \tag{2.39}$$

2.5.4.2 Weighted kNN: WKNN

One obvious refinement to the kNN algorithm is to weigh the contribution of each k nearest neighbours in relation to their distance to the input vector x_i, assigning greater weight to the closer ones (Mitchell, 1997). This approach is known as the WKNN algorithm.

To approximate discrete-valued target functions, the contribution of each neighbour is weighted according to the inverse square of its distance from x_i. This can be accomplished by modifying Equation 2.38 to the following:

$$\hat{f}(x_i) = \arg\max_{v \in V} \sum_{j=1}^{k} w_j \delta(v, f(x_j)) \tag{2.40}$$

while to approximate real-valued target functions, $\hat{f}(x_i)$ is calculated as follows:

$$\hat{f}(x_i) = \frac{\sum_{j=1}^{k} w_j f(x_j)}{\sum_{j=1}^{k} w_j} \tag{2.41}$$

where

$$w_j = \frac{\max\left[\max(d) - (d_j - \min(d))\right]}{\max(d)}.$$

The vector $d = [d_1, d_2, ..., d_k]$ is defined as the distances between the input vector x_i and the k nearest neighbors, $d_j = d(x_i, x_j)$ for $j = 1$ to k,

as in Equation 2.37, where q is the number of the input variables. The parameters $\max(d)$ and $\min(d)$ are the maximum and minimum values in d respectively. The weights w_j have the values between $\min(d)/\max(d)$ and 1; the closest sample to x_i will then have the weight value of 1, and it has the value $\min(d)/\max(d)$ in case of maximum distance.

Both kNN and WKNN (Dudani, 1976; Tan, 2005) consider only the k nearest neighbours to classify the input vector. However, once distance weighting is added, there is actually no harm in allowing all training samples to have an influence on the classification of x_i, since very distant samples will have very little effect on $\hat{f}(x_i)$. The only disadvantage of considering all samples is that the classifier will run more slowly.

Considering all training samples when classifying a new query instance is called a global method, while if only the nearest training samples are considered then it is called a local method. This suggests that the instance-based learning which is a realisation of the transductive inference falls under the category of local modelling approach.

2.5.4.3 Locally Weighted Regressions

The nearest neighbour approaches described in Sections 2.5.4.1 and 2.5.4.2 can be seen as approximating the target function $f(\mathbf{x})$ at the single input vector $\mathbf{x} = \mathbf{x}_i$. Locally weighted regression is a generalisation of this approach. It constructs an explicit approximation to f over a local regression surrounding $\mathbf{x}_i$. Locally weighted regression uses nearby or distance-weighted nearest neighbour to form this local approximation to f (Atkeson, Moore, & Schaal, 1997; Hwang, 2009; Mitchell, 1997).

The phrase "locally weighted regression" is called *local* because the function is approximated based only on data near the query point, *weighted* because the contribution of each training example is weighted by its distance from the

query point, and *regression* because this is the term used widely in statistical learning community for the problem of approximating real-value functions (Mitchell, 1997).

Given a new input vector $\mathbf{x}_i$, the general approach in locally weighted regression is to construct an approximation $\hat{f}$ that fits the training examples in the neighbourhood surrounding $\mathbf{x}_i$ (that is a number of nearest neighbours of $\mathbf{x}_i$) by giving greater weight to closer neighbours (as described in Equation 2.40). This approximation is then used to calculate the values $\hat{f}(\mathbf{x}_i)$, which is output of the estimated target value for the query instance. In locally weighted regression, the target function f is approximated near $\mathbf{x}_i$ using a linear function of the form:

$$\hat{f}(\mathbf{x}_i) = w_0 + w_1 x_i^1 + w_2 x_i^2 + ... + w_q x_i^q \tag{2.42}$$

where x_i^r denotes the value of the r^{th} attribute of instance $\mathbf{x}_i$.

To find the coefficients $w_0, w_1, w_2, ..., w_q$ various methods can be used to minimise the error in fitting such linear functions to a given set of training sample, i.e. ordinary least-square estimator (OLS), WLS (see Appendix A) or gradient descent (see Appendix B).

When being applied to solve the problem of time-series prediction, the explanatory variables of the linear function in 2.42 are current or past observations of the time-series being modelled or predicted, i.e. $x_t, x_{t-1}, \ldots, x_{t-k}$, whilst $\hat{f}(\mathbf{x}_i)$ is the prediction of that particular time-series at a certain time-point in the future, i.e. x_{t+n}, calculated based on the explanatory variables.

2.6 Methods of Reasoning

As different models of time-series analysis, outlined in previous sections, implement different methods of learning or *reasoning*, this section of the chapter gives a brief description of two different reasoning methods used in the

learning process of time-series analysis, i.e. the *inductive* reasoning and the *transductive* reasoning.

2.6.1 Inductive Reasoning

Induction or inductive reasoning, sometimes called inductive logic or inductive learning, is the process of reasoning in which the premises of an argument are believed to support the conclusion but do not entail it. Induction is a form of reasoning that makes generalisations based on individual instances. It is used to either describe properties or relations to types based on a number of observations, samples or experiences or to formulate laws based on limited observations of recurring phenomenal patterns. This method is concerned with the creation of a model (a function) from all available data representing the entire problem space, e.g. a regression formula, a neural network of MLP, SVM, etc. Afterwards, the model is applied to solve problems on a newly collected data set.

2.6.2 Transductive Reasoning

Transductive reasoning or transductive inference, introduced in Vapnik (1998), is defined as a method used to estimate the value of a potential model (function) only for a single point of space (that is, a new data vector) by utilising additional information related to that vector. While the inductive approach is useful when a global model of the problem is needed in an approximate form, the transductive approach is more appropriate for applications where the focus is not on the model, but rather on every individual case. This relates to the common sense principle which states that to solve a given problem one should avoid solving a more general problem as an intermediate step (Bosnic, Kononenko, Robnik-Sikonja, & Kukar, 2003).

2.7 Conclusion

This chapter outlines methods of time-series analysis and modelling from the statistics and machine learning domain. These methods are related to the work of developing the integrated multi-model framework for multiple time-series analysis and modelling conducted in this PhD study. The researches and developments carried out in this study are based on methods reviewed in this chapter.

Chapter 3

Dynamic Interaction Network for Multiple Time-Series Analysis and Modelling

3.1 Introduction

This chapter presents a methodology named the *Dynamic Interaction Network* (DIN) which utilises first-order differential equations and the Kalman filter (Komogortsev & Khan, 2008; S. Lee, Lim, Baek, & Sung, 1999; Tsai & Kurz, 1983; Welch & Bishop, 1995; Whittle & Schumann, 2004) in combination with the EM algorithm (Bi, 2009; Jordan & Jacobs, 1994; Lawrence & Reilly, 1990; R. Li, Bhanu, & Dong, 2008) to dynamically extract and model the pattern of interactions between variables from a set of multiple time-series data.

Standard approaches to multivariate data analysis model the relationship between multiple observed variables in the structure of a single dependent variable being influenced by several independent variables as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k, \tag{3.1}$$

where y is the dependent variable; $x_1, x_2, .., x_k$ are the independent variables; $\beta_1, \beta_2, ..., \beta_k$ are the coefficients of influence; and β_0 is a constant value. In contrast, DIN extracts and models interactions between variables through a complete, fully-connected network. Additionally, as the process of extraction

is performed dynamically, knowledge about interactions between variables is updated on a regular basis as the behaviour of the system changes dynamically over time.

The DIN model explained in this chapter aims to address the following research questions relating to the problem of modelling dynamic pattern of interaction between multiple variables and multiple time-series prediction: (1) Can a global pattern of interactions between multiple variables be dynamically modelled in a structure that is easily understood? (2) Would dynamically capturing a global pattern of interactions be of help to reveal new knowledge about complex interdependencies between multiple variables? (3) Does incorporating adaptive knowledge about a global pattern of interactions help to achieve better results in multiple time-series prediction?

3.2 Fundamentals of Multiple Variables Interactions Modelling

When studying a physical system, for instance a moving aircraft, a chemical process, or the national economy, an engineer or a researcher would try to develop a mathematical model that adequately represents some aspects of the behaviour of the observed system. In particular, researchers have been trying to model the interrelationships among variables of interest, inputs to the system and outputs from the system through physical insights, fundamental "laws", and empirical experiments. It is then expected that with such mathematical models and the tools provided by system and control theories, one should be able to investigate and make assumptions about the underlying structure of the system (Widiputra, Pears, & Kasabov, 2011b).

Based on this concept, given a set of measurements of multiple variables sampled on a regular or irregular timely basis, a mathematical model that

explains the relationships, interdependencies or interactions between the variables could be put in place. Before getting into the details of how such a model can extract pattern of interactions from multiple time-series, this section of the chapter will revisit and outline the fundamental theories and concepts that underpin the process of extraction of pattern of interactions between multiple variables in a dynamic environment.

3.2.1 A Global Model as the Method of Reasoning

A global model is a realisation of inductive reasoning that builds a single model by learning from the entire data set or problem space. The developed model is then applied to new data that arrives in the future (Baruch & Stoyanov, 1995; Y. Lee et al., 1997; Marin, Garcia-Lagos, Joya, & Sandoval, 2002). Global modelling is the most commonly-used approach for inductive reasoning (Christou & Papageorgiou, 2007; Hinojosa & Hoese, 2010; Kasabov, 2007b; Tomic, 1995). A global model is a distinct, fixed, reusable model, and it is very useful to describe the general underlying behaviour of a stochastic system.

Therefore, most of the research carried out in the field of time-series analysis and modelling has been based on the concept of global modelling. For instance, linear regression models (Hastie, Tibshirani, & Friedman, 2003), the MLP (Hagan, Demuth, & Beale, 1995; Hornik et al., 1989; S. Yang, Ho, & Lee, 2006), the RBFN (Lucks & Oki, 1999; Marinaro & Scarpetta, 2000), SVM (Boser, Guyon, & Vapnik, 1992; H. Kim, Pang, Je, Kim, & Bang, 2002; Vapnik, 1998), the Adaptive-Network-Based Fuzzy Inference System (ANFIS) (J. Jang, Sun, & Mizutani, 1997; J.-S. Jang, 1993; J.-S. Jang & Sun, 1995) and the Echo State Network (ESN) (Cernansky & Makula, 2005; X. Lin, Yang, & Song, 2009) are examples of global models. The MLP and SVM machine learning algorithms were proposed many years ago and are still the two most widely used neural network models in the domain of time-series analysis.

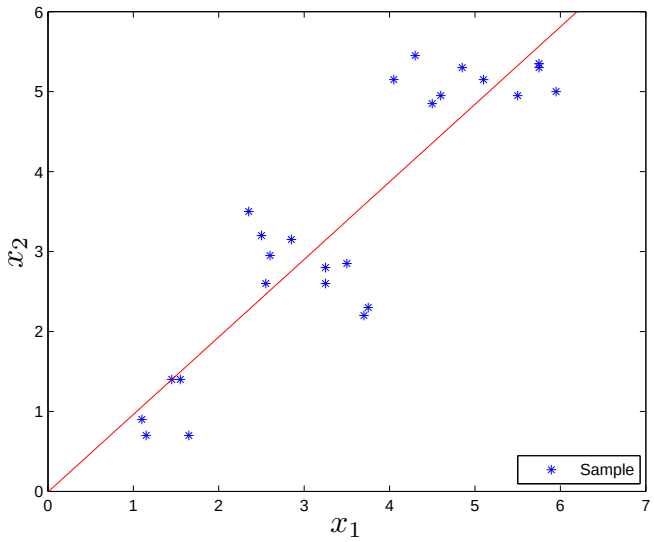


Figure 3.1. Illustration of global modelling to estimate a linear regression function from a sample data set (Hwang, 2009).

Based on the characteristics of a global model, it is then logical to use such an approach to gain knowledge about patterns of interactions from a set of measurements from multiple variables. The idea is simply to use the set of measurements as learning samples, from which a mathematical model that explains interdependencies or interactions between variables of interest is then derived. Therefore, DIN utilises this method of reasoning as its main modelling method in the learning process.

However, there are also a few limitations to the global model (Hwang, 2009):

1. First, as the model is based on all available data with the objective of minimising overall prediction error, it will be biased toward the majority of the data. A pattern without enough support will have little influence on the model. This is similar to the issue with interpolation versus extrapolation. If new data exhibits a pattern similar to some existing pattern, then conceptually a process of interpolation can be used for pattern extraction, where for this pattern enough support is provided

by the prediction. However, if the new pattern is very different from any of the existing patterns, then extrapolation needs to be performed, which carries a higher degree of uncertainty than interpolation.

2. Secondly, capturing global trends might not be sufficient enough to model the dynamics of highly volatile time-series which might lead to poor prediction of its future states. In fact, in an environment where movements of time-series is volatile, different models at different time moments might be required.

To deal with such problems, an incremental/online learning system that adapts to new data and traces its evolution needs to be applied.

3.2.2 Discrete-Time Approximation of First-Order Differential Equations

The state of interdependencies or interactions between multiple variables of a dynamic system can be modelled as a discrete-time approximation of first-order differential equations, given by:

$$x_{t+1} = Fx_t + \varepsilon_t, \tag{3.2}$$

where $x_t = (x_1, x_2, ..., x_k)$ is the observed measurements at the t-th time interval and k is the number of input variables being modelled, ε_t is a noise component with covariance $E = cov(\varepsilon_t)$, and $F = (f_{i,j})$; $i = 1$ to k, $j = 1$ to k is the transition matrix relating conditions at x_t to x_{t+1}. This equation is related to the continuous first-order differential equation $dx/dt = \Psi x + e$ by $F = \tau\Psi + I$ and $\varepsilon_t = \tau e$ where τ is the time interval. However, for the ease of modelling and the need of irregular time-course data processing, discrete approximation is more likely to be used rather than the continuous model (Aoki & Shell, 1989; Widiputra et al., 2011b).

Besides the fact that a discrete time-approximation of first-order differential equations is a tool widely used for modelling biological processes (Chan, Kasabov, & Collins, 2006; Jones, Plank, & Sleeman, 2009; Z. Wang et al., 2008)), there are two advantages in using the first-order differential equations. First, relations of multiple variables or multiple time-series can be elucidated from the transition matrix F through choosing a threshold value $(\zeta; 1 > \zeta > 0)$. If $|f_{i,j}|$ is larger than the threshold value ζ, $x_{j,t}$ is considered to have a significant influence on $x_{i,t+1}$. A positive value of $f_{i,j}$ indicates a positive influence and vice-versa. Second, they can be easily manipulated with a filter, e.g. the Kalman filter, to handle irregularly sampled data, which will finally allow parameter estimation, likelihood evaluation, model simulation and prediction.

Yet, there is a main drawback in using differential equations. It requires the estimation of k^2 parameters for the transition matrix F and $k(k-1)/2$ parameters for the noise covariance E. To minimise the number of model parameters, only F is being estimated and E is set to a small value. For instance, when observing two series, each consisting of only 4 samples, over-parameterisation may be avoided by setting the value of k to 4, as this is the maximum number of samples available before the number of parameters exceeds the amount of training data. Hence the number of model parameters, the size of F is k^2, is limited to the number of training data, $k \times 4$ samples.

To cope with irregularly sampled data, the state-space methodology is put in place. The actual trajectories are treated as a set of unobserved or hidden variables called the *state variable*; a filter can then be applied to compute their optimal estimates based on the observed measurements. The state variables that are regular or complete can now be applied to perform parameter estimations, in particular the F matrix, instead of the observed measurement that are irregular or incomplete.

This approach is better than the interpolation methods as it prevents false

modelling by trusting a fixed set of interpolated points that may be erroneous. Nevertheless, the use of the state-space methodology is also suitable for estimating the F matrix when dealing with regularly sampled data.

3.2.3 State-Space Representation

In Section 3.2.2 the term *state variable* was introduced. The state variables can be considered as the smallest possible subset of the system variables that can represent the entire state of the system at any given point in time. The phrase *system state transition function* would describe the transfer function representing the interactions between observed variables occurring at any given time moment that maps present values of the state variables to their future values.

The minimum number of state variables required to represent a given system, k, is usually equal to the order of the system's defining differential equation. If the system is represented in transfer function form, the minimum number of state variables is equal to the order of the transfer function's denominator after it has been reduced to a proper fraction. It is important to understand that converting a state-space realisation to a transfer function form may lose some internal information about the system, and may provide a description of a system which is stable, while the state-space realisation is unstable at certain points.

A mathematical model of a system as a set of input, output and state variables related by the first-order differential equations is called a *state-space representation* (Arslanalp & Tola, 2006; Simaan, Ferreira, Chen, Antaki, & Galati, 2009). State-space models are a flexible family of models which fits the modelling of many scenarios. The strongest feature of state-space models is the existence of very general algorithms for filtering, smoothing and prediction (Bay, 1998).

Multiple time-series at a particular time moment are expressed as vectors and the differential equations are written in a form of a matrix which is most suitable to be used to map the number of inputs, outputs and state variables when the dynamical system is linear and time invariant. The state-space representation (also known as the "time-domain approach") provides a convenient and compact way to model and analyse systems with multiple inputs and outputs, suitable to extract patterns of interactions in multiple time-series data. With p inputs and q outputs, one would otherwise have to write $p \times q$ Laplace transforms to encode all the information about a system. Unlike the frequency domain approach, the use of the state-space representation is not limited to systems with linear components and zero initial conditions. State-space refers to a space whose axes are the state variables, and the state of the system can be represented as a vector within that space.

To apply the state-space methodology, a model must be expressed in the following format called the *discrete-time state-space representation*:

$$x_{t+1} = \Phi x_t + w_t \tag{3.3}$$

$$y_t = A x_t + v_t \tag{3.4}$$

$$cov(w_t) = Q,\ cov(v_t) = R \tag{3.5}$$

where x_t is the system state; y_t is the observed data; Φ is the state transition matrix that relates x_t to x_{t+1} ; A is the linear connection or interaction matrix that relates x_t to y_t; w_t and v_t are uncorrelated white noise sequences whose covariance matrices are Q and R respectively. The first equation, called the *state equation*, describes the dynamic behaviour of the state variables. The other equation is called the *observation equation* and it relates the system states to the observation.

In order to represent the discrete-time model in the state-space form, the discrete-time equation:

$$x_{t+1} = F x_t + \varepsilon_t, \tag{3.6}$$

is simply substituted into the state Equation 3.3 by setting $\Phi = F$, $w_t = \varepsilon_t$ and $Q = E$. A direct mapping between the system states and the observations is created by setting $A = I$. The state transition matrix Φ (functional equivalent to F) is the parameter of interest as it relates the future response of the system to the present state and governs the dynamics of the entire system. However, in this methodology, the covariance matrices Q and R are of secondary interest and are fixed to small values to reduce the number of model's parameters.

3.2.4 The Discrete Kalman Filter

As an introductory section to the Kalman filter, this subsection of the chapter will first give a brief description of the Kalman filter and its use in computing the optimal estimates of a state-space model.

In 1960, a paper describing a recursive solution to discrete-data linear filtering problem was published by R.E. Kalman (Kalman, 1960), this proposed method is known as the Kalman filter. Being one of the most widely used techniques for modelling stochastic processes, the Kalman filter is a set of mathematical equations that provides an efficient computational process to estimate the state of a process in a way that minimises the mean of the squared error. In effect, the Kalman filter is simply an optimal recursive data processing algorithm that is also an optimal estimator for a dynamic linear system (Y. Ho & Lee, 1965; Maybeck, 1972, 1979).

Different studies have revealed that the filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modelled system is unknown (Choi & Roy, 2006; Sun, Tian, & Wang, 2008; D. Zhang & Ionescu, 2009). The optimality aspect of the Kalman filter in this case is of course in respect to virtually any criterion that makes sense. One aspect of this optimality shows that the Kalman filter incorporates all information that can be provided to it

(global modelling) (Brown, 1983). It processes all information, regardless of their precision, to estimate the current values of the observed variables, with use of:

- Knowledge of the system and measurement device dynamics;
- The statistical description of the system noise, measurement errors and uncertainty in the dynamics models;
- Any available information about initial conditions of the observed variables.

The word *recursive* mentioned before simply means that unlike other data processing methods, Kalman filter does not require previous data to be kept in a memory, therefore it will not reprocess again when new measurements of observed variables become available. This is a very important characteristic of a filter implementation.

Figure 3.2 shows how the Kalman filter can be used to find the optimal estimate of the system state. By associating the system state to how different time-series interact with each other in a specific system (e.g. stock market system, weather system, biological system, etc.), the idea is to use a Kalman filter to estimate or model how these time-series are "connected", in respect to how they interact with each other at a specific time moment and how this connection is changing dynamically over time.

The Kalman filter combines all available measurement data (i.e. stock prices, stock market indexes, gene expressions value, etc.) plus prior knowledge about the system, to produce an estimate of the system state in such a manner that the error is minimised statistically.

One may ask, if it is known that the system under investigation is a non-linear dynamic system then what is the point of implementing the Kalman filter, being a linear system modelling approach, to estimate the system state.

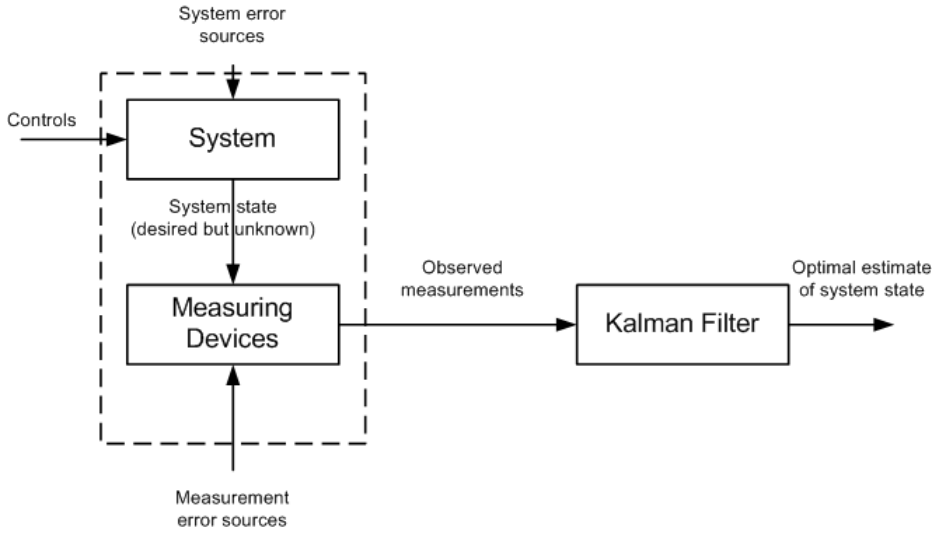


Figure 3.2. General concept of the Kalman filter implementation for system state estimation (Maybeck, 1979).

Answering this, Maybeck in his book (Maybeck, 1979) has pointed out some basic assumptions in the Kalman filter. One of them is about the justifiability of the Kalman filter for linear system modelling. Even though in the real world most systems are non-linear, it is a typical engineering approach to linearise some nominal points or trajectory to achieve a perturbation model or error model. Linear systems are desirable in that they are easier to be manipulated with engineering tools and linear system or differential equation theory is much more complete and practical than the non-linear equivalents.

The next chapters of the thesis will show that by considering complex dynamic systems of real world phenomena, i.e. the stock market indexes and the weather conditions, as linear systems, the proposed methodology is able to extract important and useful knowledge about interactive patterns between multiple time-series.

The fact is, there are means of extending the Kalman filter concept to model non-linear applications which has been done through the development of the extended Kalman filter (Jeen-Shang & Yigong, 1994; K. Kim, Lee, &

Park, 2009; Welch & Bishop, 1995). Hence, based on the actuality that most systems in the real world situation are non-linear, the use of the extended Kalman filter should then be addressed in future work.

3.2.5 The State-Space Model Estimation through Kalman Filtering

The Kalman filter estimates the states of a state-space model by using a form of feedback control. This means that the filter works by estimating the state at some point in time and then obtains feedback in the form of (noisy) measurement. Therefore, basically there are two important groups of equations in the Kalman filter formulation. The first one is the group of *time update* equations and it is responsible for projecting forward (in time) the current system state and error covariance estimates to obtain *priori* estimates for the next step. The other group of the equations is the group of *measurement update* equations. These equations act as *corrector* equations, applied to the estimated trajectories once the actual measurement value is available. In this methodology, these forward recursions are used to compute the state estimates. An example of a forward recursion is the following equation:

$$x_{t+1} = Ax_t + Bu_t + w_t, \tag{3.7}$$

where w_t denotes the process noise and is ignored in this methodology.

Indeed, the final algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems. A general concept of this algorithm can be seen in Figure 3.3, while the complete mathematical representation of the Kalman filter operation is given in Equations 3.8 to 3.12.

The specific equations for the time updates are presented below:

$$\hat{x}^-_{t+1} = A\hat{x}_t + Bu_t \tag{3.8}$$

$$P^-_{t+1} = AP_tA^{\mathrm{T}} + Q \tag{3.9}$$

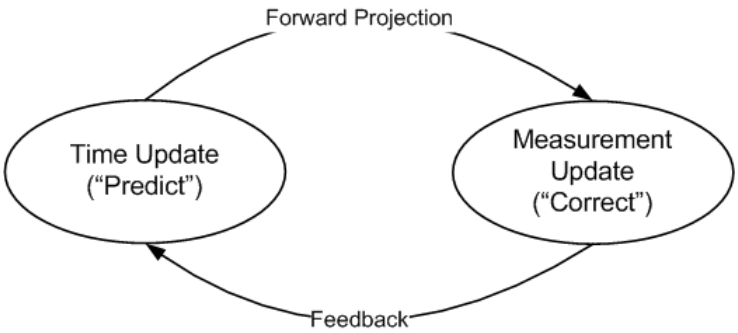


Figure 3.3. Component of equations in the Kalman filter (Welch & Bishop, 1995).

while the equations for measurement updates are presented below:

$$\hat{x}_{t+1} = \hat{x}^{-}_{t+1} + K_{t+1}\left(z_{t+1} - H\hat{x}^{-}_{t+1}\right) \tag{3.10}$$

$$K_{t+1} = P^{-}_{t+1}H^{\mathrm{T}}\left(HP^{-}_{t+1}H^{\mathrm{T}} + R\right)^{-1} \tag{3.11}$$

$$P_{t+1} = \left(I - K_{t+1}H\right)P^{-}_{t+1} \tag{3.12}$$

From the equations it can be observed how the time update equations project the state and covariance estimates forward from time moment t to moment $t+1$. The $k \times k$ matrix A, where k is the number of time-series being examined, in Equation 3.8 relates the state at current time moment t to the state at future moment $t+1$. This formula can be associated with the form of the discrete-time state-space representation in Equation 3.3. The $k \times 1$ matrix B relates the optional control input $u \in \Re^{l}$ where in the discrete-time state-space representation is considered to be a very small value close to 0, to the state x. The $m \times k$ matrix H in the measurement equations such as Equation 3.10 relates the states to the observed data (measurements) z_{t+1}. Here, this formula is also being associated to the discrete-time state-space representation in Equation 3.4.

The first task during the measurement update is to compute the Kalman gain, K_{t+1}, and the next step is to actually measure the process to obtain z_{t+1}, and then go on to generate a *posteriori* state estimate by incorporating the measurement as shown in Equation 3.10. The final step is to obtain a *posteri-*

ori error covariance estimate through Equation 3.12. After each measurement update, the process is repeated with the previous *posteriori* estimates used to project or predict the new a *priori* estimates. This recursive characteristic is one of the very appealing features of the Kalman filter, where the filter recursively conditions the current estimate on all of the past measurements.

3.3 Dynamic Interaction Network: DIN

DIN is a global modelling technique for multiple time-series that utilises the Kalman filter and the EM algorithm to perform a state-space estimation modelling by extracting a transition matrix from the inter-related multiple time-series data (Widiputra, Pears, Serguieva, & Kasabov, 2009). This transition matrix is then exploited to construct a relationship model in the form of fully connected graphs that reveal dynamic interactions between observed variables. This approach is developed based on a method proposed by Kasabov, Chan, Jain, Sidorov, and Dimitrov (2004) to identify interdependencies between genes.

The procedure to extract such a transition matrix to build interaction network in DIN is outlined as follows:

- Step 1: for $i = 1, ..., t$ of data set $\mathbf{X}$ where $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^i)$, estimate trajectories of $\mathbf{x}^{i+1}$ with the Kalman filter time update equation as follows:

$$\hat{\mathbf{y}}_e^{i+1} = \mathbf{A}\hat{\mathbf{y}}_u^i, \tag{3.13}$$

  where $\hat{\mathbf{y}}_e^{i+1}$ is the estimated trajectory of $\mathbf{x}^{i+1}$ at time-point i, and $\hat{\mathbf{y}}_u^i$ is the updated $\hat{\mathbf{y}}_e^i$ given the actual measurement of $\mathbf{x}^i$.

  In this methodology, as the initial value of $\hat{\mathbf{y}}_u^i$ where $i = 1$, value of $\mathbf{x}^1$ is used as $\hat{\mathbf{y}}_u^1$. Here, $\hat{\mathbf{y}}_u^i$ is analogous to $\hat{x}_{t+1}$ in Equation 3.10.

  At the beginning of the filtering process, transition matrix $\mathbf{A}$ can be set

to a random $n \times n$ matrix or an identity matrix where n is the number of time-series being observed. In this methodology, the second approach is used. Afterwards the error covariance of $\hat{\mathbf{y}}_e^{i+1}$ is calculated based on the Kalman filter equation (as in Equation 3.12) as follows:

$$\mathbf{P}_e^{i+1} = \mathbf{A}\mathbf{P}_u^i\mathbf{A}^{\mathrm{T}} + \mathbf{Q}, \tag{3.14}$$

where $\mathbf{P}_u^i$ is the updated previous error covariance, and Q is an $n \times n$ process noise covariance matrix which in this methodology is set to a very small value, i.e. $1e^{-5}$. In addition, at the beginning of the filtering process $\mathbf{P}_u^1$ is also set to a very small value, i.e. $1e^{-5}$.

The next process is to make corrections to the estimated trajectory $\hat{\mathbf{y}}_e^{i+1}$ (*updating*), by taking into account the actual measurement of $\mathbf{x}^{i+1}$ as follows:

$$\hat{\mathbf{y}}_u^{i+1} = \hat{\mathbf{y}}_e^{i+1} + \mathbf{K}^{i+1}(\mathbf{x}^{i+1} - \mathbf{H}\hat{\mathbf{y}}_e^{i+1}), \tag{3.15}$$

$$\mathbf{K}^{i+1} = \mathbf{P}_e^{i+1}\mathbf{H}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{P}_e^{i+1}\mathbf{H}^{\mathrm{T}})^{-1}, \tag{3.16}$$

$$\mathbf{P}_u^{i+1} = (\mathbf{I} - \mathbf{K}^{i+1}\mathbf{H})\mathbf{P}_e^{i+1}, \tag{3.17}$$

where $\mathbf{I}$ is an $n \times n$ identity matrix, $\mathbf{H}$ is a state matrix that relates the estimates state and the actual measurement. However, as we assumed that the estimates state is equal to the actual measurement, then in this methodology $\mathbf{H}$ is defined as an identity matrix.

Additionally, $\mathbf{R}$ is the actual measurement noise covariance matrix being set to a very small value, i.e. $1e^{-5}$;

- Step 2: after all estimates of the states $\mathbf{y}_e$ have been calculated the next step is to improve the estimates of the states $\mathbf{y}_e^i$ by incorporating the actual measurements. This process is known as the Kalman smoothing process (Hartikainen & Sarkka, 2010; Moiseenko & Saenko, 1994; Sarkka, Vehtari, & Lampinen, 2007). The smoothed estimates of the

states $\mathbf{y}_e^i$ is denoted as $\hat{\mathbf{y}}_s^i$.

In general, given t actual measurements of $\mathbf{x}$ which is denoted by $\mathbf{X}$, moving backward starting from $i = t, t-1, ..., 2$ smoothed values of $\hat{\mathbf{y}}_e^i$ and its error covariance are calculated by these equations below,

$$\hat{\mathbf{y}}_s^{i-1} = \hat{\mathbf{y}}_u^{i-1} + \mathbf{J}^{i-1}(\hat{\mathbf{y}}_s^i - \hat{\mathbf{y}}_e^i), \tag{3.18}$$

$$\mathbf{J}^{i-1} = \mathbf{P}_u^{i-1}\mathbf{A}^{\mathrm{T}}[P_e^i]^{-1}, \tag{3.19}$$

$$\mathbf{P}_s^{i-1} = \mathbf{P}_u^{i-1} + \mathbf{J}^{i-1}(\mathbf{P}_s^i - \mathbf{P}_e^i)[\mathbf{J}^{i-1}]^{\mathrm{T}}, \tag{3.20}$$

which is initialised by,

$$\hat{\mathbf{y}}_s^t = \hat{\mathbf{y}}_u^t \text{ and } \mathbf{P}_s^t = \mathbf{P}_u^t$$

The smoothing process described above applies only to the estimated data and not to the historical empirical data, and concerns only the state from a single previous step. Therefore, it can be implemented in an on-line setting.

- Step 3: the next step of the process is the parameters estimation of the model using the EM algorithm. Since in this methodology our main interest is the $\mathbf{A}$ transition matrix we hold the other parameters of the Kalman filter model $\mathbf{Q}, \mathbf{H}$, and $\mathbf{R}$ at their initial value.

  Considering the states of $\mathbf{y}^i$ as hidden variables, while $\mathbf{X}$ are the observations or actual measurements, then based on Welling (2001) the only sufficient statistics that need to be calculated in the E-step (expectation step) are:

$$\mathbf{E}[\mathbf{y}^i \big| \mathbf{X}] = \hat{\mathbf{y}}_s^i \qquad i = 1, ..., t \tag{3.21}$$

$$\mathbf{E}[\mathbf{y}^i\mathbf{y}^i \big| \mathbf{X}] = \mathbf{P}_s^i + \hat{\mathbf{y}}_s^i\hat{\mathbf{y}}_s^i \qquad i = 1, ..., t \tag{3.22}$$

$$\mathbf{E}[\mathbf{y}^i\mathbf{y}^{i-1} \big| \mathbf{X}] = \mathbf{C}_s^i + \hat{\mathbf{y}}_s^i\hat{\mathbf{y}}_s^{i-1} \qquad i = 2, ..., t \tag{3.23}$$

Here $\mathbf{C}_s$ is the *lag-one covariance smoother* of the smoothed trajectories defined as follows (Welling, 2001):

$$\mathbf{C}_s^{i-1} = \mathbf{P}_u^{i-1}[\mathbf{J}^{i-2}]^{\mathrm{T}} + [\mathbf{J}^{i-1}]^{\mathrm{T}}(\mathbf{C}_s^i - \mathbf{A}\mathbf{P}_u^{i-1})[\mathbf{J}^{i-1}]^{\mathrm{T}}, \tag{3.24}$$

which is initialised by:

$$\mathbf{C}_s^t = (\mathbf{I} - \mathbf{K}^t\mathbf{H})\mathbf{A}\mathbf{P}_u^{t-1}. \tag{3.25}$$

Consequently, total statistics for the complete observations are calculated as follows:

$$\mathbf{Stat_1} = \sum_{i=1}^{t} \mathbf{E}[\mathbf{y}^i\mathbf{y}^i\big|\mathbf{X}] \tag{3.26}$$

$$\mathbf{Stat_2} = \sum_{i=2}^{t} \mathbf{E}[\mathbf{y}^i\mathbf{y}^{i-1}\big|\mathbf{X}] \tag{3.27}$$

For the M-step (maximisation step) the transition matrix $\mathbf{A}$ is then updated to maximise the expectation value of the joint probability density function over the posterior density using this equation below (Welling, 2001):

$$\mathbf{A_{new}} = \mathbf{Stat_2} \times [\mathbf{Stat_1}]^{-1}. \tag{3.28}$$

Alternating E-steps and M-steps will converge to the maximum likelihood estimates of matrix $\mathbf{A}$. By incorporating the recursive property of the equations, this procedure can be utilised to estimate the optimal state of a dynamic system in an on-line mode;

- Step 4: the next step of the procedure is the computation of total log-likelihood of the system based on the model's parameters estimation. The total log-likelihood is calculated as follows:

$$\mathbf{L} = \sum_{i=1}^{t} \log(\det(\zeta^i)) + [\mathbf{e}^i]^{\mathrm{T}}[\zeta]^{-1}\mathbf{e}^i, \tag{3.29}$$

where

$$\mathbf{e}^i = \mathbf{x}^i - \mathbf{H}\hat{\mathbf{y}}_e^i \tag{3.30}$$

$$\zeta_i = \mathbf{H}\mathbf{P}_e^i\mathbf{H}^{\mathrm{T}} + \mathbf{R} \tag{3.31}$$

- Step 5: while $(\mathbf{A} - \mathbf{A_{new}}) > S_{cr}$, where S_{cr} is the stopping criteria of the transition matrix estimation process, the process goes back to Step 1, in which new estimated trajectories will be recalculated, where now $\mathbf{A} = \mathbf{A_{new}}$. In this methodology S_{cr} is set to a very small value $1e^{-5}$ to confirm that the estimation process would only stop when transition matrix $\mathbf{A}$ converges, that is $\mathbf{A_{new}} \approx \mathbf{A}$.

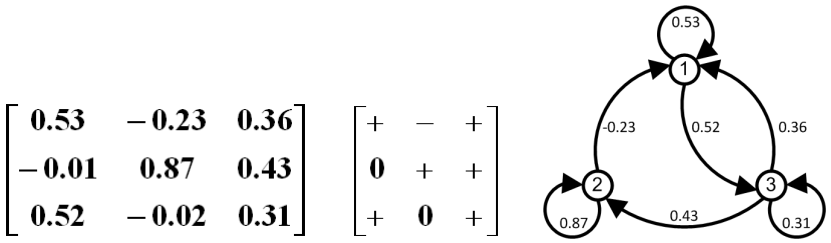
  Another stopping criteria that can be implemented in the parameters estimation process is the total log-likelihood value, $\mathbf{L}$. However, this approach requires human or user involvement to decide whether current total log-likelihood value is sufficient enough for the estimation process to stop.

  Else, the process continues to Step 6;

- Step 6: as the estimated transition matrix $\mathbf{A}$ converges, the algorithm keeps the value of $\mathbf{A}$ that represents the interactions between multiple time-series under examination.

The DIN is useful in detecting influences and evaluating the degree to which a time-series influences or is being influenced by others. Through capturing dynamic influences among time-series, one should be able to predict the behaviour of multiple time-series simultaneously.

The construction of a DIN from a transition matrix is illustrated in Figure 3.4. Here, only the most significant time-series relationships defined by a threshold value are elucidated from the transition matrix. All transitions $x_{i,j}$ with values in $\mathbf{A}$ that are greater than 0.1 are flagged as positive in Figure 3.4b, i.e. time-series x_i is influencing time-series x_j. By analogy, values



(a) (b) (c)

Figure 3.4. Illustration of interaction network construction in DIN. (a) is the transition matrix; (b) is the corresponding influence matrix when a threshold of 0.1 is used; (c) is the interaction network.

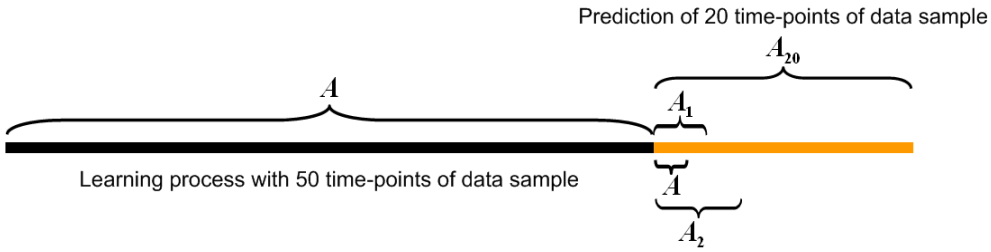


Figure 3.5. Illustration of incremental learning in DIN. Transition matrix **A** is being re-estimated as new observations become available.

below −0.1 are labelled as negative in Figure 3.4b. The values and direction of influence are reflected in the directed network diagram depicted in Figure 3.4c.

The ability to capture time varying patterns of inter-relationships between multiple time-series is an important pre-requisite to predicting the future values of the series. The DIN meets this requirement as it is able to evolve its structure dynamically as new data comes. As such a modelling and prediction process with on-line learning is implemented, whereby the transition matrix and the DIN model are updated at every time step.

As shown in Figure 3.5, a DIN model is initially trained over a certain number of time moments, e.g. 50 time-points of data. Afterward, extracted DIN model is used to predict the next time moment's values. This DIN model is then updated at the following time-point with new data arriving whereupon

the updated DIN model is used to predict values for next time moment and so forth. This process is performed simply by using the recursive measurement update property of the Kalman filter and the EM algorithm as outlined in the algorithm above. The learning process then continues into the future, where prediction and training operations are interleaved with each other.

3.4 Experiments on Synthetic Data

In order to evaluate the capability of DIN in extracting patterns of dynamic interactions from a set of multiple time-series data, a synthetic data set consisting of a number of interrelated time-series is constructed in this study. To accomplish such a task, three different time-series trajectories regulated by the polynomial functions defined below, are created.

$$f_1(x) = \frac{1}{30}\left(x^6 - 22.5x^5 + 189.5x^4 - 735.75x^3 + 1296.56x^2 - 822.66x + 180\right)$$
$$f_2(x) = 0.08x^6 - 1.8x^5 + 15.16x^4 - 58.86x^3 + 103.72x^2 - 65.81x + 8$$
$$f_3(x) = -0.1x^6 + 2.25x^5 - 18.95x^4 + 73.5x^3 - 129x^2 + 85x - 7.5$$

Trajectories of these three initial series are depicted in Figure 3.6. However, to increase the complexities of the trajectories and to introduce the component of interdependency, another four series are generated from these initial three series. The four series are generated based on the calculations below:

$$\begin{aligned} S_1 &= f_1(x) \times f_2(x) \\ S_2 &= f_1(x) \times f_3(x) \\ S_3 &= f_2(x) \times f_3(x) \\ S_4 &= (f_3(x) \times S_3)/10 \end{aligned} \tag{3.32}$$

These four series are then used in this study to evaluate the ability of the proposed methodology to extract patterns of dynamic interactions between multiple time-series. Trajectories of these four series are depicted in

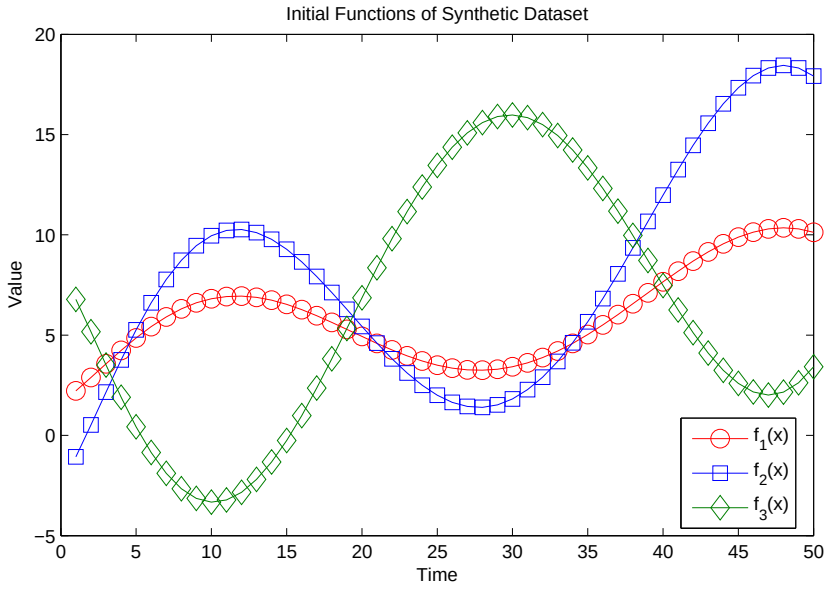


Figure 3.6. Trajectories of the original polynomial functions used to construct the synthetic data set consisting of 4 time-series data.

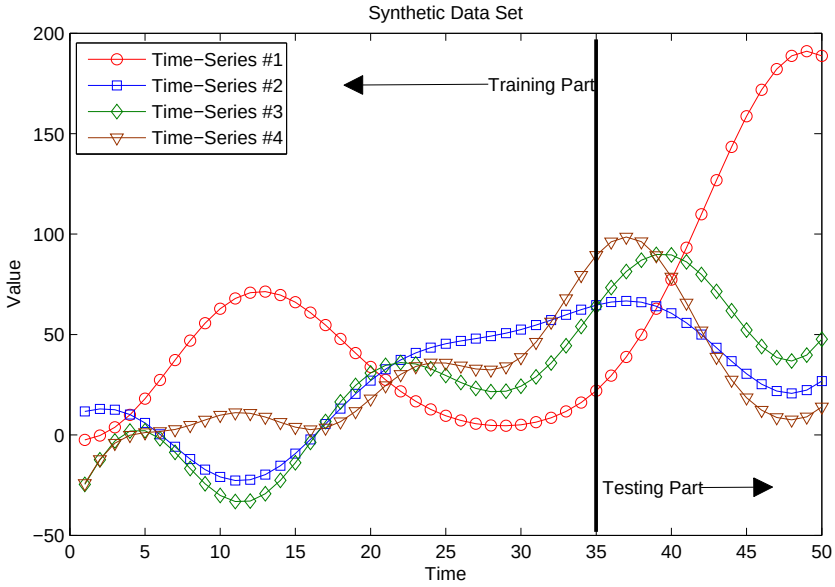


Figure 3.7. Trajectories of the synthetic data set.

Figure 3.7. Furthermore, an assessment as to whether the dynamics of interactions between multiple time-series could be used to perform simultaneous multiple time-series prediction is also conducted.

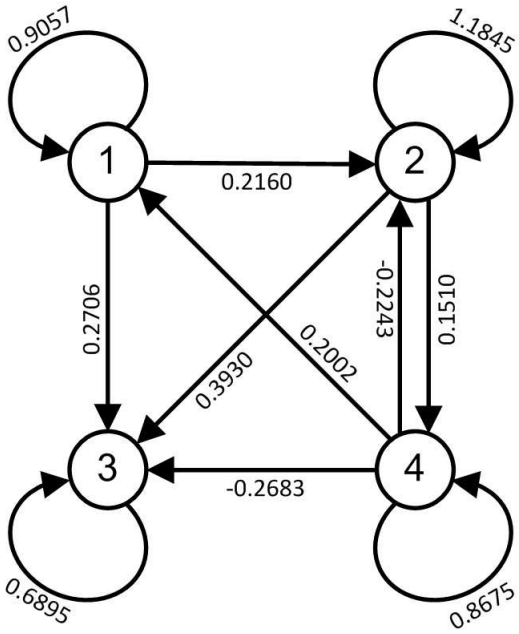
The experiment is performed by splitting the data set into two different parts, one for training and the other for testing. The training data set is utilised to learn and extract the initial transition matrix using the algorithm explained in Section 3.3. Whereas, the test data set is utilised to evaluate whether DIN is capable of capturing changes in pattern of interactions between time-series. Here, the first 35 time-points of the complete data set is used as the training data set, and the rest of 15 time-points are defined as the test data set.

Feeding the training data set to DIN produces the initial transition matrix as shown in Figure 3.8a. Utilising this transition matrix, a graph depicting interaction network between the observed series is then generated as illustrated in Figure 3.8b. The interaction network model in Figure 3.8 (series are numbered according to their node numbers in the graph) shows that during the first 35 time moments Series #1 influenced Series #2 and Series #3 positively. This outcome is in agreement with the process of how Series #1, Series #2 and Series #3 were constructed. Both Series #2 and #3 were constructed based on f_3 being multiplied to f_1 and f_2 which are the two functions used to construct Series #1. Therefore, it is expected to discover the existence of positive interactions between the three series.

In addition, the interaction network also shows that Series #3 is being positively influenced by Series #2. Yet again, this outcome also relates to the process of how Series #3 was generated. The initial component used to construct Series #3 and #2 is f_3. However, instead of being multiplied to functions that retain opposite form, to generate Series #2 and #3 f_3 is multiplied to functions that hold similar form, i.e. f_1 and f_2 respectively. Therefore, it is reasonable to have an interaction network model which shows

$$A = \begin{bmatrix} 0.9057 & 0 & 0 & 0.2002 \\ 0.2160 & 1.1845 & 0 & -0.2243 \\ 0.2706 & 0.3930 & 0.6895 & -0.2683 \\ 0 & 0.1510 & 0 & 0.8675 \end{bmatrix}$$

(a) The transition matrix



(b) The interaction network

Figure 3.8. Extracted transition matrix and its interaction network from the training data set, the first 35 points of the synthetic data set.

the existence of positive relationships between the two series.

Furthermore, the graph recognises that Series #4 is the most influencing series compared to the other three series. Series#4 highly influenced Series#3, this is again in agreement with the process of how Series #4 was constructed. Equation 3.32 shows that Series #4 was constructed using Series #3 as one of its main components, and therefore the existence of strong relationship between the two series is expected. Additionally, Series #4 also gave a significant influence on Series #2. This is again in agreement with the construction

process that is of Series #4, where f_3 (being one of the components used to construct Series #2) was also used as one of the constructing components in Series #4.

At the same time, the interaction network also shows that Series #4 was influencing Series #1 as well. Even though in the construction process Series #4 did not directly interact with Series #1, an observation of the trajectories of the four series gives a clear view that in comparison to Series #2 and #3, Series #4 is trajecting in a more similar shape to that of Series #1. This actuality suggests that there might exist a positive relationship between Series #1 and #4 which is recognised by the interaction network model.

Additionally, some rules or understanding that can be extracted and learned based on the interaction network model for the prediction of upcoming values of the series under observation are:

- Series #1 received both positive internal and external influence. The external influence to Series #1 is coming from Series #4. The upcoming value of Series #1 is expected to increase by 0.9057 when current value of Series #1 goes up by one and by 0.2002 for a one-unit change in Series #4. This form of interdependency can be modelled mathematically as follows:

$$S_1(t+1) = 0.9057S_1(t) + 0.2002S_4(t) \tag{3.33}$$

- Series #2 received both external positive and negative influences from Series #1 and #4. Nevertheless, the upcoming value of Series #2 is also highly dependent to its current value. When current value of Series #2 and Series #1 goes up by one, it is expected that the upcoming value of Series #2 increases by 0.2160 and 1.1845 while at the same time it will decrease by 0.2243 for a one-unit change in Series #4. This relationship can be modelled in a linear equation model as follows:

$$S_2(t+1) = 0.2160S_1(t) + 1.1845S_2(t) - 0.2243S_4(t) \tag{3.34}$$

- In comparison to the other series, Series #3 is the most influenced series in the synthetic dynamic system under evaluation. Series #3 received both external positive and negative influences from the other series. The upcoming value of Series #3 will increase by 0.2706, 0.3930 and 0.6895 for a one-unit change in Series #1, #2 and itself at current time-point respectively. Additionally, when Series #4 goes up by one then it is expected that the upcoming value of Series #3 will decrease by 0.2683. A linear equation model for the prediction of Series #3 then can be formed as follows:

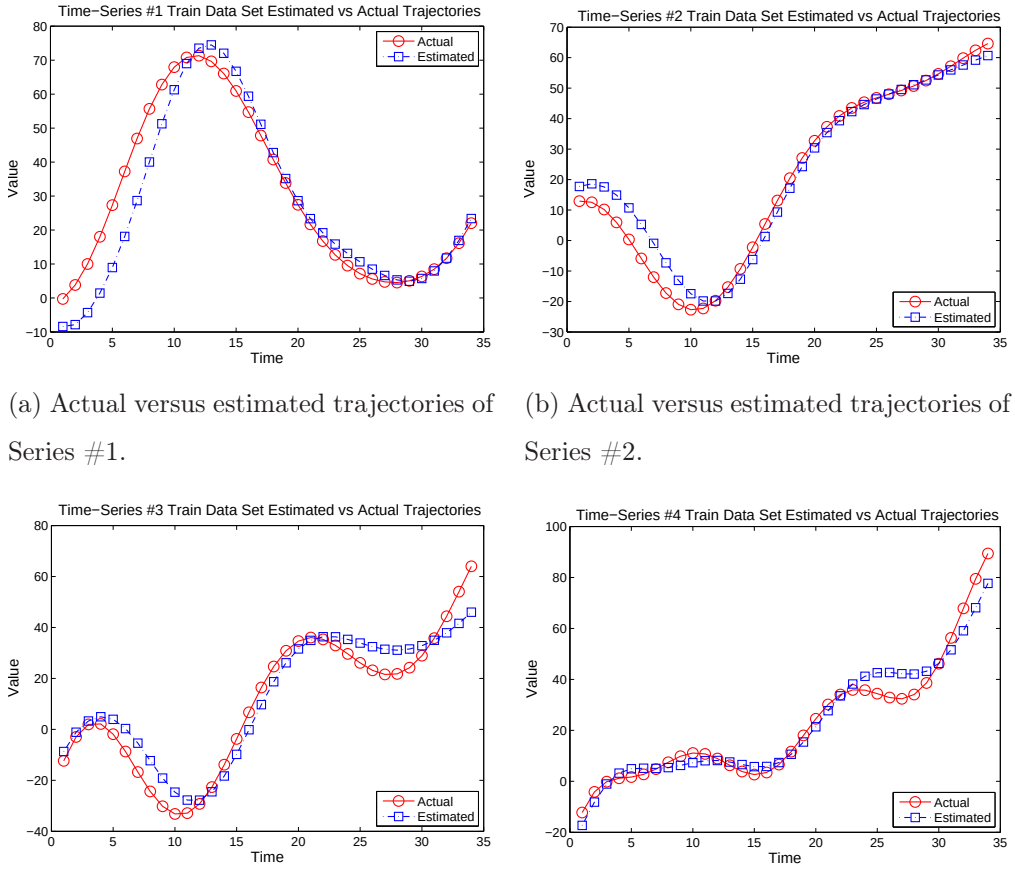
$$S_3(t+1) = 0.2706S_1(t)+0.3930S_2(t)+0.6895S_3(t)-0.2683S_4(t) \quad (3.35)$$

- Whilst the upcoming value of Series #4 is being considerably affected by its current value, it also received a significant external influence from Series #2. Series #4 is predicted to increase by 0.1510 for a one-unit change in Series #2 at current time-point and by 0.8675 for a one-unit change in itself. The upcoming value of Series #4 then can be estimated with this linear equation model:

$$S_4(t+1) = 0.1510S_2(t) + 0.8675S_4(t) \quad (3.36)$$

Summarising these findings, we can say that in the training period there exists a complex form of interdependencies between Series #1, #2, #3 and #4. However, it is found that in general Series #1 and Series #4 are the two series that influenced movement of the other series most significantly. Therefore, these two series can be identified as the main variables that govern the states of the synthetic dynamic system under evaluation. This conclusion is drawn from the interaction network model and is consistent with the manner in which the synthetic data set was constructed.

All of these outcomes indicate that DIN is capable of extracting an important and useful pattern of interactions from a set of multiple time-series data.



(a) Actual versus estimated trajectories of Series #1.

(b) Actual versus estimated trajectories of Series #2.

(c) Actual versus estimated trajectories of Series #3.

(d) Actual versus estimated trajectories of Series #4.

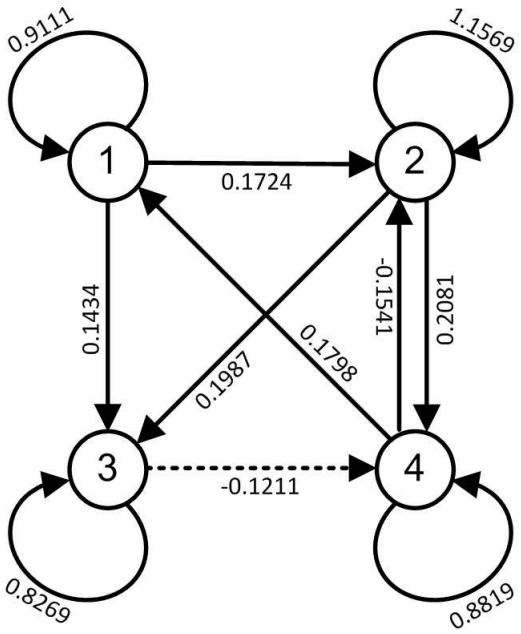
Figure 3.9. Plot of estimated trajectories of synthetic data set by DIN in the training period.

However, there exists a limitation in the methodology. The key constraint of DIN is that it is only capable to model interactions between series in a linear form. Therefore, it is a challenge to be able to extend the methodology to capture and model non-linear interactions between multiple time-series. This is a task that needs to be addressed in future research of implementing the extended Kalman filter to replace the currently used standard Kalman filter.

To evaluate whether the extracted interaction network model can be of help in predicting upcoming values of the observed series simultaneously, tra-

$$A = \begin{bmatrix} 0.9111 & 0 & 0 & 0.1798 \\ 0.1724 & 1.1569 & 0 & -0.1541 \\ 0.1434 & 0.1987 & 0.8269 & 0 \\ 0 & 0.2081 & -0.1211 & 0.8819 \end{bmatrix}$$

(a) The transition matrix



(b) The interaction network

Figure 3.10. Adapted transition matrix and its interaction network after 5 points from the test data set are added. A new interaction between Series #2 and Series #3 is represented by a dashed line. Initial transition matrix and its interaction network is shown in Figure 3.8

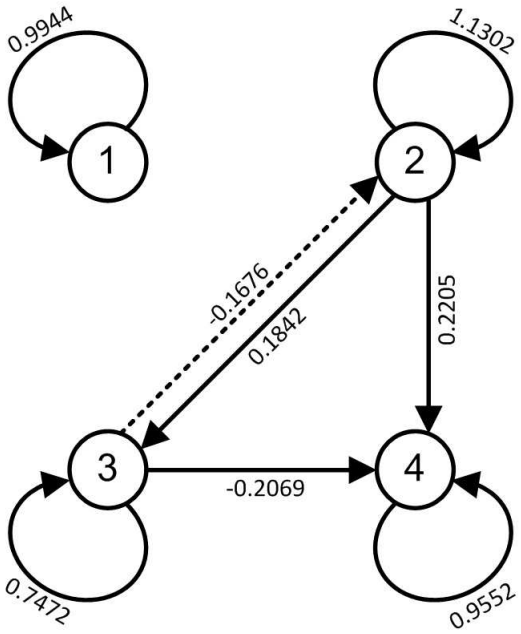
.

jectories of the first 35 time moments of the synthetic data set are re-generated using the transition matrix.

Figure 3.9 illustrates the estimated trajectories generated using the interaction network model compared to the actual ones. It is clearly seen that the estimated trajectories closely match the actual trajectories. This result

$$A = \begin{bmatrix} 0.9944 & 0 & 0 & 0 \\ 0 & 1.1302 & -0.1676 & 0 \\ 0 & 0.1842 & 0.7472 & 0 \\ 0 & 0.2205 & -0.2069 & 0.9552 \end{bmatrix}$$

(a) The transition matrix



(b) The interaction network

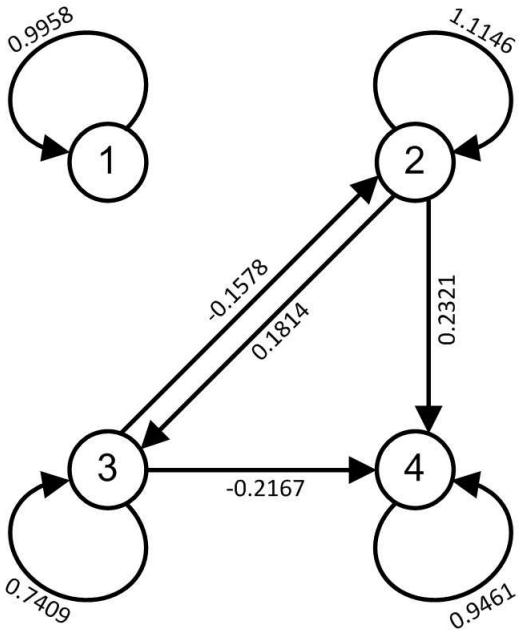
Figure 3.11. Adapted transition matrix and its interaction network after 10 points of test data set are added. Identified new interactions are denoted by the dashed-line.

confirms that the interaction network model captures important relationships between series that is useful in estimating their future states.

Following the capture of the dynamics of interactions between series, new instances from the test data set are then conferred to DIN. Figures 3.10, 3.11 and 3.12 respectively depict the interaction network model after 5, 10 and 15 points of test data entered the system. In general, these figures show that even though patterns of interactions between the four series is dynamically changing, as new interactions are emerging and existing interactions are dis-

$$A = \begin{bmatrix} 0.9958 & 0 & 0 & 0 \\ 0 & 1.1146 & -0.1578 & 0 \\ 0 & 0.1814 & 0.7409 & 0 \\ 0 & 0.2321 & -0.2167 & 0.9461 \end{bmatrix}$$

(a) The transition matrix



(b) The interaction network

Figure 3.12. Adapted transition matrix and its interaction network after 15 points of test data set are added. Identified new interaction is denoted by the dashed-line.

appearing, some interactions between the series remain stable over time.

For instance, Figure 3.10 shows that after feeding 5 points of test data, DIN identifies that Series #3 is no longer being influenced by Series #4 yet it is now Series #4 who received influence from Series #3. However, it is also observable that in general pattern of interactions between the four series remains stable.

Interestingly, a dramatic change happened to the interaction network model after feeding 10 points of test data, as shown in Figure 3.11. The interaction

Table 3.1

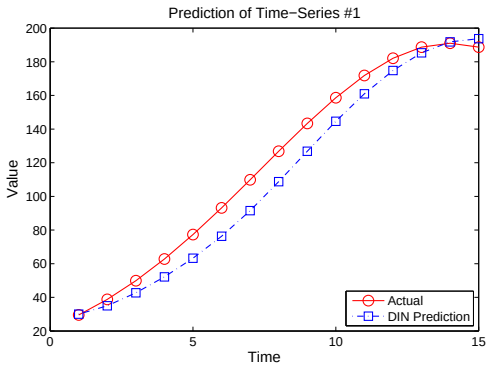
DIN prediction error rates in RMSE for the synthetic test data set.

No	Variable	DIN RMSE
1	Series #1	11.9658
2	Series #2	5.5067
3	Series #3	7.4662
4	Series #4	7.6928

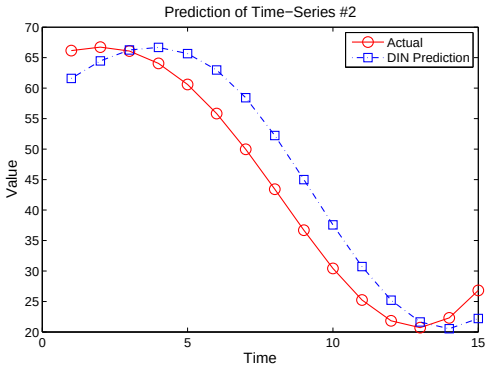
network model shows that even though interactions between Series #2, #3 and #4 remain stable, Series #1 now is not connected to the other series and received influence only from itself. This outcome is in agreement with the behaviour of the four series being observed. It is observable from the plot of trajectories of the four series, as depicted in Figure 3.7, that in the testing period trajectory of Series #1 is shifting towards a different direction that is of the other series. As Series #1 has now changed its course, it is then expected that the interaction network model between the four series is also changing, i.e. Series #1 is now being identified to have no interaction with the other series.

This condition materialises until the end of the testing period and therefore a similar interaction network model is discovered at the end of the testing period as illustrated in Figure 3.12. This outcome confirms DIN ability to capture the dynamics of interactions between series and then to evolve its structure based on recent behaviour of the series under examination.

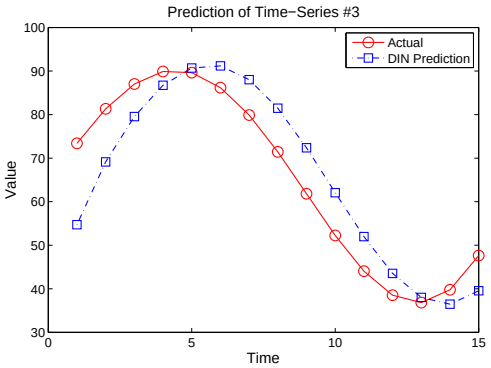
An evaluation of DIN performance when predicting movement of the test data set is illustrated in Figure 3.13. Additionally, the prediction error rates, as measured by the Root Mean Square Error (RMSE) are shown in Table 3.1. Both Figure 3.13 and Table 3.1 confirm the ability of the DIN to predict move-



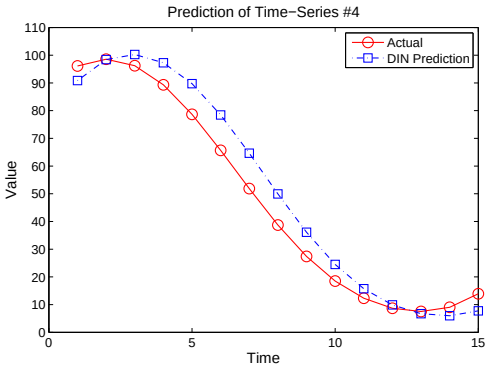
(a) Actual versus estimated trajectories of Series #1.



(b) Actual versus estimated trajectories of Series #2.



(c) Actual versus estimated trajectories of Series #3.



(d) Actual versus estimated trajectories of Series #4.

Figure 3.13. Plot of estimated trajectories of synthetic data set by DIN in the testing period.

ment of multiple time-series data simultaneously with a reasonable degree of accuracy.

Furthermore, a comparative analysis of prediction results obtained from the adaptive DIN model against those obtained from methods applied on single time-series prediction is also conducted. In this comparative analysis MLR, MLP, DENFIS and random walk models are put in place to predict movement of the four time-series individually. The random walk model is a naive time-series analysis method which assumes that next value of a time-

series is equal to current value. In the conducted experiments the random walk without drift model is defined simply by:

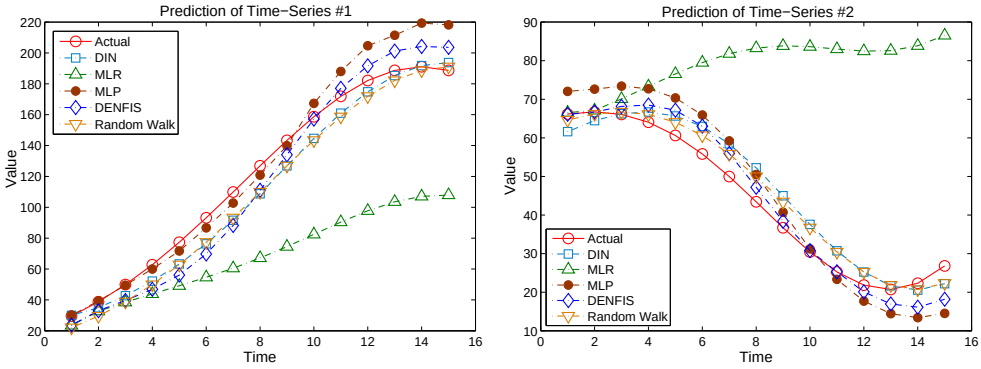
$$x_{t+1} = x_t. \tag{3.37}$$

Throughout the experiments conducted in this study the adaptive DIN model was trained with the training data set and then the test data set was utilised for adaptive training, whilst MLR, MLP and DENFIS are applied as batch models which were trained with the training data set and then tested on the test data set with no incremental learning. Additionally, due to its nature (as defined in Equation 3.37 the random walk model is applied as an incremental model.

Figure 3.14 depicts the comparison of the actual trajectories to the predicted trajectories computed by DIN, MLP, MLP, DENFIS and random walk.

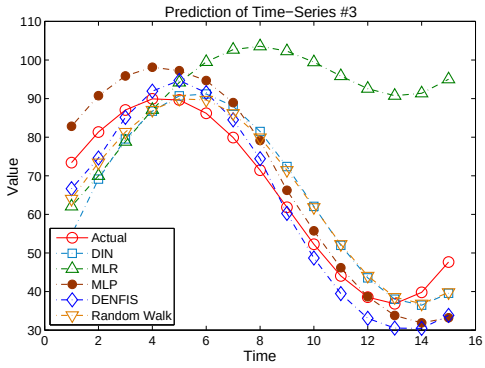
Table 3.2 outlines the calculated RMSE for each method. It is clearly seen that in general DIN performs better than the other methods applied on single time-series prediction. This outcome indicates that predicting movement of multiple time-series simultaneously by considering the pattern of interactions leads to a more accurate result compared to predicting movement of a single time-series individually.

Nevertheless, it is also observable in Table 3.2 that random walk model in some cases offers better prediction accuracy compared to the other methods, including DENFIS and DIN. However, as this model simply assumes that the next value of a variable being estimated is equal to its current value, random walk produces a shadow plot of the observed data, lagging exactly one period behind and providing no useful knowledge related to the observed system. Therefore, this model can be assumed to have no predictive power (Widiputra et al., 2011a). Moreover, in the conducted experiments DENFIS was incapable of performing adaptive learning process as it was applied as a batch model. This might be the key reason of having prediction results which

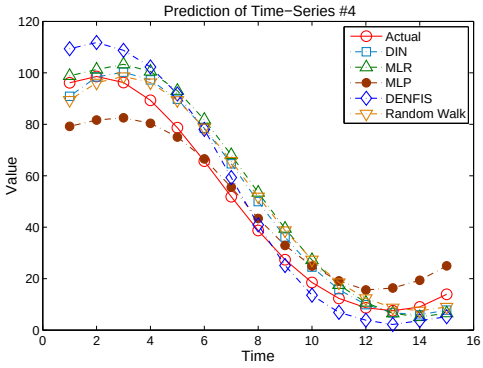


(a) Estimated trajectories of Series #1.

(b) Estimated trajectories of Series #2.



(c) Estimated trajectories of Series #3.



(d) Estimated trajectories of Series #4.

Figure 3.14. Plot of estimated trajectories of synthetic data set by DIN versus prediction by other widely-used methods applied on single time-series prediction in the testing period.

show that in general the random walk model is superior to DENFIS.

3.5 Conclusion

Given a set of measurements of multiple time-series, the DIN model is capable of extracting the dynamics of interactions between these observed variables by modelling the governing interdependencies as discrete-time approximation of first-order differential equations.

The methodology treats the actual trajectories or actual observed mea-

Table 3.2

Comparison of DIN prediction error rates against methods applied on single time-series: MLR, MLP, DENFIS and Random Walk, in RMSE.

No	Variable	DIN	MLR	MLP	DENFIS	Random Walk
1	Series #1	11.9658	62.0580	15.1441	14.3226	12.6634
2	Series #2	5.5067	43.7277	7.5743	4.6464	4.3419
3	Series #3	7.4662	37.0639	7.6572	6.1003	6.4474
4	Series #4	7.6928	10.1890	8.7729	8.8631	8.4156

surements as a set of unobserved or hidden variables using the state-space representation, and then applies a reliable and stable estimator algorithm, i.e. the EM algorithm, to compute the transfer function that maps past system states to present states. Being a set of mathematical equations, the Kalman filter provides a rigorous and computationally efficient way of estimating the future states of given data when the state-space representation is applied to model the system.

Experiments with synthetic data reveal some insights about the applications of the Kalman filter and the EM algorithm as the main components of the DIN:

1. Kalman filter is a robust computing method that can be used to estimate the actual trajectories of multiple time-series data even when data is only available at irregular time steps;

2. The transition matrix provides a valuable method of modelling inter-relationships between multiple variables which can be used to construct an interaction network model;

3. The interaction network model not only exposes essential relationships;

it can also be employed to predict future values of observed variables with an acceptable degree of accuracy.

Additionally, in order to evaluate its capability for dealing with real-world data, in the upcoming chapters of the thesis, DIN is also implemented to extract important pattern of interactions from multiple time-series data from real world applications, such as the finance and environment data sets.

Nevertheless, as DIN is only capable of modelling the dynamics of interactions between multiple time-series in a linear form, an interesting task would be to explore the possibilities of implementing the extended Kalman filter that was developed, to estimate systems with non-linear relationships between the contributing variables.

Chapter 4

Localised Trend Model for Multiple Time-Series Analysis and Modelling

4.1 Introduction

The common realisation of inductive reasoning is the construction of global models that cover the entire problem space, e.g. a regression formula, a neural network (Calcagno et al., 2010; Popescu, Balas, Perescu-Popescu, & Mastorakis, 2009; Rossi & Conan-Guez, 2005), an SVM (Cristianini & Shawe-Taylor, 2000; Fung & Mangasarian, 2001; Tong & Chang, 2001), etc. Global models are built using all historical data and thus can be used to predict future trends. However, the trajectories that global models produce often fail to track localised changes that take place at discrete points in time. This is due to the fact that trajectories produced by global models tend to smooth localised deviations by averaging the effects of such deviations over a long period of time (Widiputra et al., 2011a).

In reality, localised disturbances may be of great significance as they capture the conditions under which a time series deviates from the norm. For example, financial markets react very favourably when interest rates are cut or when better than expected Economic fundamentals are announced by a Government under which they operate. To accurately capture such phenomena

requires a discontinuity in the global trajectory function and this goes against the fundamental design philosophy behind the construction of global models. Furthermore, it is of interest to capture similar deviations from a global trajectory that take place repeatedly over time, in other words to capture recurring deviations from the norm that are similar in shape and magnitude. Such localised phenomena can only be captured accurately by localised models that are built only on data that defines the phenomenon under consideration and are not contaminated by data outside the underlying phenomenon.

In relation to this, recent research in the field of machine learning has focused on model ensembles that use a mixture of models to achieve better overall accuracy. Several studies have reported that an ensemble of models works better than a single global model (Cevikalp & Polikar, 2008; Islam, Yao, Shahriar Nirjon, Islam, & Murase, 2008; H. Kim et al., 2002; Lei, Yang, & Wu, 2006; Nguyen, Abbass, & McKay, 2008; Pang, 2004; Yao & Liu, 1998, 1996; Zhou & Jiang, 2003). There are many strategies that are commonly used to create an ensemble: bagging (Hothorn & Lausen, 2005; H. Kim et al., 2002; Nanni & Lumini, 2006), boosting (Scholz & Klinkenberg, 2007; C. Zhang & Zhang, 2008) and clustering (Kasabov & Song, 2002) are well known strategies. Depending on the strategy used, the ensembles generally try to either generate a different view of a problem or break down the problem into smaller problems and tackle each problem independently or sometimes a mix of both methods are used.

In relation to the idea that a more detailed and accurate behaviour of the observed data set can be captured by constructing a number of localised models, a local modelling approach of multiple time-series analysis is proposed in this chapter, aiming to address the following two research questions: (1) Would creating sub-models that capture recurring behaviour in different time localities from multiple time-series result in a better prediction accuracy? (2) Do these localised models allow a more comprehensive understanding of the

nature of dynamic relationship between multiple variables of the dynamic system under observation?

4.2 General Concept of Local Model

Local models (Kasabov, 2001, 2007b; Lucks & Oki, 1999; Poggio, 1994; Yamada, Yamashita, Ishii, & Iwata, 2006) is a type of model ensemble that breaks down the problem into many smaller sub-problems, based on its position in the problem space. Local models can be built by grouping together data that has similar behaviour. For example when the value of a variable suddenly increases significantly and then maintains the increased value over a period of time, a natural cluster containing the time points that define this heightened activity can be defined. Different types of phenomena will define their own clusters. Models can then be developed for each cluster (i.e. local regressions) that will yield better accuracy over the local problem space covered by the model in contrast to a global model.

In local modelling, individual models are created to evaluate the output function for only a subset of the problem space, e.g. a set of rules over a number of clusters or a set of local regressions, etc. Having a set of local models offers greater flexibility as predictions can be either on the basis of a single model or, if needed, at a global level by combining the predictions made by the individual local models (Kasabov, 2007b), For an example see Figure 4.1. Additionally, it is expected that local models would enable us to capture recent trends in the data and relate them to similar behaviour from the past. This is in contrast to a global model that takes into account *all* past activity, thus resulting in diluting the effects of recent trends in the data (Widiputra et al., 2011a).

However, having clustering as the key process in this approach means that the quality of the clusters is an important foundation for this type of model

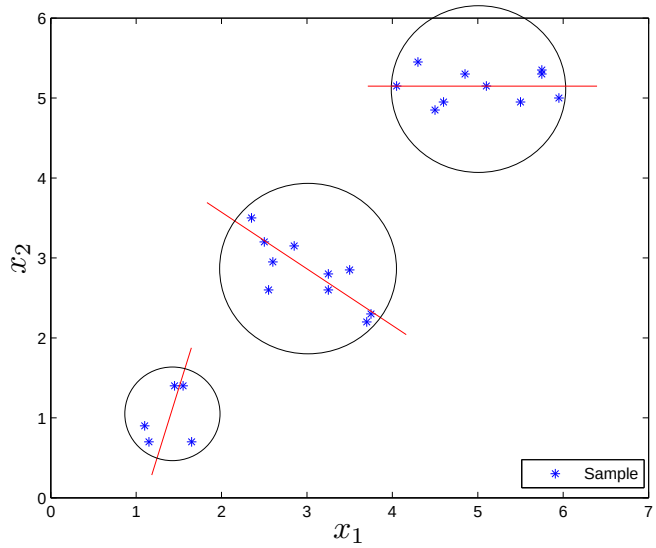


Figure 4.1. Illustration of local modelling to create localised linear regression models from clusters of sample data set in a 2-D space (Hwang, 2009).

as well. The data clustering parameters often need to be adjusted, according to the sub-model's requirements or the characteristic of the problem. Many models, such as linear regression need the number of input vectors to be significantly greater than the number of variables and, therefore, the clusters must be large enough to support this type of sub-model. Hence, local models may require more training data than global models to ensure that each sub-model is trained with sufficient number of input vectors.

4.3 Localised Trend Model: LTM

It is interesting to note that most of the research carried out in the field of time-series modelling and prediction have based their approach on the concept of inductive reasoning (Holland, Holyoak, Nisbett, & Thagard, 1989), in which a number of historical data samples are used to construct a single global model covering the entire training data set space. Nevertheless, as argued previously, local modelling is needed to cover subsets of the problem space that the global

model cannot cover with sufficient accuracy. The local model is another type of realisation of inductive reasoning. A system can be represented by a collection of local models trained on a given data set. However, when applied to new data, only one model or a subset of the relevant models will actually contribute to the solution.

This chapter outlines a methodology for the construction of local models for multiple time-series containing profiles of relationships between series from different time localities (Widiputra et al., 2011a, 2011c). The construction of local models in the proposed methodology consists of two main steps: (1) the continuous extraction of profiles of relationships between time-series over time, and (2) the detection and clustering of recurring trends of movement in time-series when a particular profile emerges.

The principal objective of the methodology is to construct a repository of profiles and recurring trends whose structure will dynamically evolve as changes take place in the observed non-stationary environment. This repository will then be utilised as a *knowledge-base* containing a key data resource to learn and understand the underlying behaviour of the system and to estimate future states of the system's variables; that is to perform a multiple time-series prediction. To realise such an objective a 2-level local modelling process is utilised within the proposed methodology.

The first level of local modelling deals with the extraction of profiles of relationships between series in a sub-space of the given multiple time-series data in which the methodology utilises a cross correlation analysis to elucidate the existence of relationships between pairs of time-series that influence each other. The second level of local modelling is used to capture and cluster recurring trends of movement that take place in time-series when a particular profile is emerging. Here, the methodology employs a non-parametric regression analysis in combination with the ECM (Q. Song & Kasabov, 2001). Detailed explanation of this local modelling method, which is termed as the *Localised*

Trend Model and denoted as LTM, is outlined in the upcoming sections.

4.3.1 Extracting Profiles of Relationships from Multiple Time-Series

Most of the work in clustering time-series data has concentrated on *sample clustering* rather than *variable clustering* (Rodrigues, Gama, & Pedroso, 2008). However, one of the key tasks in this methodology is to group together series or variables, and *not* samples that are highly correlated and have similar shapes of movement, as it is considered that multiple local models representing clusters of similar profiles will provide a better basis than a single global model for predicting future movements of the multiple time-series.

An example would be predicting the movement of 5 global stock market indexes i.e. New Zealand, Australia, Hong Kong, Japan and United States. If one is able to learn that at the current time moment New Zealand and Australia are moving together collectively, Hong Kong and Japan are progressing mutually, while the United States travels by itself, then it would be relevant to use only data of stock market indexes from the past which possesses the same profiles of relationships to predict future values of these stock market indexes, rather than to use the entire data set.

Algorithm 4.1 outlines the scheme for clustering together similar time series. The first step in extracting profiles of relationships between multiple time-series is the computation of cross-correlation coefficients between the observed time-series using Pearson's correlation analysis. Statistically significant correlations, which are determined through the use of the *t-test* with a confidence level of 95% are used. After the most significant correlations between time-series have been identified, the *Rooted Normalised One-Minus Correlation* (RNOMC) coefficients(R. Kim, Ji, & Wong, 2006; Rodrigues et al., 2008) (known henceforth as *normalised correlation* in this chapter) is cal-

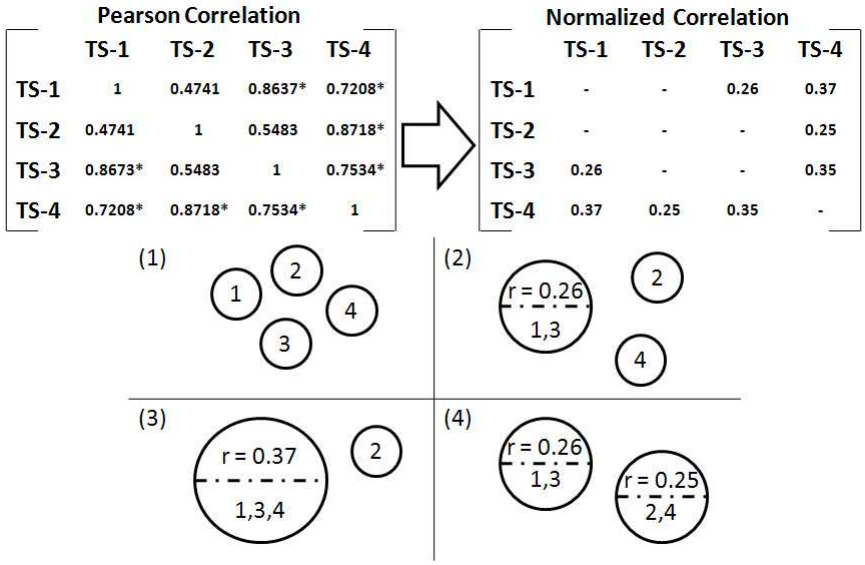


Figure 4.2. The Pearson's correlation coefficient matrix is calculated from a given multiple time-series data (TS-1,TS-2,TS-3,TS-4), and then converted to *normalised correlation*, Equation 4.1 before the profiles are finally extracted. Figure is extracted from (Widiputra et al., 2011a, 2011c).

culated to assess the degree of dissimilarity between a pair of time series (a, b). The normalised correlation is given by:

$$\text{RNOMC}(a, b) = \sqrt{\frac{1 - corr(a, b)}{2}} \tag{4.1}$$

The normalised correlation coefficient ranges from 0 to 1, in which 0 denotes high similarity and 1 signifies the opposite condition. In LTM, when two or more series are grouped into the same cluster, then the cluster's diameter is expected to show the highest dissimilarity between two series belonging to the same cluster (as shown in Figure 4.2). Therefore, LTM uses RNOMC in place of the actual correlation as it represents the degree of dissimilarity between a pair of time-series. This approach was also applied by Rodrigues et al. (2008) when performing a hierarchical clustering of time-series data streams.

Thereafter, the last stage of the algorithm is to extract profiles of relationships from the normalised correlation matrix. The methodology used in this

Algorithm 4.1 Extracting profiles of relationship of multiple time-series

Input: X, where $X_1, X_2, ..., X_n$ are observed time-series
Output: profiles of relationships between multiple time-series

calculate the *normalised correlation* coefficient [Equation 4.1] of X
for each time-series $X_1, X_2, ..., X_n$ **do**
  //pre-condition: X_i,X_j do not belong to any cluster
  if (X_i,X_j *are* correlated) AND (X_i, X_j do not belong to any cluster) **then**
    allocate X_i,X_j together in a new cluster
  end if
  //pre-condition: X_i belongs to a cluster; X_j does not belong to any cluster
  if (X_i,X_j *are* correlated) AND (X_i belongs to a cluster) **then**
    if (X_j *is* correlated with all X_i cluster member) **then**
      allocate X_j to cluster of X_i
    else if (X_i,X_j correlation > max(correlation) of X_i with its cluster member) AND (X_j *is not* correlated with any of X_i cluster member) **then**
      remove X_i from its cluster; allocate X_i,X_j together in a new cluster
    end if
  end if
  //pre-condition: X_i and X_j belong to different cluster
  if (X_i,X_j *are* correlated) AND (X_i, X_j belong to different cluster) **then**
    if (X_i *is* correlated with all X_j cluster member) AND (X_j *is* correlated with all X_i cluster member) **then**
      merge cluster of X_i,X_j together
    else if (X_i,X_j correlation > max(correlation) of X_j with its cluster member) AND (X_j *is* correlated with all X_i cluster member) **then**
      remove X_j from its cluster; allocate X_j to cluster of X_i
    else if (X_i,X_j correlation > max(correlation) of both X_i,X_j with their cluster member) AND (X_i *is not* correlated with one of X_j cluster member) AND (X_j *is not* correlated to any of X_i cluster member) **then**
      remove X_i,X_j from their cluster; allocate X_i,X_j together in a new cluster
    end if
  end if
end for
return clusters of multiple time-series

step is outlined in lines 3 to 24 of Algorithm 4.1. The process of extracting profiles of relationships is illustrated in Figure 4.2. Basically, the fundamental concept behind this algorithm is to group multiple time-series with comparable fashion of movement whilst ensuring that all time-series that belong to the same cluster are correlated and hold significant level of similarity.

The underlying concept behind Algorithm 4.1 is closely comparable to the clustering algorithm known as *Clustering Affinity Search Technique* (CAST) (Ben-Dor & Yakhini, 1999). However, Algorithm 4.1 works by dynamically creating new clusters, deleting and merging existing clusters as it evaluates the coefficient of similarity between time-series or observed variables. Therefore, Algorithm 4.1 is considerably different to CAST which creates a single cluster at a time and performs updates by adding new elements to the cluster from a pool of elements, or by removing elements from the cluster and returning it to the pool as it evaluates the affinity factor of the cluster to which the elements belong.

After the profiles have been extracted, the next step is to mine and cluster trends of movement from each profile. This process is explained in the next section. Additionally, as the time-complexity of Algorithm 4.1 is $O(\frac{1}{2}(n^2-n))$, where n is the number of multiple time-series being analysed. In order to avoid expensive re-computation and extraction of profiles, extracted profiles of relationships are stored and updated dynamically instead of being computed on the fly.

4.3.2 Clustering Recurring Trends of a Time-Series

Maintaining profiles of relationships between multiple time-series enables the method to identify the time-series that influences the most the movement of other time-series in a particular time locality. However, this type of knowledge by itself does not offer any predictive power to estimate future values of

multiple time-series.

To predict future values of multiple time-series simultaneously, information about different shapes of movement across a group of correlated multiple time-series needs to be acquired and maintained. Therefore, it is necessary to extract trends of movements from all time-series belonging to a particular profile, and to utilise them in constructing multiple local models that can be used to predict future trends.

4.3.2.1 Trends Extraction with Polynomial Regression

The first approach used in LTM is to model trends of movement of a time-series with polynomial functions calculated by using the polynomial regression analysis (Freund, William, & Sa, 2006; Widiputra, Kho, Lukas, Pears, & Kasabov, 2009). Polynomial regression enables a non-linear trend of movement that emerges in a specific time locality to be captured and modelled appropriately by LTM.

General Principles

The fundamental idea of the technique is to create a number of clusters containing similar recurring trends of movement in different time localities from each series in the multiple time-series data set being observed. A function to estimate the upcoming movement of a group of similar and recurring trends is then obtained by utilising all samples that belong to a particular cluster. Furthermore, as additional knowledge the total number of created clusters would indicate the number of different trends of movement that exist in a particular series.

In the clustering process, trends of movement are captured from a stream of time-series data, and this process starts with an empty set of clusters. When the first trend becomes available, a new cluster is created; the cluster centre is

defined by the coefficients of the polynomial function, and its cluster radius is initially set to zero. When more trends are presented one after another, some of the created clusters will then be updated through changing their centres and increasing their radii. Which cluster will be updated and how much it will be changed depends on the position of the current trend in the input space. A cluster will not be updated any more when its radius reaches a pre-defined limit or threshold.

The utilised clustering process was inspired by the ECM (Q. Song & Kasabov, 2001), see Section 2.5.2.3. However, a crucial modification is made to the distance measure used by ECM. Defining similarity between objects in a clustering process is a very important step. Different definitions of similarity would result in diverse solutions or various groups of samples. In the clustering process, the Euclidean distance is the most widely used method to measure distance between objects (Brummer & Strydom, 1997; Mukherjee, Chen, & Gangopadhyay, 2006; Remy & Thiel, 2005; L. Wang, Zhang, & Feng, 2005).

The Euclidean distance provides an efficient way to calculate distance between two points or objects simply by calculating distances between variables or features used to describe the objects. However, this methodology clusters not objects but sets of polynomial function and we are interested more in calculating similarity between trends rather than the distance between them. Therefore, the use of the Euclidean distance is considered to be improper and a different method of distance measure is used in place of the Euclidean distance. The method is known as the *angular separation* or the Cosine distance (Qian, Sural, Gu, & Pramanik, 2004; Zou & Umugwaneza, 2008).

By using the Cosine distance, similarity between two points or objects is computed in relation to their magnitudes and directions. Therefore, it can be considered as an appropriate approach to calculate level of similarity between two polynomial functions. Let f_1 and f_2 be two polynomial functions of r^{th}

degree described as follows:

$$\begin{aligned} f_1 &= \beta_{10} + \beta_{11}x + \beta_{12}x^2 + ... + \beta_{1r}x^r + \varepsilon, \text{ and} \\ f_2 &= \beta_{20} + \beta_{21}x + \beta_{22}x^2 + ... + \beta_{2r}x^r + \varepsilon \end{aligned} \tag{4.2}$$

Then similarity between these two polynomial functions S_{f_1,f_2} is calculated using the following equation:

$$S_{f_1,f_2} = \frac{\sum_{i=0}^{r}(\beta_{1i}.\beta_{2i})}{\sqrt{\sum_{i=0}^{r}\beta_{1i}^2.\sum_{i=0}^{r}\beta_{2i}^2}}, \tag{4.3}$$

where β_{1i} and β_{2i} are the coefficients of each part of respective polynomial function f_1 and f_2, and $i = 1, \ldots, r$.

Learning Algorithm of Trends Clustering

Based on the fundamental idea and principles outlined in the General Principles section, a detailed algorithm is used to cluster recurring trends of a time-series when polynomial functions are employed to model trends of movement. It is described as follows:

- In the procedure of trends clustering in which the trend of movement is represented by a polynomial function the following indexes are used:
  - training data chunks or snapshots: $i = 1, 2, ...$;
  - number of clusters: $j = 1, 2, ..., m$;
  - input variables: n;
  - order of polynomial function: $s = 1, 2, ..., r$.
- Step 1: perform autocorrelation analysis on the time-series data set from which trends of movement will be extracted and clustered. Number of lag, as outcome of the autocorrelation analysis where $lag > 0$, with

highest correlation coefficient is then taken as the size of data chunk or snapshot window denoted as n. The process will then progress by performing a bootstrap sampling process through all data chunks or snapshots;

- Step 2: create the first cluster C_1 by simply taking the trend of movement of the first data chunk or snapshot $\mathbf{x}_1$, from the input stream as the first cluster centre c_1, and set the cluster radius R_1 to 0.
  Estimate a set of polynomial functions or regression models $\mathbf{y}_i$, starting from order 1 to the highest pre-defined order r, through polynomial regression analysis.
  Using an r^{th} order polynomial regression would then result in an r number of polynomial functions or regression models that may explain the trend of movement as follows,

$$\mathbf{y}_i \begin{cases} f_1(\mathbf{x}_i) = a_{10} + a_{11}\mathbf{x}_i + \varepsilon_1 \\ f_2(\mathbf{x}_i) = a_{20} + a_{21}\mathbf{x}_i + a_{22}\mathbf{x}_i^2 + \varepsilon_2 \\ \quad \vdots \\ f_r(\mathbf{x}_i) = a_{r0} + a_{r1}\mathbf{x}_i + a_{r2}\mathbf{x}_i^2 + \cdots + a_{rr}\mathbf{x}_i^r + \varepsilon_r \end{cases} \tag{4.4}$$

  where $\mathbf{x}_i = (x_1^{(i)}, \ldots, x_n^{(i)})$ and a_{jk} are the coefficients of a polynomial function.
  The best-fit regression model y_i which represents the i-th trend of movement is then selected from the system of polynomials $\mathbf{y}_i$ by finding the function with the minimum value of sum of square residuals defined by:

$$\text{SSR} = \sum_{t=1}^{n} \left(x_t^{(i)} - \hat{f}_s(x_t^{(i)}) \right)^2, \tag{4.5}$$

  where s is the order of the polynomial function or regression model with the minimum sum of square residuals and $\hat{f}_s$ is a particular polynomial function used to compute the estimated trajectories.

The coefficients of each part of the best regression model are then used as features vector describing the trend of movement that exists in that particular data chunk or snapshot as follows:

$$\mathbf{a}_s = (a_{s0}, a_{s1}, a_{s2}, \ldots, a_{sr}), \quad s \leq r$$

where r is the highest degree of the polynomial regression order used to extract trend of movement from a data chunk or snapshot.

To gain knowledge about upcoming trends of movement when a particular trend emerges in a locality of time, the algorithm also models subsequent trajectories from data chunks:

$$\mathbf{y}_i^{(u)} \begin{cases} f_1(\mathbf{x}_i^{(u)}) = a_{10} + a_{11}\mathbf{x}_i^{(u)} + \varepsilon_1 \\ f_2(\mathbf{x}_i^{(u)}) = a_{20} + a_{21}\mathbf{x}_i^{(u)} + a_{22}\mathbf{x}_i^{(u)2} + \varepsilon_2 \\ \quad \vdots \\ f_r(\mathbf{x}_i^{(u)}) = a_{r0} + a_{r1}\mathbf{x}_i^{(u)} + a_{r2}\mathbf{x}_i^{(u)2} + \cdots + a_{rr}\mathbf{x}_i^{(u)r} + \varepsilon_r \end{cases} \tag{4.6}$$

where $\mathbf{x}_i^{(u)} = (x_1^{(i)}, \ldots, x_n^{(i)}, x_{n+1}^{(i)})$.

Utilising the same polynomial regression analysis as explained previously, the best-fit regression model $y_i^{(u)}$ is then selected from $\mathbf{y}_i^{(u)}$ by finding a model with the minimum value of sum of square residuals.

$$\text{SSR} = \sum_{t=1}^{n+1} \left(x_t^{(i)} - \hat{f}_{s'}(x_t^{(i)}) \right)^2, \tag{4.7}$$

where s' is the order of polynomial function or regression model for the upcoming trend with minimum sum of square residual. The coefficients of each part of the best-fit regression model are then used as a feature vector describing the upcoming trend of movement as follows:

$$\mathbf{a}_{s'} = (a_{s'0}, a_{s'1}, a_{s'2}, \ldots, a_{s'r}), \quad s' \leq r$$

- Step 3: if all data chunks or snapshots of the data stream have been processed, the algorithm terminates, else current data chunk or snapshot

$\mathbf{x}_i$, is taken. The trend of movement y_i is then extracted from this data chunk or snapshot using the same process as described in Step 2, and the distances between current trend and all m already created cluster centres are calculated as follows: $D_{i,j} = 1 - \text{CosineDistance}(y_i, c_j), j = 1, 2, ..., m$;

- Step 4: find cluster C_a (with centre c_a and cluster radius R_a) from all m existing cluster centres through calculating the values $S_{i,j} = D_{i,j} + R_j, j = 1, 2, ..., m$, and then choosing the cluster centre c_a with the minimum value $S_{i,a}$:

$$S_{i,a} = D_{i,a} + R_a = \min(S_{i,j}), j = 1, 2, ..., m,$$

where

$$D_{i,a} = 1 - \text{CosineDistance}(y_i, c_a);$$

- Step 5: if $S_{i,a} > 2 \times Dthr$, current trend of $\mathbf{x}_i$, y_i, does not belong to any existing clusters. A new cluster is then created in the same way as described in Step 2 and algorithm returns to Step 3. As in the ECM, $Dthr$ is the clustering parameter that would affect the number of clusters to be created;

- Step 6: if $S_{i,a} \leq 2 \times Dthr$, current trend of $\mathbf{x}_i$, y_i, joins cluster C_a. Cluster C_a is updated by moving its centre c_a, and increasing the value of its radius R_a. The updated radius R_a^{new} is set to be equal to $S_{i,a}/2$ and the new centre c_a^{new} is located at the point on the line connecting current trend y_i, and C_a's mean trends of movement, represented by the cluster centre c_a. Distance from the new centre c_a^{new} to current trend y_i, is equal to R_a^{new}. The algorithm then returns to Step 3.

4.3.2.2 Trends Extraction with Kernel Regression

Utilising the polynomial regression analysis in modelling trends of movement of a time-series has an important consequence; that is the estimated model of a particular trend of movement is highly dependent on a pre-determined type of a polynomial function. Generally speaking, to represent trends of movement from different localities of time, the methodology requires a pre-determined form or function, i.e. r^{th} degree polynomial function, to be provided. This represents a significant limitation of the methodology.

As trends of movements change dynamically in different time localities, the use of a fixed single pre-determined function, e.g. a 3^{rd} degree polynomial function, to model every trend of movement might not be adequate. Using a 3^{rd} degree polynomial function as a pre-determined function offers the flexibility to model trend of movement in the form of a constant, linear function, quadratic function, or 3^{rd} degree polynomial function itself (as in Equation 4.4). However, when more complex trends of movement emerge in different time localities, then the 3^{rd} degree polynomial function might not be sufficient any more. Instead, higher degree polynomial functions might be required to better model these more complex trends of movement.

Therefore, to overcome such limitations this section outlines the use of a different method of regression analysis that does not require a predetermined form or function to be presented in modelling trends of movement.

General Principles

The next version of the algorithm to cluster recurring trends of movements from localised sets of time-series data will use a *non-parametric* regression method in place of the polynomial regression which falls under the category of *parametric* regression analysis. The nature of non-parametric regression analysis which does not take a predetermined form or function that relates

the response to the predictors and does not assume that the observed data set is drawn from a Gaussian data distribution serves as the core motivation for this alteration.

Apart from the replacement of the technique to extract and model trends of movement, the new algorithm employs the same principles as the methodology of clustering recurring trends with polynomial regression, as explained in the previous section. Again, the fundamental idea behind the methodology is to constitute multiple local models that provide knowledge to estimate upcoming trends of movement of a time-series by clustering recurring trends of movement from different localities of time in the past.

Kernel Regression

In parametric regression of the form $y = f(x) + e$, where f is a known smooth function, the appropriate form of f has to be pre-determined prior to the regression process. Conversely, in non-parametric regression, f is some unknown smooth function and is not required to be specified prior to the regression process. Instead, a data-driven technique is put into action in determining the shape of the curve.

Nevertheless, similar to parametric regression, a weighted sum of the y observations is used to obtain the fitted values. However, instead of using equal weights as in the ordinary least squares or weights proportional to the inverse of variance as is often the case in the weighted least squares, a different rationale determines the choice of weights in a non-parametric regression.

The *kernel regression* technique is a non-parametric method in statistics used to find a non-linear relation between a pair of random vectors $\mathbf{X} = (X_1, X_2, ..., X_n)$ and $\mathbf{Y} = (Y_1, Y_2, ..., Y_n)$, where n is the number of input/output variables. The goal of kernel regression is to obtain and use appropriate weight vector $\mathbf{w}$, in finding a regression function $f(\mathbf{x}, \mathbf{w})$, such

that the function is a best-fit match to the data set $(\mathbf{X}, \mathbf{Y})$. Here, $\mathbf{x}$ is the extended smaller value of the original data $\mathbf{X}$ at a certain small step dx calculated using a specific kernel function.

For instance, let a Gaussian membership function be the chosen kernel function; then $\mathbf{x}$ is calculated at domain j where $j = 1, 2, ..., (\frac{n}{dx} + 1)$, in a certain small step dx i.e. $dx = 0.1$, from the original data $\mathbf{X} = (X_1, X_2, ..., X_n)$ by the following equation:

$$\mathbf{x}_j = K(x_j, X_i) = \exp\left(-\frac{(x_j - X_i)^2}{2\alpha^2}\right), \tag{4.8}$$

where $x_j = dx \times (j-1)$, α is a pre-defined kernel bandwidth and $i = 1, 2, ..., n$. To estimate Y_j at domain value x_j, Nadaraya and Watson proposed one of the more commonly used kernel regression formulae (Nadaraya, 1964; Watson, 1969), known as the *Nadaraya-Watson* kernel weighted average (Bierens, 1994; Georgiev, 1988; Shapiai, Ibrahim, Khalid, Jau, & Pavlovich, 2010) defined as follows:

$$\hat{Y}_j = f_j(\mathbf{x}_j, \mathbf{w}) = \frac{\sum_{i=1}^{n} w_i K(x_j, X_i)}{\sum_{i=1}^{n} K(x_j, X_i)}. \tag{4.9}$$

The kernel weights $\mathbf{w} = (w_1, w_2, ..., w_n)$ in Equation 4.9, can then be estimated using a common parameter estimator method, i.e. OLS or WLS, such that the following objective functions is minimised:

$$\text{SSR} = \sum_{i=1}^{n} (Y_i - \hat{Y}_j), \forall\ \hat{Y}_j \text{ where } x_j = X_i. \tag{4.10}$$

In LTM, kernel regression with the Nadaraya-Watson kernel weighted average is utilised to estimate the regression function, in the form of a kernel weight vector that is a best-fit match to the trajectory of a time-series in a particular time locality. Consequently, the calculated kernel weight vector $\mathbf{w}$ as an outcome of the regression process is used as a feature vector to represent the trend of movement.

Learning Algorithm of Trends Clustering

The first step of the learning algorithm is to define the size of data chunk or snapshot window from which the trend of movement will be extracted. This is done by applying autocorrelation analysis to the time-series under examination. The next step is to extract trends of movements by performing a bootstrap sampling process through all available data chunks or snapshots. This process of extracting trends of movement is achieved by utilising the kernel regression method as explained in the previous section of the chapter. Consequently, as an outcome of the kernel regression analysis, the computed kernel weight vectors are then used as feature vectors to represent trends of movements in this methodology.

Thereafter, the algorithm employs a clustering process to group similar and recurring trends of movement. Recurring trends are grouped by a modified version of ECM (Q. Song & Kasabov, 2001) in which the Correlation distance is used in place of the Euclidean distance to measure similarity between a kernel weight vector and a cluster centre. Additionally, in this methodology a cluster centre represents the mean of trends of movement calculated as an average value of all kernel weight vectors which belonging to the same cluster. As new observations become available, new data chunks or snapshots are presented to the system. Accordingly, new clusters containing new trends of movement may be created while some existing clusters are updated. A new cluster is created when the algorithm recognises that a new non-comparable trend of movement has emerged. Consequently, existing clusters are updated when a data chunk or snapshot with recurring trends of movement is identified.

Clusters of trends of movement are then stored in each extracted relationship profile. This information about relationships between series and trends of movements will then be exploited through the use of a *knowledge repository*

to perform simultaneous multiple time-series predictions. The algorithm to cluster recurring trends of a time-series when the Kernel regression is put in place to extract trends of movement is outlined below. The algorithm has the same basic structure as the one in Section 4.3.2.1:

- In the procedure of trends clustering in which trend of movement is represented by a kernel weight vector the following indexes are used:
  - number of data chunks or snapshots: $i = 1, 2, ...;$
  - number of clusters: $l = 1, 2, ..., m;$
  - number of input and output variables: $k = 1, 2, ..., n.$
- Step 1: perform the autocorrelation analysis on the time-series data set from which trends of movement will be extracted and clustered. Number of lags, as an outcome of the autocorrelation analysis where $lag > 0$, with highest correlation coefficient, is then taken as the size of data chunk or snapshot window denoted as n. The process will then progress by performing a bootstrap sampling process through all data chunks or snapshots;
- Step 2: create the first cluster C_1 by simply taking $\mathbf{w}_1$, which is the trend of movement of the first data chunk or snapshot $\mathbf{X}^{(1)} = (X_1^{(1)}, X_2^{(1)}, ..., X_n^{(1)})$, from the input stream as the first cluster centre c_1 and set the cluster radius R_1 to 0.

  In this methodology, the i-th trend of movement represented by the kernel weight vector $\mathbf{w}_i = (w_{i1}, w_{i2}, ..., w_{in})$ is calculated using the Nadaraya-Watson kernel weighted average formula defined as follows:

$$\hat{X}_j^{(i)} = f_j(\mathbf{x}_j^{(i)}, \mathbf{w}_i) = \frac{\sum_{k=1}^{n} w_{ik} x_{jk}}{\sum_{k=1}^{n} x_{jk}}. \tag{4.11}$$

Here $\mathbf{x}_j^{(i)} = (x_{j1}^i, ..., x_{jk}^i)$ is the extended smaller value of the original data $\mathbf{X}^{(i)}$ at domain j and certain small step dx where $j = 1, 2, ..., (\frac{n}{dx} + 1)$. The extended smaller value $\mathbf{x}_j = (x_{j1}, ..., x_{jk})$ is then calculated using the Gaussian MF equation as follows:

$$x_{jk} = K(x_j, k) = \exp\left(-\frac{(x_j - k)^2}{2\alpha^2}\right), \tag{4.12}$$

where $x_j = dx \times (j - 1)$, $k = 1, 2, ..., n$ and α is a pre-defined kernel bandwidth.

The kernel weight $\mathbf{w}_i$ is estimated using OLS such that the following objective functions is minimised:

$$\text{SSR} = \sum_{k=1}^{n}(X_k^{(i)} - \hat{X}_j^{(i)}), \forall\ \hat{X}_j^{(i)} \text{ where } x_j = k. \tag{4.13}$$

To gain knowledge about upcoming trends of movement when a particular trend emerges in a locality of time, the algorithm also models the next trajectory of a data chunk or snapshot by:

$$\hat{X}_j^{(i)(u)} = f_j(\mathbf{x}_j^{(u)}, \mathbf{w}_i^{(u)}) = \frac{\sum_{k=1}^{n+1} w_{ik}^{(u)} K(x_j^{(u)}, k)}{\sum_{k=1}^{n+1} K(x_j^{(u)}, k)}, \tag{4.14}$$

where $x_j^{(u)} = dx \times (j^{(u)} - 1); j^{(u)} = 1, 2, ..., (\frac{n+1}{dx} + 1); k = 1, 2, \ldots, n + 1$ and the kernel weights $\mathbf{w}_i^{(u)} = (w_{i1}^{(u)}, w_{i2}^{(u)}, ..., w_{i(n+1)}^{(u)})$.

- Step 3: if there are no more data chunks or snapshots, the algorithm terminates; else the next data chunk or snapshot, $\mathbf{X}^{(i)}$, is taken. The trend of movement from $\mathbf{X}^{(i)}$ is then extracted as in Step 2, and distances between current trend and all m existing cluster centres are calculated by

$$D_{i,l} = 1 - \text{CorrelationCoefficient}(\mathbf{w}_i, c_l), \tag{4.15}$$

where $l = 1, 2, ..., m$. If, for an existing cluster C_l, $D_{i,l} \leq R_l$, then the current trend joins cluster C_l and the step is repeated; else continue to next step;

- Step 4: find a cluster C_a (with centre c_a and cluster radius R_a) from all m existing cluster centres by calculating the values of $S_{i,a}$ given by

$$S_{i,a} = D_{i,a} + R_a = \min(S_{i,l}), \tag{4.16}$$

where $S_{i,l} = D_{i,l} + R_l$ and $l = 1, 2, ..., m$.

- Step 5: if $S_{i,a} > 2 \times Dthr$, where $Dthr$ is a threshold that determines the maximum size of a cluster radius, then current trend of $\mathbf{X}^{(i)}$, $\mathbf{w}_i$, does not belong to any existing clusters. A new cluster is then created in the same way as described in Step 2, and the algorithm returns to Step 3;

- Step 6: if $S_{i,a} \leq 2 \times Dthr$, current trend $\mathbf{w}_i$ joins cluster C_a. Cluster C_a is updated by moving its centre, c_a, and increasing the value of its radius, R_a. The updated radius R_a^{new} is set to $S_{i,a}/2$ and the new centre c_a^{new} is now the mean value of all trends of movement that belong to cluster C_a. Distance from the new centre c_a^{new} to current trend $\mathbf{w}_i$, is equal to R_a^{new}. The algorithm then returns to Step 3.

4.3.3 LTM for Multiple Time-Series Modelling and Prediction

Figure 4.3 illustrates how a repository containing profiles of relationships and recurring trends (the knowledge repository) is built and maintained. Using data from the first data chunk or snapshot, the algorithm extracts two profiles of relationship in the multiple time-series by creating two clusters. The first cluster represents a profile whereby time-series #1 and time-series #3

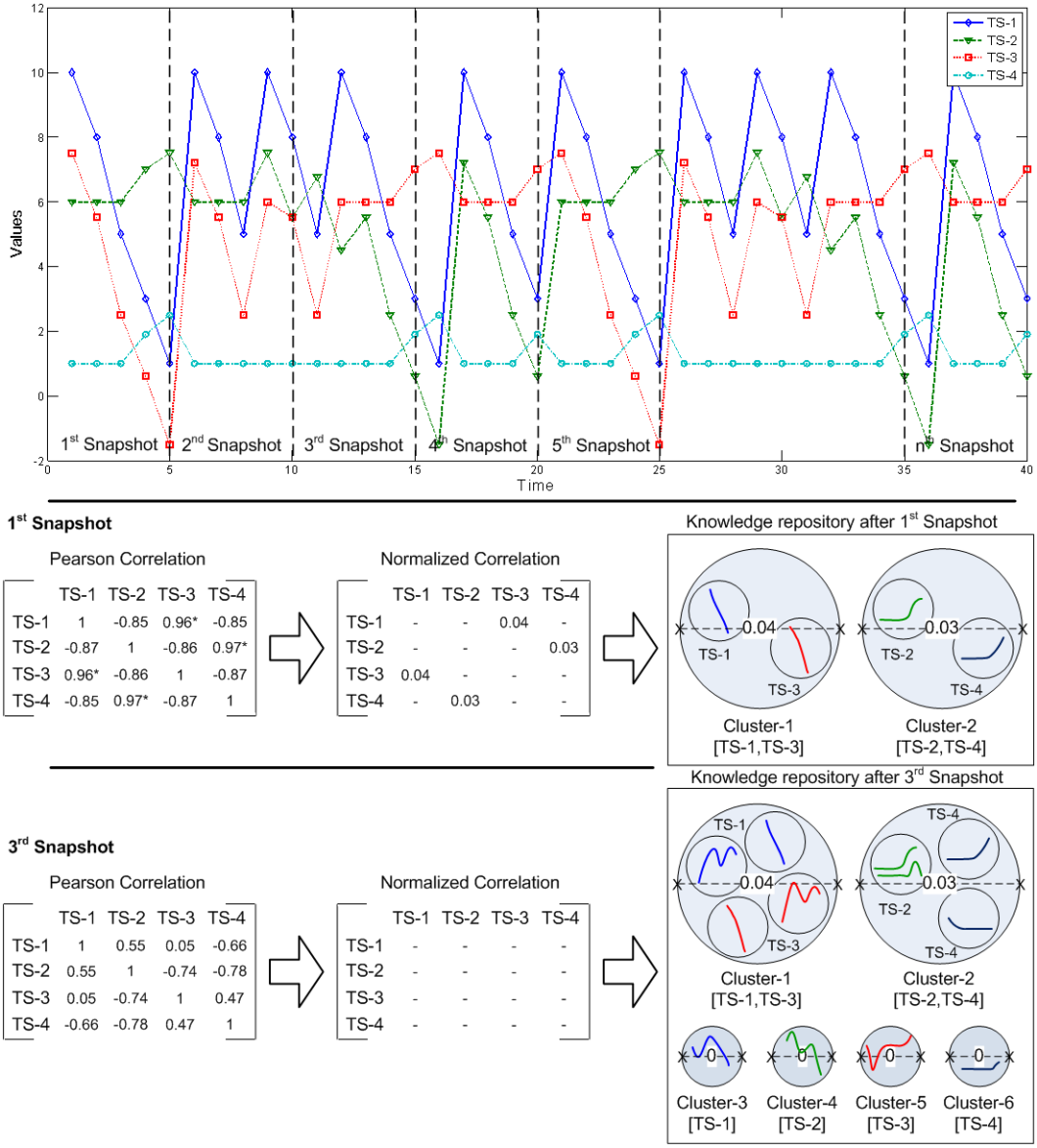


Figure 4.3. Creation of knowledge repository (profiles of relationships and recurring trends).

are correlated and moving together, while the second cluster is a profile of relationship whereby time-series #2 and time-series #4 are progressing in a similar fashion.

Trends of movement of each time-series that belongs to a particular profile is then extracted and kept within the profile. As illustrated in Figure 4.3,

after extracting the trend of movement from each time-series in the first profile denoted by Cluster-1[TS-1,TS-3], the algorithm creates and stores two other trends in Cluster-1[TS-1,TS-3], denoted by TS-1 and TS-3. Here TS-1 and TS-3 represent trends of movement of time-series #1 and #3 when they are correlated. The same process is then applied to the second profile of time-series #2 and time-series #4 denoted by Cluster-2[TS-2,TS-4].

As the second data chunk becomes available, the algorithm applies the same procedure to extract profiles of relationships. As it retains the same profiles from the second data chunk or snapshot which are [TS-1,TS-3] and [TS-2,TS-4], the algorithm does not create any new cluster in the knowledge repository. However, as it extracts trends of movement from each time-series, the algorithm finds that the second data chunk holds a different type of behaviour compared to the first data chunk.

Consequently, the algorithm updates the information about trends of movement of each time-series in all existing profiles. New clusters of trends are then created and stored in Cluster-1[TS-1,TS-3] as well as in Cluster-2[TS-2,TS-4] to represent the new behaviour exhibited by the second data chunk or snapshot. For Cluster-1[TS-1,TS-3], two instances representing trends are created to represent a new form of relationship between the pair of time-series #1 and #3 that differs from the one which existed in the first data chunk, whereas for Cluster-2[TS-2,TS-4] only a single new instance is created. This is because the trend of movement for time-series #2 in the second data chunk is comparable to the existing instance and therefore it joins the cluster on its own.

Additionally, as the algorithm processes the third data chunk or snapshot, it realises that within this locality of time the four series are uncorrelated and moving individually. As a result, new profiles are created represented by four new clusters: Cluster-3[TS-1],Cluster-4[TS-2],Cluster-5[TS-3], and Cluster-6[TS-4] each denoting specific trends of movement. The procedure continues until there is no more data chunks to be processed.

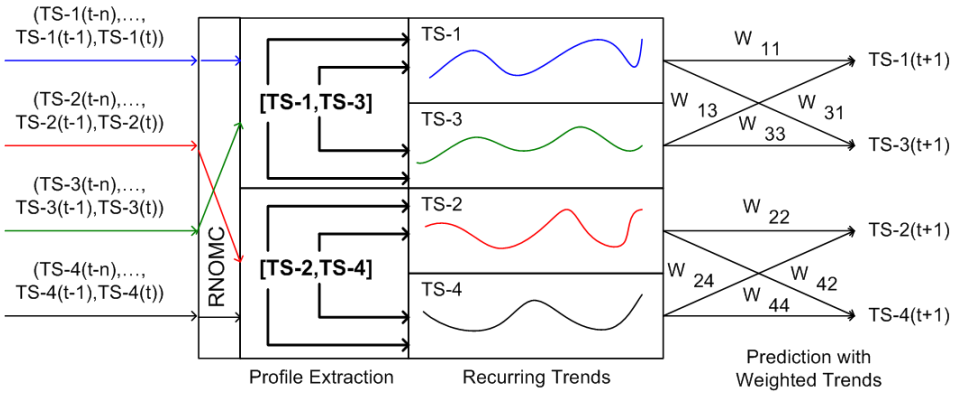


Figure 4.4. Multiple time-series prediction using profiles of relationships and recurring trends.

The process of constructing the knowledge repository can be considered a form of spatio-temporal modelling, whereby different shapes of trends (spatio) are extracted continuously over time (temporal). The repository illustrates how relationships between observed time-series or variables change dynamically over different time localities, retaining different shapes of movement (trends).

After the repository has been built, there are two further steps that need to be performed before prediction can take place. The first is to extract current profiles of relationships between the multiple series. Thereafter, matches are found between the current trajectory and previously stored profiles from the past. Predictions are then made by implementing a weighting scheme that gives more importance to pairs of series that belong to the same profile and retain comparable trends of movement. The weight $w_{i,j}$ for given pair i, j of a series is given by the distance of similarity between them.

The prediction process is illustrated in Figure 4.4, whilst the procedure of predicting movements of multiple time-series simultaneously using the knowledge repository is outlined as follows:

- Step 1: after the knowledge repository **KR** has been initialised using

the training data set and as new data $\mathbf{x}_t = (x_t^{(1)}, x_t^{(2)}, ..., x_t^{(i)})$ becomes available, a new data set $\mathbf{X}_t^{'}$ is constructed:

$$\mathbf{X}_t^{'} = \begin{bmatrix} x_{t-(n-1)}^{(1)} & x_{t-(n-2)}^{(1)} & \cdots & x_t^{(1)} \\ x_{t-(n-1)}^{(2)} & x_{t-(n-2)}^{(2)} & \cdots & x_t^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_{t-(n-1)}^{(i)} & x_{t-(n-2)}^{(i)} & \cdots & x_t^{(i)} \end{bmatrix}$$

where n is the size of data chunk, t is the current time step and i is an index that varies over the time-series;

- Step 2: extract profiles of relationship $\mathbf{p}_t = (p_t^1, p_t^2, ..., p_t^k)$ where $1 \leq k \leq i$, from current data set using $\mathbf{X}_t^{'}$ as described in previous section;

- Step 3: find profiles $\mathbf{p}_{kr} = (p_{kr}^1, p_{kr}^2, ..., p_{kr}^k)$ from previously constructed knowledge repository $\mathbf{KR}$ where $\mathbf{p}_{kr} = \mathbf{p}_t$;

- Step 4: for each series $x^{(i)} \in p_t^k$ extract its current trend of movement $\mathbf{w}_t^{(i)}$ as described in previous section;

- Step 5: for each series $x^{(i)} \in p_t^k$ find j cluster centres of recurring trends in p_{kr}^k, where $j = 1, 2, ..., m$ is the number of series belonging to profile p_{kr}^k by calculating minimum distances between $\mathbf{w}_t^{(i)}$ to all existing cluster centres of recurring trends in p_{kr}^k as follows:

$$D_{i,j} = 1 - \max(\text{CorrelationCoefficient}(\mathbf{w}_t^{(i)}, c_j^l)), \tag{4.17}$$

where $l = 1, 2, ...$ is the number of clusters of recurring trends of series j in p_{kr}^k;

- Step 6: calculate next value of $x^{(i)}$ using the j found cluster centres of recurring trends, by giving more weight w to cluster centres that are closer to $\mathbf{w}_t^{(i)}$. *Note: in this methodology cluster centres c_j of recurring trends represent trends of movement of time-series in a particular profile.*

The weight $w_{i,j}$ that is assigned to cluster centre j when predicting the next value of $x^{(i)}$ is calculated as follows:

$$w_{i,j} = \frac{\max(\mathbf{D}) - (D_{i,j} - \min(\mathbf{D}))}{\max(\mathbf{D})}, \tag{4.18}$$

where $\max(\mathbf{D})$ and $\min(\mathbf{D})$ are the maximum and minimum values of distance vector $\mathbf{D} = (D_{i,1}, D_{i,2}, ..., D_{i,j})$.

In addition, next value of $x^{(i)}$ is given by:

$$x_{t+1}^{(i)} = \frac{\sum_{m=1}^{j} w_{i,m} c_m}{\sum_{m=1}^{j} w_{i,m}}. \tag{4.19}$$

4.4 Experiments on Synthetic Data

In this section of the chapter, to evaluate LTM's capability to analyse, model and predict multiple time-series, a pre-generated synthetic multiple time-series data set that has been introduced and used in Chapter 3 is once again used. The synthetic data set (introduced in Chapter 3) was generated in a way that the series are correlating with each other; in which some of the series are moving towards the same direction in similar fashion, whilst some other series are progressing in the opposite way. Since the principal form of correlation between series in the synthetic data set is known priorly, experimenting with synthetic data should allow us to evaluate whether LTM is capable of extracting the underlying behavior of multiple time-series data.

As in Chapter 3, the experiment in this chapter is performed by splitting the data set into two different parts, a training segment and a testing segment. The training segment is put in place to evaluate LTM's capability of extracting profiles of relationships from multiple time-series data. Additionally, the test data set is utilised to evaluate LTM's capability to incrementally learn and revise its knowledge repository when new patterns of relationships are

emerging while maintaining information about past profiles of relationships that are relatively stable over time. Here, the first 35 time-points of the complete data set is for training, whilst the remaining 15 time-points are used for incremental learning and evaluation.

Both LTM with polynomial regression (LTM-PR) and LTM with Kernel regression (LTM-KR) employ the same methodology to extract profiles of relationships from a set of multiple time-series data. Therefore, identical profiles of relationships will be obtained when the two methodologies are used for multiple time-series analysis and modelling. However, LTM-PR utilises a parametric regression analysis to mine and model trends of movement while LTM-KR utilises the non-parametric one. As a result, even though both methods will obtain identical profiles of relationships when applied to the same set of multiple time-series data, the clusters of trends of movement that are created will be different. Since prediction is directly dependent on the composition of the clusters, it is thus necessary to assess the impact of the different clustering models produced by the two different regression methods on the accuracy of prediction.

Figure 4.5 depicts profiles of relationships extracted from the first 35 time moments of the synthetic multiple time-series data set by both LTM-PR and LTM-KR. The circles in Figure 4.5 represent profiles of relationships captured from the multiple time-series under observation, where cluster radius represents the level of dissimilarity between time-series in the same cluster or profile while relative positioning of the labels indicates the degree of similarity in behaviour. For instance, in the circle plotted at coordinate $(16, 19)$ Series #3 is positioned closer to Series #4 than to Series #2. This indicates that Series #3 is more similar to Series #4 and therefore they have higher degree of correlation compared to that of Series #2 and Series #3 or Series #2 and Series #4.

The colour of a circle represents number of occurrences of a particular

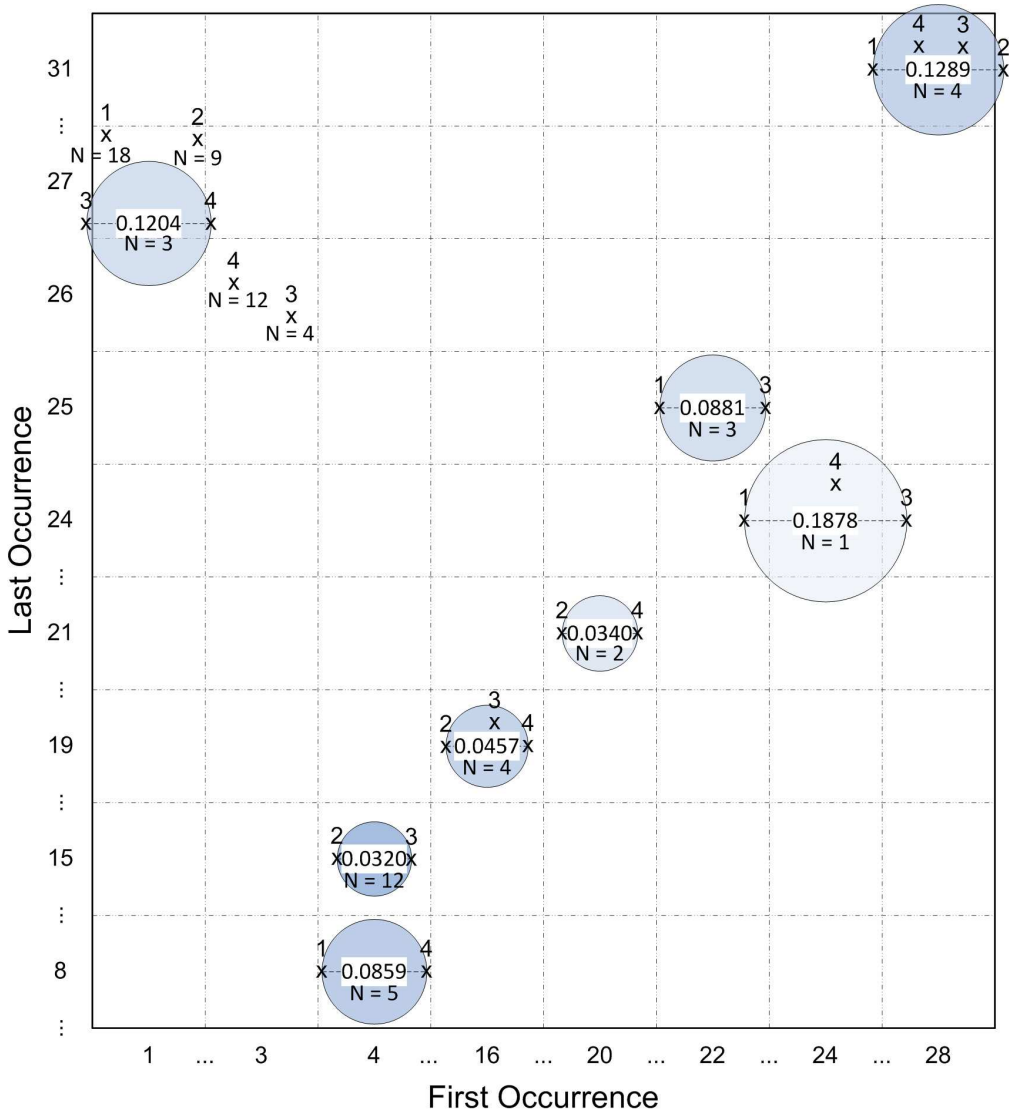


Figure 4.5. Profiles of relationships of the first 35 time-points from a synthetic data set. The numbers #1, #2, #3 and #4 represent Series #1, Series #2, Series #3 and Series #4 respectively.

profile. Darker colour indicates that a significant number of occurrences has been detected. Additionally, the exact number of occurrences of a particular profile is also given by N. The x-axes represents the initial time moment when a particular profile emerged, whilst the y-axes represents the last time moment when a particular profile was recognised and captured.

Table 4.1

List of extracted profiles of relationships from the training set of synthetic data.

No	Profiles	Occurrence	Time Locality
1	[1]	18	1, 2, 3, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 26, 27
2	[2]	9	1, 2, 3, 22, 23, 24, 25, 26, 27
3	[3]	4	3, 20, 21, 26
4	[4]	12	3, 9, 10, 11, 12, 13, 14, 15, 22, 23, 25, 26
5	[1; 3]	3	22, 23, 25
6	[1; 4]	5	4, 5, 6, 7, 8
7	[2; 3]	12	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
8	[2; 4]	2	20, 21
9	[3; 4]	3	1, 2, 27
10	[1; 3; 4]	1	24
11	[2; 3; 4]	4	16, 17, 18, 19
12	[1; 2; 3; 4]	4	28, 29, 30, 31

Figure 4.5 reveals that Series #2 and #3 are closely related and that they are moving together continuously in a significant interval of time moments. This information is discovered by analysing the circle plotted at coordinate $(4, 15)$. The circle plotted at $(4, 15)$ represents profile of relationships between Series #2 and Series #3. The small size of the cluster radius signifies that the correlation level between Series #2 and #3 is very high. Furthermore, being plotted at coordinate $(4, 15)$ the circle discloses that this particular profile started to emerge at time-point 4 and was sustained for the next 11 steps up to time point 15. This piece of information indicates that in the period under evaluation, Series #2 and Series #3 are frequently and continuously moving

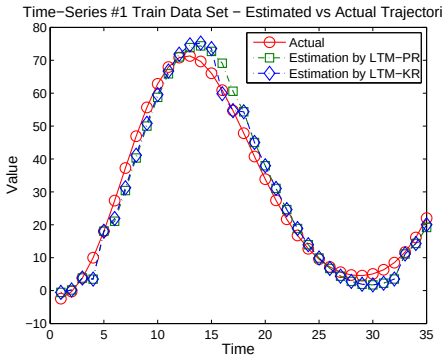
in a similar fashion.

This discovery is in agreement with how Series #2 and #3 were constructed. As outlined in Section 3.4), Series #2 and #3 were constructed from the same polynomial functions. Therefore, it can be expected that the two series have a significant level of correlation. Furthermore, this result is also in agreement with outcomes from the DIN model (see Section 3.4), which discover the existence of positive interaction between Series #2 and #3.

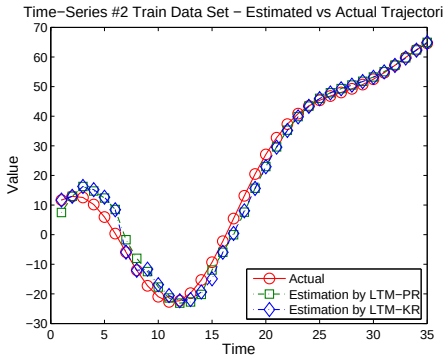
Even though Figure 4.5 shows that there is a significant number of episodes when Series #1 and #4 are moving individually, Figure 4.5 also indicates that there exists a period when Series #1 and #4 are closely related and moving together in a similar manner, which is from time-points 4 to 8. Another discovery from Figure 4.5 is that in general it can be observed that Series #1 and #4 tend to move mutually with the other series during the period of evaluation. Yet again, this finding is in agreement with outcomes from the DIN model which suggests that there is a complex form of interdependencies between the four series.

Based on the analysis of Figure 4.5 outlined above, some conclusions can be made as follows:

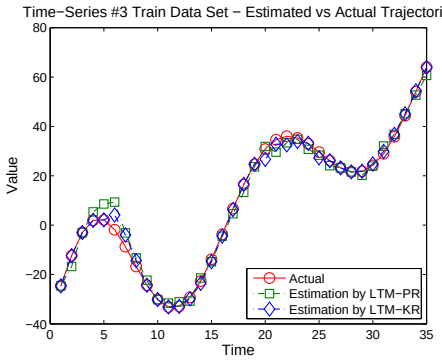
- Series #2 has a stronger relationship with Series #3 compared to the other series and is more likely to move in a similar fashion;
- There exists a period at the beginning of the observation period when Series #1 is moving in a similar course with Series #4. However, there is also a significant number of episodes (towards the end of the observation period) when the two series are tend to progress individually;
- The relationships between Series #2, Series #3 and Series #4 are more significant when compared to the relationship between them and Series #1 throughout the observation period.



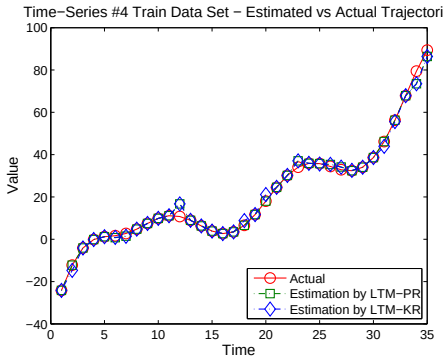
(a) Actual versus estimated trajectories of Series #1.



(b) Actual versus estimated trajectories of Series #2.



(c) Actual versus estimated trajectories of Series #3.



(d) Actual versus estimated trajectories of Series #4.

Figure 4.6. Plot of estimated trajectories of synthetic data set by LTM in the training period.

These inferences are in agreement both with the trajectories of the four series (as illustrated in Figure 3.7) and the extracted interaction networks given by DIN (see Section 3.3). These results suggest LTM's capability to dynamically capture, model and maintain profiles of relationships of multiple time-series over time.

To evaluate LTM's capability of predicting movement of multiple time-series simultaneously, we exploited the constructed knowledge repository to estimate movement of the four series in the training period. Figure 4.6 illus-

Table 4.2

RMSE prediction rates for LTM on the synthetic data set.

No	Variable	Train		Test	
		KR	PR	KR	PR
1	Series #1	3.6226	4.0924	7.8696	8.3671
2	Series #2	3.3031	3.4024	4.1942	4.3328
3	Series #3	1.7731	3.1322	4.8642	5.0498
4	Series #4	1.8580	1.6420	5.4827	5.5959

trates the estimated trajectories calculated by both LTM-PR and LTM-KR against the actual trajectories. It is clearly seen that the estimated trajectories produced by both LTM-PR and LTM-KR match closely with the actual trajectories. This outcome confirms the capability of both LTM-PR and LTM-KR to predict movements of multiple time-series simultaneously with a reasonable degree of accuracy. Yet, analysis of RMSE as outlined in Table 4.2 reveals that LTM-KR performs better compared to LTM-PR when estimating trajectories of the synthetic multiple time-series in the training period.

This outcome indicates that the use of Kernel regression helps to model trends of movement of a time-series more accurately which will produce better prediction compared to the use of polynomial regression. To verify this statement, another evaluation is conducted by presenting the test data to both LTM-PR and LTM-KR.

The first analysis that needs to be made is to find out how the knowledge repository has changed in relation to current behaviour of the four series. Figure 4.7 depicts the state of profiles of relationships after the test data is conferred to both LTM-PR and LTM-KR. Yet again, as both LTM-PR and LTM-KR utilise the same methodology to extract profiles of relationships

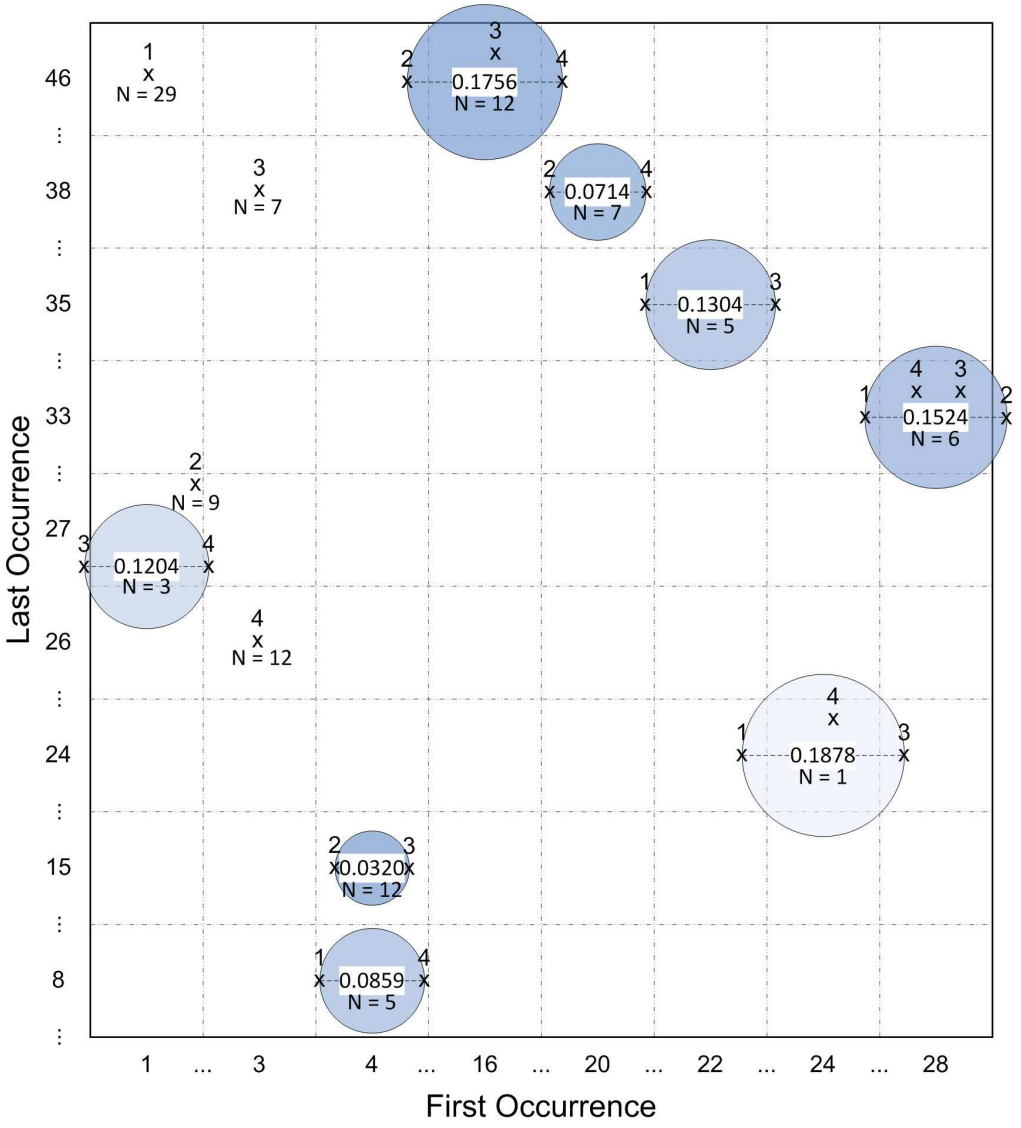


Figure 4.7. Profiles of relationships after 15 points of test data set is conferred to LTM. The number #1, #2, #3 and #4 represent Series #1, Series #2, Series #3 and Series #4 respectively.

from a set of multiple time-series data, an identical structure, as shown in Figure 4.7, is acquired by both models.

The most significant dynamic in the test data that can be easily spotted is how Series #2, #3 and #4 are now frequently moving together. This is encapsulated by the circle plotted at coordinate $(16, 46)$. Previously in

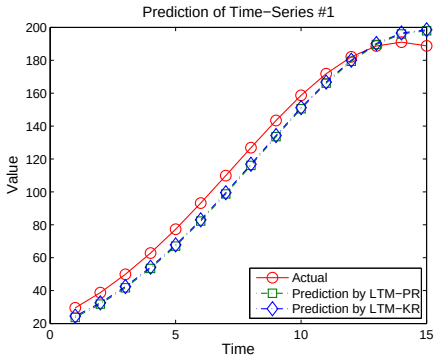
Table 4.3

List of extracted profiles of relationships from the complete set of synthetic data.

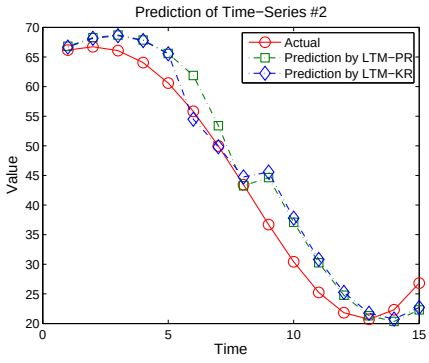
No	Profiles	Occurrence	Time Locality
1	[1]	29	1, 2, 3, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 26, 27, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46
2	[2]	9	1, 2, 3, 22, 23, 24, 25, 26, 27
3	[3]	7	3, 20, 21, 26, 36, 37, 38
4	[4]	12	3, 9, 10, 11, 12, 13, 14, 15, 22, 23, 25, 26
5	[1; 3]	5	22, 23, 25, 34, 35
6	[1; 4]	5	4, 5, 6, 7, 8
7	[2; 3]	12	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
8	[2; 4]	7	20, 21, 34, 35, 36, 37, 38
9	[3; 4]	3	1, 2, 27
10	[1; 3; 4]	1	24
11	[2; 3; 4]	12	16, 17, 18, 19, 39,40, 41, 42, 43, 44, 45, 46
12	[1; 2; 3; 4]	6	28, 29, 30, 31, 32, 33

Figure 4.5 this circle showed an insignificant number of only 4 episodes of occurrence. However, Figure 4.7 shows a dramatic change in the collective movement of these series. The circle indicates that after time-point 35 these three series are moving closely together towards the end of the observation period with a significant number of 12 episodes. Another noteworthy piece of information that can be learnt is that Series #1 retains its individuality by consistently moving alone towards the end of the observation period.

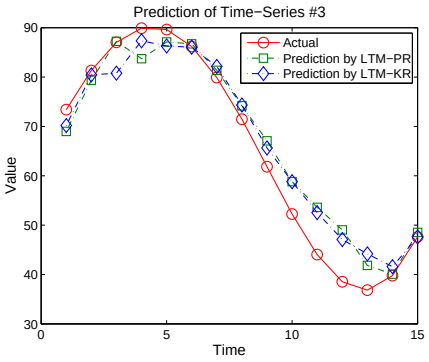
Based on these outcomes, a conclusion can be made regarding the testing period is that Series #2, #3 and #4 are strengthening their relationships and



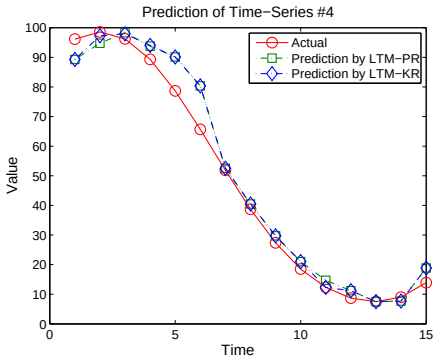
(a) Actual versus estimated trajectories of Series #1.



(b) Actual versus estimated trajectories of Series #2.



(c) Actual versus estimated trajectories of Series #3.



(d) Actual versus estimated trajectories of Series #4.

Figure 4.8. Plot of estimated trajectories of synthetic data set by LTM in the testing period.

are moving collectively in a similar fashion, whereas Series #1 is moving by itself in different direction and is uncorrelated with the other series.

These discoveries are again in agreement with the behaviour of the four series during the testing period, as illustrated in Figure 3.7 and with results from the DIN model. From Figure 3.7, it can be easily recognised that in the testing period the trajectory of Series #1 is moving in a different fashion and in different directions compared to the other three series. Additionally, this result also confirms LTM's capability to dynamically learn new profiles

Table 4.4

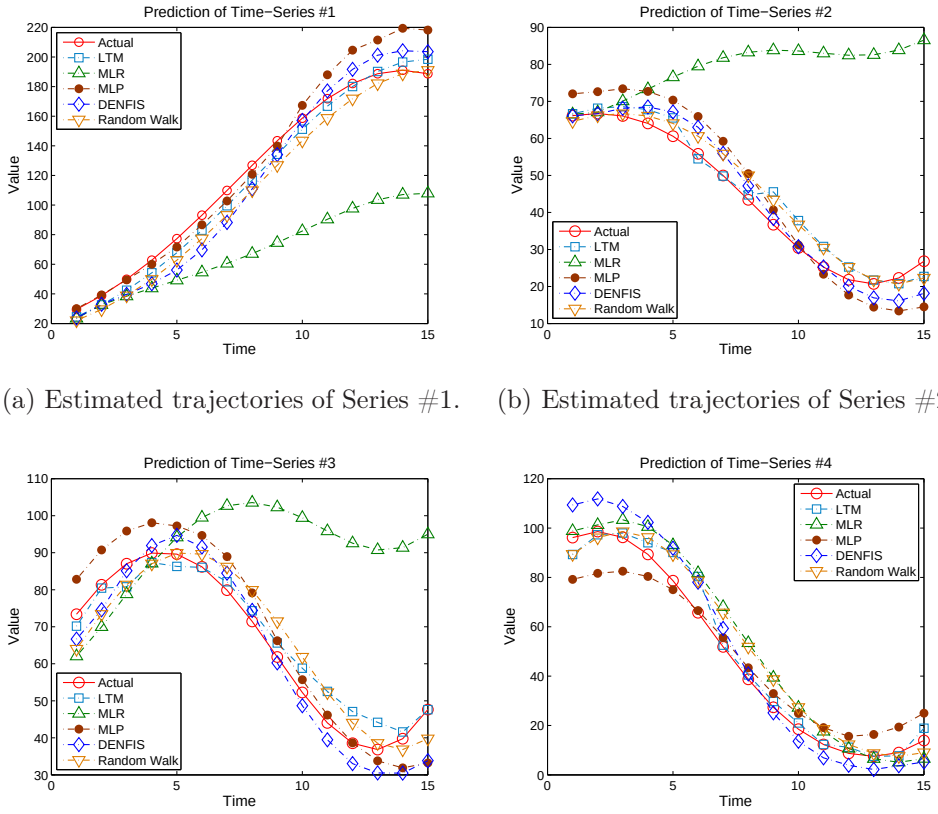
Comparison of LTM prediction error rates against methods applied on single time-series: MLR, MLP, DENFIS and random walk model, in RMSE.

No	Variable	LTM	MLR	MLP	DENFIS	Random Walk
1	Series #1	7.8696	62.0580	15.1441	14.3226	12.6634
2	Series #2	4.1942	43.7277	7.5743	4.6464	4.3419
3	Series #3	4.8642	37.0639	7.6572	6.1003	6.4474
4	Series #4	5.4827	10.1890	8.7729	8.8631	8.4156

of relationships as new observations become available to the system.

Figure 4.8 depicts the performance of both LTM-PR and LTM-KR when predicting movement for the test data set. The figures clearly show that predicted trajectories calculated by both LTM-PR and LTM-KR yet again match closely the actual trajectories as in the training phase. Furthermore, calculated error rates in RMSE as outlined in Table 4.2 reveal that LTM-KR slightly outperforms LTM-PR. This discovery supports our previous assumption that states that the use of Kernel regression helps to model trends of movement of a time-series more accurately and therefore results in a better prediction accuracy.

As a second task of the experiment, a comparative analysis between LTM (the one that utilises the Kernel regression to model trends of movements) and some of the well established methods applied on single time-series prediction, i.e. MLR, MLP, DENFIS and random walk model, is conducted. Throughout the experiments the LTM was trained with the training data set and then utilised the test data set for adaptive training, whilst MLR, MLP and DENFIS were applied as batch models that were trained with the training data set and then tested on the test data set with no incremental learning. Additionally,



(a) Estimated trajectories of Series #1. (b) Estimated trajectories of Series #2.

(c) Estimated trajectories of Series #3. (d) Estimated trajectories of Series #4.

Figure 4.9. Plot of estimated trajectories of synthetic data set by LTM versus prediction by other methods applied on single time-series prediction in the testing period.

random walk model was applied as an incremental model due to its nature.

Figure 4.9 depicts the comparison of the actual trajectories compared to the predicted trajectories calculated by LTM, MLR, MLP, DENFIS and random walk prediction methods. Plotted trajectories indicate that predictions by LTM match closely with the actual trajectory when compared to the other methods applied on single time-series significantly. Consequently, calculated RMSE as outlined in Table 4.4 confirms that generally LTM is superior compared to methods applied on single time-series. This result indicates that predicting movement of multiple time-series simultaneously by utilising knowl-

edge about profiles of relationships and recurring trends leads to much better results rather than predicting movement of single time-series individually.

4.5 Conclusion

The chapter presented a methodology for constructing local models of multiple time-series which captured recurring behaviour in the data. The recurring specific behaviour in this methodology is described as recurring relationships between pairs of time-series that influence each other, and recurring trends of movement within the series. The localised trend model of multiple time-series denoted as LTM, reveals that time-varying profiles of significant and strong relationships along with the recurring trends of movement in multiple time-series can be captured, modelled and maintained. Additionally, the experimentation on synthetic data set undoubtedly prove that the LTM demonstrates the ability to:

1. Extract profiles of relationships and recurring trends from multiple time-series data;

2. Perform simultaneous prediction of multiple time-series with excellent precision;

3. Evolve, by continuing to extract profiles of relationships and recurring trends over time when new data samples become available.

A further direction related to the refinement of this methodology is to explore the use of correlation analysis methods that are capable of detecting non-linear correlations between observed variables (i.e. correlation ratio, Copula, etc.) in place of the Pearson's correlation coefficient when extracting profiles of relationships in multiple time-series data.

Chapter 5

Multivariate Transductive Neuro-Fuzzy Inference System for Multiple Time-Series Analysis and Modelling

5.1 Introduction

In contrast to learning methods that construct a general, explicit description of the target function when training examples are provided, transductive reasoning methods simply store the training examples. Generalising beyond these examples is postponed until a new instance must be classified. A key advantage of this type of learning method is that instead of estimating the target function once for the entire instance space, this method is capable of constructing local and specific estimation models for each new instance that needs to be classified or predicted (Kasabov, 2007b; Kasabov & Pang, 2003; Mitchell, 1997; Q. Song & Kasabov, 2004, 2005, 2006; Q. Song, Ma, & Kasabov, 2006; Vapnik, 1998). The kNN (Soucy & Mineau, 2001; Yamada et al., 2006) and WKNN (Dudani, 1976; Tan, 2005) algorithms, which fall under the category of *instance-based* learning, are well-known realisations of transductive reasoning.

This type of learning offers the following benefits over global and local

models (Hwang, 2009):

1. In a real world problem where the amount of data increases on an ongoing basis, instance-based learning will utilise the latest data but only the part of the data that is relevant to the new input vector is used to build a model or make the decision, and

2. since only a relevant subset of the input vectors in the sample data set is used to derive the solution, it may reduce the effect of outliers, or incorrect identification of sub-problems.

The limitation of instance-based learning is in its reliance on a good definition of problem space utilised to build the solution. A good definition of the problem space is important to any type of reasoning. However, it may be more so for instance-based learning through transductive reasoning. It is because the definition of problem space affects the performance of the similarity function used to identify the neighbourhood, i.e. a subset of input vectors in the training data that are relevant to the new test input vector (Mitchell, 1997).

Despite its limitations, instance-based learning has been widely used to solve classification problems such as text classification (Joachims, 1999), heart disease diagnostics (Wu, Bennett, Cristianini, & Shawe-Taylor, 1999), synthetic data classification using graph-based approach (C. Li & Yuen, 2001), digit and speech recognition (Joachims, 2003), promoter recognition in bioinformatics (Kasabov & Pang, 2003), image recognition (J. Li & Chua, 2003) and image classification (Proedrou, Nouretdinov, Vovk, & Gammerman, 2002), micro-array gene expression classification (West et al., 2001) and biometric tasks such as face surveillance (F. Li & Wechsler, 2004).

Furthermore, this reasoning method is also used in prediction tasks such as finding if a given drug binds to a target site (Weston et al., 2003), evaluating prediction reliability in regression (Bosnic et al., 2003) and providing

additional measures to determine reliability of predictions made in medical diagnosis (Kukar, 2003). However, the use of this learning method for time-series analysis and prediction, in particular multiple time-series, has not been widely studied except for the preliminary study by Widiputra in 2008 which investigated the possibility of using the WKNN in predicting movement of multiple stock market indexes (Widiputra, Pears, & Kasabov, 2009).

As multiple data streams consist of various variables producing examples continuously over time, the basic idea behind the methodology proposed in this chapter of the thesis is simply to find and model relationships between these streams of data at a particular time moment and then to search for similar patterns of relationship from the past. These patterns will then be utilised to constitute a specific model (i.e. weighted localised linear regression, localised fuzzy rules, etc.) to predict future values of multiple time-series simultaneously.

The methodology of transductive reasoning for multiple time-series analysis proposed in this chapter is named the *Multivariate Transductive Neuro-Fuzzy Inference System* (mTNFI). The mTNFI, introduced and explained in this section is an extension of the Neuro-Fuzzy Inference method for transductive reasoning denoted as NFI (Q. Song & Kasabov, 2005), in which modifications were made so that the new methodology is capable of performing multiple time-series data analysis and modelling.

In addition, the proposed methodology addresses some research questions as follows: (1) As transductive approaches develop an individual model over the new input vector instantaneously, would they provide a specific and better local generalisation compared to methods of inductive inference? (2) Can the state of relationships between multiple variables at a particular time-point be utilised to identify a number of samples from the complete training set that are most relevant in constructing specific and better local model of current input vector? (3) Does this specific individual local model offer better prediction

accuracy compared to methods of inductive inference, i.e. MLR, MLP, etc.?

5.2 NFI Model for Transductive Reasoning

Since the proposed transductive approach of multiple time-series analysis and modelling outlined in this chapter is an extension of previously developed model of transductive neuro-fuzzy inference system named the NFI (Q. Song & Kasabov, 2005), then for the sake of completeness this section of the chapter briefly discussed the general principles behind the NFI and some learning rules that were applied on it.

Transductive inference is concerned with the estimation of a function at a single point of the space only. For every new input vector $\mathbf{x}_i$, that needs to be processed for a prognostic task, the nearest neighbours N_i, which form a sub-data set D_i, are derived from an existing data set D and, if necessary, generated from an existing model M. A new model M_i is dynamically created from these samples to approximate the function at point x_i, see Figures 5.1 and 5.2. The system is then used to calculate the output value y_i, for the input vector $\mathbf{x}_i$ as in Figures 5.1 and 5.2. The simplest transductive reasoning method, found yet to be useful, is the kNN algorithm that was outlined and explained in Chapter 2.

General Principles

The NFI for transductive reasoning is a dynamic neural-fuzzy inference system with local generalisation proposed by Q. Song and Kasabov (2005) in which, either *Zadeh-Mamdani* (Zadeh, 1965, 1973), or *Takagi-Sugeno* (Takagi & Sugeno, 1985) type fuzzy inference is used. The local generalisation means that in a sub-space of the whole problem space (local area) a model is created from N_i training samples that are closest to the input vector $\mathbf{x}_i$, which

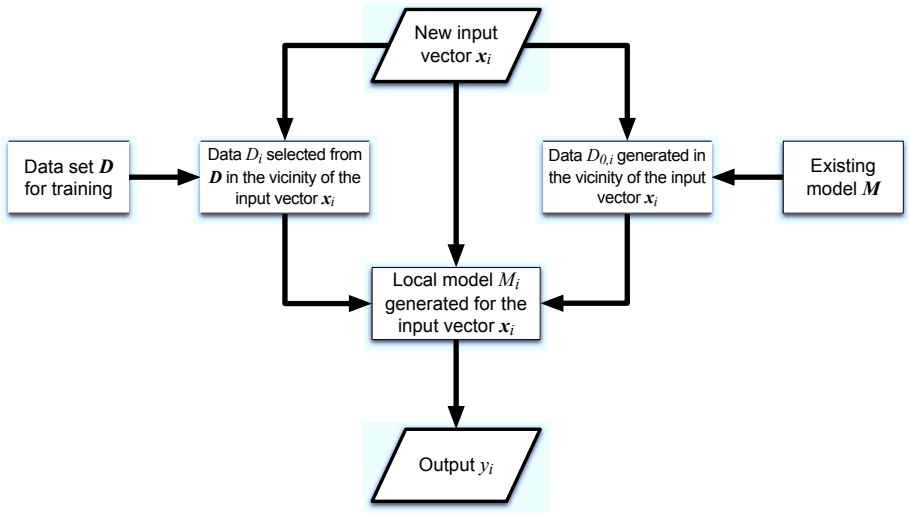


Figure 5.1. Block diagram of a *transductive* reasoning system. An individual model M_i is trained for every new input vector $\mathbf{x}_i$ with the use of samples D_i selected from a data set D, and samples $D_{0,i}$ generated from an existing model (formula) M (*if such a model exists*). The data samples in both D_i and $D_{0,i}$ are similar to the new vector $\mathbf{x}_i$ according to defined similarity criteria (Q. Song & Kasabov, 2005).

performs generalisation in this area.

In the Zadeh-Mamdani type of NFI model, Gaussian fuzzy membership functions are applied in each fuzzy rule for both antecedent and consequent parts, while for the Takagi-Sugeno type of NFI model the consequent part is presented by a linear or non-linear function. A back propagation learning algorithm (Amari, 1990) is used for optimising the parameters of the fuzzy membership functions (in both Zadeh-Mamdani and Takagi-Sugeno types). The distance between two vectors x and y is measured in the NFI model as the *normalised Euclidean distance* defined as follows (the values range between 0 and 1):

$$\|\mathbf{x}-\mathbf{y}\| = \left(\frac{1}{q}\sum_{j=1}^{q}(x_j-y_j)^2\right)^{1/2} \tag{5.1}$$

where $\mathbf{x},\mathbf{y} \in \Re^q$ and q is number of input variables.

To partition the input space N_i for creating and obtaining initial values of

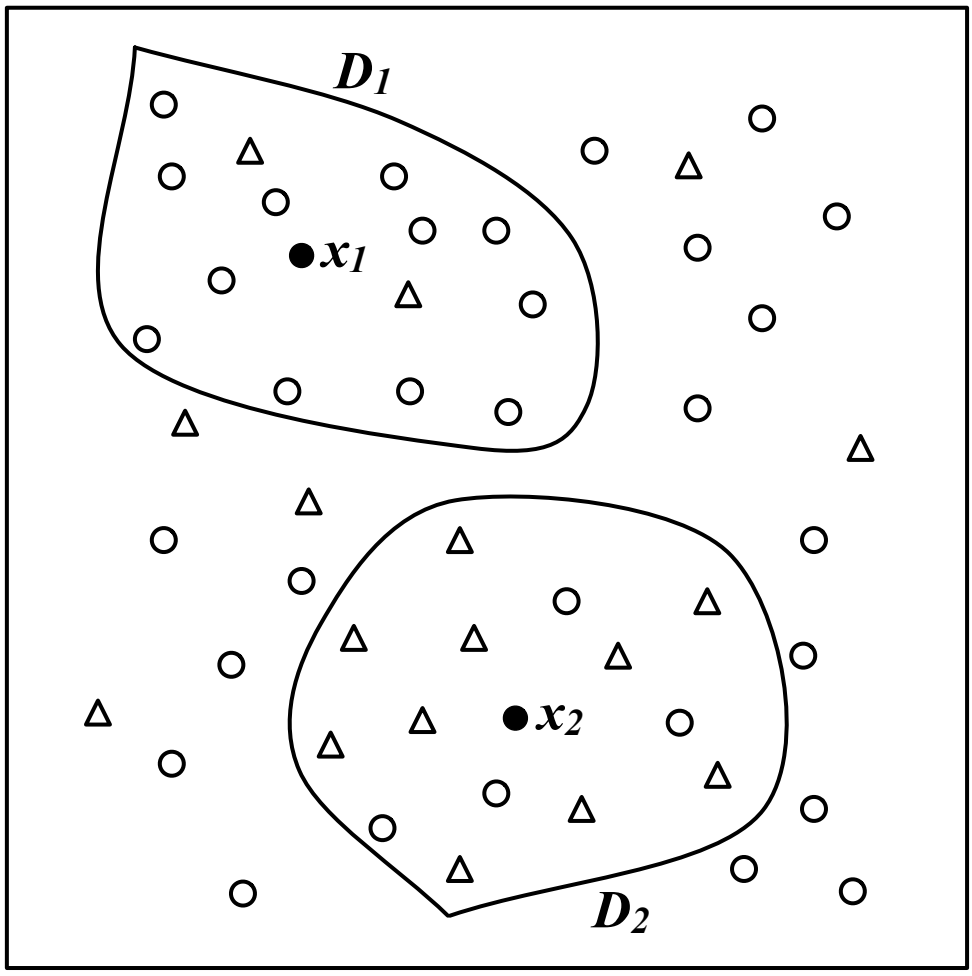


Figure 5.2. In the centre of a transductive reasoning system is the new data vector (here represented by $\mathbf{x}_1$ and $\mathbf{x}_2$), surrounded by a fixed number of nearest data samples selected from the training data D and generated from an existing model M.

fuzzy rules, the ECM (Q. Song & Kasabov, 2001), is applied and the cluster centres and radii are respectively taken as initial values of the centres and widths of the Gaussian membership functions (for both Zadeh-Mamdani and Takagi-Sugeno types). The ECM performs a scatter partition that has relatively small number of clusters covering the space. For the Takagi-Sugeno type of NFI model, the training samples that belong to a cluster are used for

creating a linear function as a local model for output function evaluation.

NFI Learning Algorithm

For each new input vector $\mathbf{x}_i$, the NFI model for transductive reasoning performs the following learning algorithm:

- In the NFI learning algorithm, the following indexes are used:
  - training data samples: $i = 1, 2, ..., N$;
  - input variables: $j = 1, 2, ..., Q$;
  - fuzzy rules: $l = 1, 2, ..., M$;
  - learning epoch: $k = 1, 2, ...$;
- Step 1: search in the training data set based on the input space to find N_i training samples that are closest to $\mathbf{x}_i$. The value for N_i can be predefined based on experience, or optimised through the application of an optimisation procedure. In the NFI model the former approach is used. Here N_i can be considered as the number of nearest neighbours in the k-NN algorithm;
- Step 2: calculate the distances $d_j; j = 1, 2, ..., N_i$ between each of these samples and $\mathbf{x}_i$ using the normalised Euclidean distance (as in Equation 5.1). Calculate the weights $w_j = 1 - (d_j - \min(d)); j = 1, 2, ..., N_i$ where $\min(d)$ is the minimum value in the distance vector $d = [d_1, d_2, ..., d_{N_i}]$;
- Step 3: use the ECM clustering algorithm to cluster and partition the input sub-space that consists of N_i selected training samples;
- Step 4: create fuzzy rules and set their initial parameter values according to the clustering results of the ECM; for each cluster, the cluster centre is taken as the centre of a fuzzy membership function (Gaussian function) and the cluster radius is taken as the width;

- Step 5: apply the steepest descent method (back-propagation) to optimise the parameters of the fuzzy rules in the local model M_i following equations $5.6 - 5.15$;

- Step 6: calculate the output value y_i, for the input vector $\mathbf{x}_i$, applying fuzzy inference over the set of fuzzy rules that constitute the local model M_i.

The procedure of optimising the parameters in the NFI model (Step 5 in the above algorithm) is described as follows:

- Consider the system having Q inputs, one output and M fuzzy rules defined initially through the ECM clustering procedure; the l-th rule would have the form of

  R_l : if $x_1 = F_{l,1}$ and $x_2 = F_{l,2}$ and $x_q = F_{l,q}$ then $y = G_l$ (Zadeh-Mamdani type)

  or

  R_l : if $x_1 = F_{l,1}$ and $x_2 = F_{l,2}$ and $x_q = F_{l,q}$ then $y = n_l$. (Takagi-Sugeno type)

  Here, $F_{l,q}$ are fuzzy sets defined by the following Gaussian type membership function,

$$\text{GaussianMF} = \alpha \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{5.2}$$

  where μ is the centre of the fuzzy membership function, and σ is the width. In the NFI model, the centre of the fuzzy membership function is initially defined by the cluster centre, while the width is defined by the cluster radius. For the Zadeh-Mamdani type, G_l is of a similar type as $F_{l,q}$, while for the Takagi-Sugeno type, n_l is defined by a linear function as follows:

$$n_l = \beta_{l,0} + \beta_{l,1}x_1 + \beta_{l,2}x_2 + ... + \beta_{l,q}x_q. \tag{5.3}$$

where $\beta_{l,0}$ is the *intercept* of the l-th rule, $\beta_{l,1}$ is the parameter associated with x_1 of the l-th rule, $\beta_{l,2}$ is the parameter associated with x_2 of the l-th rule, and so on.

- Using the modified centre average defuzzification procedure (Bezdek, 1981) the output value of the system can be calculated for an input vector $\mathbf{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,p})$ as follows:

$$f(\mathbf{x}_i) = \frac{\sum_{l=1}^{M} \frac{G_l}{\delta_l^2} \prod_{j=1}^{Q} \alpha_{lj} \exp\left(-\frac{(x_{ij}-\mu_{lj})^2}{2\sigma_{lj}^2}\right)}{\sum_{l=1}^{M} \frac{1}{\delta_l^2} \prod_{j=1}^{Q} \alpha_{lj} \exp\left(-\frac{(x_{ij}-\mu_{lj})^2}{2\sigma_{lj}^2}\right)} \quad \text{(Zadeh-Mamdani type)} \tag{5.4}$$

or

$$f(\mathbf{x}_i) = \frac{\sum_{l=1}^{M} n_l \prod_{j=1}^{Q} \alpha_{lj} \exp\left(-\frac{(x_{ij}-\mu_{lj})^2}{2\sigma_{lj}^2}\right)}{\sum_{l=1}^{M} \prod_{j=1}^{Q} \alpha_{lj} \exp\left(-\frac{(x_{ij}-\mu_{lj})^2}{2\sigma_{lj}^2}\right)} \quad \text{(Takagi-Sugeno type)} \tag{5.5}$$

where μ_{lj} and σ_{lj} are the centre and standard deviation of the Gaussian MF of the j-th input variable in rule l derived from the cluster centre and the cluster radius respectively. In addition, α_{lj} and δ_l are the parameters of Zadeh-Mamdani and Takagi-Sugeno type of inference system.

- Suppose the NFI model is given a training input-output data pair $(\mathbf{x}_i, t_i)$, the system minimises the following objective function (a weighted error function):

$$E = \frac{1}{2} w_i \left(f(x_i) - t_i\right)^2 \tag{5.6}$$

where w_i (computed in Step 2 of the NFI learning algorithm) is the contribution weight assigned to the training input-output data pair $(\mathbf{x}_i, t_i)$.

The steepest descent algorithm (backpropagation) (Amari, 1990; Werbos, 1974) is used then to obtain the formulas for the optimisation of the parameters $G_l, \delta_l, \alpha_{lj}, \mu_{lj}$ and σ_{lj} of Zadeh-Mamdani type NFI model such that the value of E from (5.6) is minimised.

$$\begin{aligned} G_l(k+1) &= G_l(k) - \frac{\eta_G}{\delta_l^2(k)} w_i \Phi(\mathbf{x}_i) \\ &\quad \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \qquad (5.7) \\ \delta_l(k+1) &= \delta_l(k) - \frac{\eta_\delta}{\delta_l^3(k)} w_i \Phi(\mathbf{x}_i) \\ &\quad \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \\ &\quad \times \left(f^{(k)}(\mathbf{x}_i) - G_l(k)\right) \qquad (5.8) \\ \alpha_{lj}(k+1) &= \alpha_{lj}(k) - \frac{\eta_\alpha}{\delta_l^2(k)\alpha_{lj}(k)} w_i \Phi(\mathbf{x}_i) \\ &\quad \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \\ &\quad \times \left(G_l(k) - f^{(k)}(\mathbf{x}_i)\right) \qquad (5.9) \\ \mu_{lj}(k+1) &= \mu_{lj}(k) - \frac{\eta_\mu}{\delta_l^2(k)\sigma_{lj}^2(k)} w_i \Phi(\mathbf{x}_i) \\ &\quad \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \\ &\quad \times \left(G_l(k) - f^{(k)}(\mathbf{x}_i)\right)(x_{ij} - \mu_{lj}(k)) \qquad (5.10) \\ \sigma_{lj}(k+1) &= \sigma_{lj}(k) - \frac{\eta_\sigma}{\delta_l^2(k)\sigma_{lj}^3(k)} w_i \Phi(\mathbf{x}_i) \\ &\quad \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \\ &\quad \times \left(G_l(k) - f^{(k)}(\mathbf{x}_i)\right)(x_{ij} - \mu_{lj}(k))^2 \qquad (5.11) \end{aligned}$$

where

$$\Phi(\mathbf{x}_i) = \frac{\prod_{j=1}^{Q} \alpha_{lj} \exp\left(-\frac{(x_{ij}(k) - \mu_{lj}(k))^2}{2\sigma_{lj}^2(k)}\right)}{\sum_{l=1}^{M} \frac{1}{\delta_l^2} \prod_{j=1}^{Q} \alpha_{lj} \exp\left(-\frac{(x_{ij}(k) - \mu_{lj}(k))^2}{2\sigma_{lj}^2(k)}\right)}$$

The steepest descent algorithm (backpropagation) is also used to obtain the formulas for the optimisation of the parameters $\boldsymbol{\beta}_l, \alpha_{lj}, \mu_{lj}$ and σ_{lj} of

the Takagi-Sugeno type NFI model such that the value of E from (5.6) is minimised.

$$\beta_{l0}(k+1) = \beta_{l0}(k) - \eta_\beta w_i \Phi(\mathbf{x}_i) \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \tag{5.12}$$

$$\beta_{lj}(k+1) = \beta_{lj}(k) - \eta_\beta x_{ij} w_i \Phi(\mathbf{x}_i) \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \tag{5.13}$$

$$\alpha_{lj}(k+1) = \alpha_{lj}(k) - \frac{\eta_\alpha}{\alpha_{lj}(k)} w_i \Phi(\mathbf{x}_i) \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \times \left(n_l(k) - f^{(k)}(\mathbf{x}_i)\right) \tag{5.14}$$

$$\mu_{lj}(k+1) = \mu_{lj}(k) - \frac{\eta_\mu}{\sigma_{lj}^2(k)} w_i \Phi(\mathbf{x}_i) \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \times \left(n_l(k) - f^{(k)}(\mathbf{x}_i)\right) \left(x_{ij} - \mu_{lj}(k)\right) \tag{5.15}$$

$$\sigma_{lj}(k+1) = \sigma_{lj}(k) - \frac{\eta_\sigma}{\sigma_{lj}^3(k)} w_i \Phi(\mathbf{x}_i) \times \left(f^{(k)}(\mathbf{x}_i) - t_i\right) \times \left(n_l(k) - f^{(k)}(\mathbf{x}_i)\right) \left(x_{ij} - \mu_{lj}(k)\right)^2 \tag{5.16}$$

where

$$\Phi(\mathbf{x}_i) = \frac{\prod_{j=1}^{Q} \alpha_{lj} \exp\left(-\frac{\left(x_{ij}(k) - \mu_{lj}(k)\right)^2}{2\sigma_{lj}^2(k)}\right)}{\sum_{l=1}^{M} \prod_{j=1}^{Q} \alpha_{lj} \exp\left(-\frac{\left(x_{ij}(k) - \mu_{lj}(k)\right)^2}{2\sigma_{lj}^2(k)}\right)},$$

and $\eta_G, \eta_\delta, \eta_\beta, \eta_\alpha, \eta_\mu$ and η_σ are learning rates for updating the parameters $G_l, \delta_l, \boldsymbol{\beta}_l, \alpha_{lj}, \mu_{lj}$ and σ_{lj} respectively.

5.3 A Novel Method for Multiple Time-Series Analysis & Modelling: mTNFI

As it has been explained in the introductory Section 5.1 of this chapter, the transductive model for multiple time-series analysis and modelling proposed in this study is an extension of the NFI model for transductive reasoning. The proposed method is named the mTNFI. The general principles of the proposed method and its learning algorithm are outlined in the upcoming sections.

General Principles

Essentially, mTNFI employs the same principles as the NFI model, where a dynamic neural-fuzzy inference system with Gaussian membership function is constructed from a set of located nearest neighbours of a new input vector as illustrated in the block diagram of mTNFI in Figure 5.3. However, as the mTNFI model is intended to perform multiple time-series analysis and modelling, some alterations were necessary to the basic NFI model.

The first modification was made in the method used for finding training samples that are most related to the new input vector. The NFI model considers input vector $\mathbf{x}_i$ as a feature vector and uses the normalised Euclidean distance (as in Equation 5.1) to find the closest N_i training samples. Yet, as previous studies have found that dynamic relationships exists between multiple time-series from a specific setting, the basic idea behind the mTNFI model is to use the state of relationship in $\mathbf{x}_t$, where $\mathbf{x}_t = (x_{1t}, x_{2t}, \ldots, x_{qt})'$ and x_{qt} is an expression level of time-series q at time moment t as a feature vector instead of its actual values to locate N_i training samples. This approach has also been investigated in (Widiputra, Pears, & Kasabov, 2009). It should be noted that in mTNFI an input vector is denoted as $\mathbf{x}_t$ instead of $\mathbf{x}_i$ to represent the temporal aspects of the data set. Found instances will then be

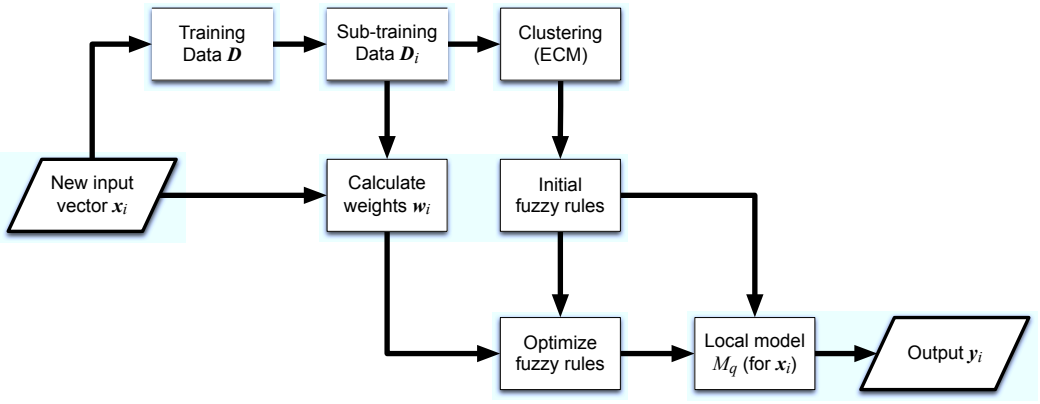


Figure 5.3. Block diagram of the proposed mTNFI model inspired by the NFI model for transductive reasoning.

utilised in constructing a specific inference system to predict future values of the series under examination.

In the mTNFI model, the state of relationship between multiple time-series at a particular time point t, is defined by calculating the *ratio of first order rate of changes* denoted as $R_{\mathbf{X}_t}$, from multiple time-series under examination as follows:

$$R_{\mathbf{X}_t} = \begin{bmatrix} 1 & \frac{x_{1t} - x_{1(t-1)}}{x_{2t} - x_{2(t-1)}} & \cdots & \frac{x_{1t} - x_{1(t-1)}}{x_{qt} - x_{q(t-1)}} \\ \frac{x_{2t} - x_{2(t-1)}}{x_{1t} - x_{1(t-1)}} & 1 & \cdots & \frac{x_{2t} - x_{2(t-1)}}{x_{qt} - x_{q(t-1)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{x_{qt} - x_{q(t-1)}}{x_{1t} - x_{1(t-1)}} & \frac{x_{qt} - x_{q(t-1)}}{x_{2t} - x_{2(t-1)}} & \cdots & 1 \end{bmatrix} \tag{5.17}$$

where $R_{\mathbf{X}_t}$ now is the features matrix describing the state of relationship in $\mathbf{x}_t$. Additionally, the process assumes that $x_{i0} = 0; i = 1, 2, ..., q$. The methodology then employs a feature matrix $R_{\mathbf{X}_t}$ to find N_t closest training samples that form a sub-data set D_t from an existing data set D. Additionally, in place of the normalised Euclidean distance, mTNFI uses the Correlation Coefficient distance measure to quantify similarity level between different features

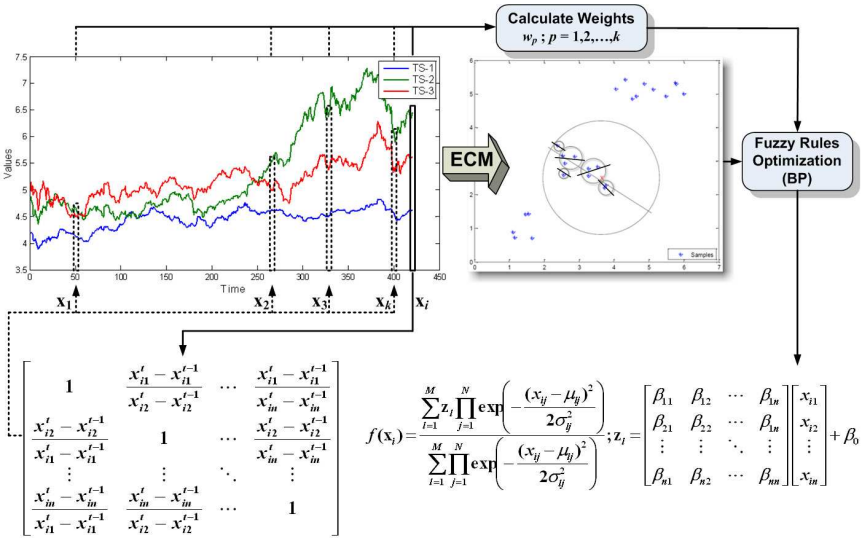


Figure 5.4. Illustration of finding the nearest neighbours, constructing the fuzzy inference system, and output calculation in mTNFI.

matrices defined by,

$$S_{R_{\mathbf{X}_t} R_{\mathbf{X}_i}} = 1 - \frac{\sum_{j=1}^{q}\sum_{k=1}^{q} \left(R_{\mathbf{X}_t j,k} - \overline{R}_{\mathbf{X}_t}\right)\left(R_{\mathbf{X}_i j,k} - \overline{R}_{\mathbf{X}_i}\right)}{\sqrt{\left(\sum_{j=1}^{q}\sum_{k=1}^{q} \left(R_{\mathbf{X}_t j,k} - \overline{R}_{\mathbf{X}_t}\right)^2 \sum_{j=1}^{q}\sum_{k=1}^{q} \left(R_{\mathbf{X}_i j,k} - \overline{R}_{\mathbf{X}_i}\right)^2\right)}} \tag{5.18}$$

where

$$\overline{R}_{\mathbf{X}_t} = \frac{1}{q^2}\sum_{j=1}^{q}\sum_{k=1}^{q} R_{\mathbf{X}_t j,k},$$
$$\overline{R}_{\mathbf{X}_i} = \frac{1}{q^2}\sum_{j=1}^{q}\sum_{k=1}^{q} R_{\mathbf{X}_i j,k},$$

and $i = 1, 2, \ldots, t-1$.

As mTNFI is intended to perform multiple time-series analysis and modelling, the second modification made to the NFI model is the replacement of the linear function in the consequent part of the fuzzy rule by the Takagi-

Sugeno type of fuzzy rule with a different form of linear function as follows:

$$f(\mathbf{x}_t)\begin{cases} y_{1t} = \beta_0 + \beta_{11}x_{1t} + \beta_{12}x_{2t} + ... + \beta_{1q}x_{qt} \\ y_{2t} = \beta_0 + \beta_{21}x_{1t} + \beta_{22}x_{2t} + ... + \beta_{2q}x_{qt} \\ \quad \vdots \\ y_{pt} = \beta_0 + \beta_{p1}x_{1t} + \beta_{p2}x_{2t} + ... + \beta_{pq}x_{qt} \end{cases}, \tag{5.19}$$

where p is the number of dependent variables and q is the number of explanatory variables. However, in mTNFI $p = q$ as the number of time-series being explained is the same as the number of the explanatory time-series. Equation 5.19 can be represented in a more general and simplified form:

$$\mathbf{y}_t = \boldsymbol{\beta}_0 + \boldsymbol{\beta}\mathbf{x}_t \tag{5.20}$$

where $\mathbf{y}_t$ is a vector of dependent variables, $\mathbf{x}_t$ is a vector of independent variables and $\boldsymbol{\beta}$ is the coefficients matrix that maps $\mathbf{x}_t$ to $\mathbf{y}_t$. Representing the consequent part of the fuzzy rule by a linear function with multi dependent and independent variables gives rise to the ability of modelling interactions between observed variables and performing multiple time-series prediction at a particular time point.

Other than the two modifications outlined above, the mTNFI model utilises the same process as the NFI model. As such, for every new input vector, the algorithm dynamically constructs a neural-fuzzy inference system with local generalisation. As in the NFI model, the mTNFI model also employs the Evolving Clustering Method (ECM) proposed by Q. Song and Kasabov (2001), to partition the input sub-space that consists of N_t selected training samples. A local model LM_t for input vector $\mathbf{x}_t$ will then be constituted in the form of a fuzzy inference system using a set of created fuzzy rules derived from the clustering process.

mTNFI Learning Algorithm

For each new input vector $\mathbf{x}_t$ of multiple time-series, the mTNFI model performs the following learning algorithm:

- In the mTNFI learning algorithm, the following indexes are used:
  - training data samples: $t = 1, 2, ..., N$;
  - input variables: $i, j = 1, 2, ..., q$;
  - fuzzy rules: $l = 1, 2, ..., M$;
  - learning epoch: $k = 1, 2, ...$;
- Step 1: construct the ratio of first order rate of changes from input vector $\mathbf{x}_t$ to form features matrix $R_{\mathbf{X}_t}$, using Equation 5.17;
- Step 2: search in the training data set based on the input space, N_t training samples that are closest to $\mathbf{x}_t$ which form a sub-data set $D_t = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_j); j = 1, 2, ..., N_t$, by utilising features matrix $R_{\mathbf{X}_t}$ and calculating features matrices $R_{\mathbf{X}_i}; i = 1, 2, ..., t-1$ from all training samples. Closest training samples in mTNFI are defined using the Correlation Coefficient distance measure, as described in Equation 5.18. Additionally, in mTNFI the value for N_t is pre-defined based on experience, where N_t can be considered as the number of nearest neighbours when being related to the k-NN algorithm;
- Step 3: calculate the distances $d_j; j = 1, 2, ..., N_t$ between each of the training samples in D_t and input vector $\mathbf{x}_t$, and calculate the weights $w_j = 1 - (d_j - \min(\mathbf{d})); j = 1, 2, ..., N_t$ where $\min(\mathbf{d})$ is the minimum value in the distance vector $\mathbf{d} = (d_1, d_2, ..., d_{N_t})$;
- Step 4: use the ECM clustering algorithm to cluster and partition the input sub-space D_t, that consists of N_t selected training samples;

- Step 5: create Takagi-Sugeno type fuzzy rules by representing the consequent part of the rules as a linear function with multiple dependent and independent variables (as in Equation 5.20), and set their initial parameter values according to the clustering results of the ECM.

  For each cluster, the cluster centre is taken as the centre of a fuzzy membership function (Gaussian function) μ and the cluster radius is taken as the width σ.

  Consider at time point t the system under examination has q inputs and outputs, where q is the amount of time-series being observed. As an outcome of this step, M fuzzy rules are defined initially through the ECM clustering procedure, and the l-th rule has the form of:

$$R_l : \text{if } x_{1t} = F_{l1} \text{ and } x_{2t} = F_{l2} \text{ and } ... \text{ and } x_{qt} = F_{lq}, \text{ then } \mathbf{y}_t = f_l(\mathbf{x}_t).$$

  Here, F_{lk} are fuzzy sets of x_k in cluster l, where $k = 1, 2, ..., q$, defined by the following Gaussian type membership function:

$$\text{GaussianMF}_l(x_k) = \alpha_{lk} \exp\left(-\frac{(x_k - \mu_{lk})^2}{2\sigma_{lk}^2}\right). \tag{5.21}$$

  Additionally, $f_l(\mathbf{x}_t)$ in R_l is represented as linear function with multiple dependent and independent variables as follows:

$$f_l(\mathbf{x}_t) = \begin{cases} y_{1t} = \beta_0 + \beta_{11}x_{1t} + \beta_{12}x_{2t} + ... + \beta_{1q}x_{qt} \\ y_{2t} = \beta_0 + \beta_{21}x_{1t} + \beta_{22}x_{2t} + ... + \beta_{2q}x_{qt} \\ \quad \vdots \\ y_{pt} = \beta_0 + \beta_{p1}x_{1t} + \beta_{p2}x_{2t} + ... + \beta_{pq}x_{qt} \end{cases}.$$

  The M created fuzzy rules are then utilised to constitute the local model LM_t in the form of Takagi-Sugeno inference system;

- Step 6: apply the steepest descent method (back propagation) to optimise the parameters of the fuzzy rules in the local model LM_t.

Suppose the mTNFI model is given a training input-output data pair $(\mathbf{x}_t, \mathbf{y}_t)$, the parameters are being optimised by minimising the objective function (a weighted error function) as follows,

$$E = \frac{1}{2} w_t \sum_{k=1}^{q} \left(\hat{y}_{kt} - y_{kt}\right)^2, \tag{5.22}$$

where w_t is the weight of the input-output training data pair, $\hat{y}_t = f(\mathbf{x}_t)$ and q is again the amount of time-series being observed.

In mTNFI, the training input-output data pairs used to optimise the fuzzy rules' parameters are the N_t selected training samples from Step 2, $(\mathbf{x}, \mathbf{y}) = ((\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_j, \mathbf{y}_j)); j = 1, 2, ..., N_t$. The weight of each input-output training data pair w_j, is defined in Step 3;

- Step 7: calculate the output value $f(\mathbf{x}_t)$, for the input vector $\mathbf{x}_t$, applying fuzzy inference over the set of fuzzy rules that constitute the local model LM_t using the modified centre average defuzzification procedure as follows:

$$f(\mathbf{x}_t) = \frac{\sum_{l=1}^{M} f_l(\mathbf{x}_t) \prod_{j=1}^{q} \alpha_{lj} \exp\left(-\frac{(x_{jt} - \mu_{lj})^2}{2\sigma_{lj}^2}\right)}{\sum_{l=1}^{M} \prod_{j=1}^{q} \alpha_{lj} \exp\left(-\frac{(x_{jt} - \mu_{lj})^2}{2\sigma_{lj}^2}\right)}.$$

The procedure for optimising the parameters $\boldsymbol{\beta}_l, \alpha_{lj}, \mu_{lj}$ and σ_{lj} of the Takagi-Sugeno type in the mTNFI model (Step 6 in the mTNFI algorithm) is carried out using the steepest descent method such that the value of E from (Equation 5.22) is minimised. The optimisation equations for each parameter are

then defined as follows:

$$\beta_{l0}(k+1) = \beta_{l0}(k) - \eta_\beta w_t \Phi(\mathbf{x}_t) \times \left(\frac{1}{q} \sum_{i=1}^{q} (\hat{y}_{it}(k) - y_{it}) \right) \tag{5.23}$$

$$\beta_{li,j}(k+1) = \beta_{li,j}(k) - \eta_\beta x_{jt} w_t \Phi(\mathbf{x}_t) \times (\hat{y}_{it}(k) - y_{it}) \tag{5.24}$$

$$\alpha_{li}(k+1) = \alpha_{li}(k) - \frac{\eta_\alpha}{\alpha_{li}(k)} w_t \Phi(\mathbf{x}_t) \times \left(\frac{1}{q} \sum_{i=1}^{q} (\hat{y}_{it}(k) - y_{it}) \right) \times \left(\frac{1}{q} \sum_{i=1}^{q} (\hat{y}_{lit}(k) - \hat{y}_{it}(k)) \right) \tag{5.25}$$

$$\mu_{li}(k+1) = \mu_{li}(k) - \frac{\eta_\mu}{\sigma_{li}^2(k)} w_t \Phi(\mathbf{x}_t) \times \left(\frac{1}{q} \sum_{i=1}^{q} (\hat{y}_{it}(k) - y_{it}) \right) \times \left(\frac{1}{q} \sum_{i=1}^{q} (\hat{y}_{lit}(k) - \hat{y}_{it}(k)) \right) (x_{it} - \mu_{li}(k)) \tag{5.26}$$

$$\sigma_{li}(k+1) = \sigma_{lj}(k) - \frac{\eta_\sigma}{\sigma_{li}^3(k)} w_t \Phi(\mathbf{x}_t) \times \left(\frac{1}{q} \sum_{i=1}^{q} (\hat{y}_{it}(k) - y_{it}) \right) \times \left(\frac{1}{q} \sum_{i=1}^{q} (\hat{y}_{lit}(k) - \hat{y}_{it}(k)) \right) (x_{it} - \mu_{li}(k))^2 \tag{5.27}$$

where

$$\hat{y}_t(k) = f^{(k)}(\mathbf{x}_t); \hat{y}_{lt}(k) = f_l^{(k)}(\mathbf{x}_t),$$

and

$$\Phi(\mathbf{x}_t) = \frac{\prod_{i=1}^{q} \alpha_{li} \exp\left(-\frac{(x_{it}(k) - \mu_{li}(k))^2}{2\sigma_{li}^2(k)} \right)}{\sum_{l=1}^{M} \prod_{i=1}^{q} \alpha_{li} \exp\left(-\frac{(x_{it}(k) - \mu_{li}(k))^2}{2\sigma_{li}^2(k)} \right)},$$

$\eta_\beta, \eta_\alpha, \eta_\mu$ and η_σ are learning rates for updating the parameters $\boldsymbol{\beta}_l, \alpha_{li}, \mu_{li}$ and σ_{li} respectively.

5.4 Experiments on Synthetic Data

In this section of the chapter, to evaluate mTNFI's capability in multiple time-series analysis, modelling and prediction, a pre-generated synthetic multiple time-series data set is once again utilised. The data set has been introduced and used in Chapter 3 and Chapter 4. As in Chapters 3 and 4, the experiment in this chapter is performed by splitting the data set into two different parts, where the first 35 time-points of the complete data set are used as the training data set and the rest of the 15 time-points are defined as the test data set.

However, as mTNFI falls under the category of transductive reasoning or instance-based learning, no training phase is required to construct global or local models as in DIN (see Chapter 3) or LTM (see Chapter 4). Consequently, the training data set is put in place as the initial search space. A number of nearest neighbours that form a sub-data set to construct specific estimation models will then be located from this search space. Additionally, the experiment employs an incremental testing process, which means that whenever a new instance arrives the accuracy of predictions is first tested before it is added to the training set or search space as a training example.

Table 5.1 outlines calculated RMSE of the mTNFI when used to predict movement of the four series of the synthetic data set. It is clear that the mTNFI predicts movement of multiple time-series with good accuracy and in general performs better than the proposed global and local model of multiple time-series prediction (DIN and LTM). In addition, trajectories of predictions as plotted in Figure 5.5 show that the predictions made by the mTNFI closely match with the actual trajectories.

As the mTNFI creates unique sets of fuzzy rules to construct the inference

Table 5.1

mTNFI (transductive model) prediction error rates in RMSE for the synthetic data set against DIN (global model) and LTM (local model).

No	Variable	mTNFI	DIN	LTM
1	Series #1	7.2361	11.9658	7.8696
2	Series #2	6.3772	5.5067	4.1942
3	Series #3	3.0931	7.4662	4.8642
4	Series #4	1.9668	7.6928	5.4827

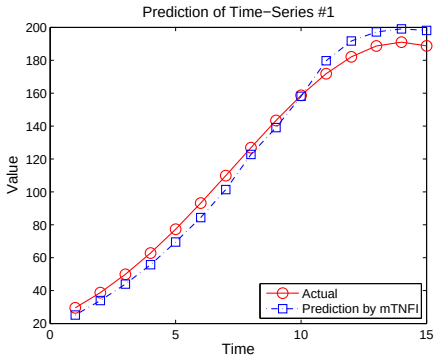
Table 5.2

Number of fuzzy rules created by mTNFI during the testing period.

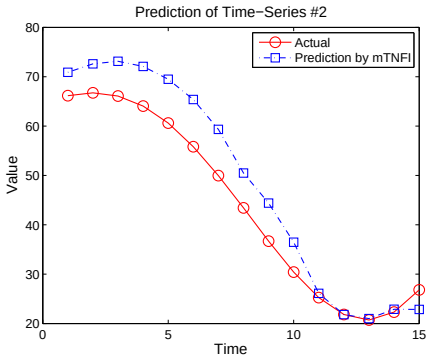
No	Time-Point	Number of Created Rules
1	$1, 4, 6, 7, 12, 13, 15$	2
2	$2, 3, 9, 10, 14$	3
3	$5, 8, 11$	4

system for every new input vector then when predicting movement of the 15 time moments of the test data set the system actually creates different 15 fuzzy inference systems. Each of these inference systems contains a different number of fuzzy rules depending on the results of the clustering process as explained in the previous section of the chapter. Table 5.2 outlines the number of fuzzy rules created for each of the 15 points of the test data set. Additionally, Figure 5.6 describes in detail the set of fuzzy rules created by the mTNFI when predicting movement of the last time-point of the test data set.

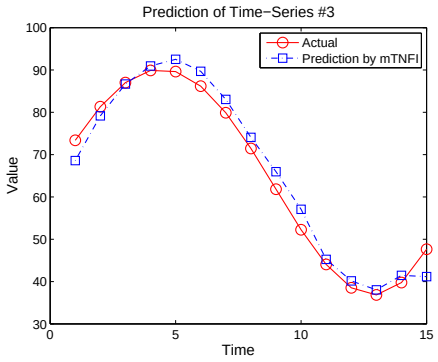
The rules outlined in Figure 5.6 indicate that when Series #1 is moving towards a different direction that is of Series #2, #3 and #4 while Series #2, #3 and #4 are moving together in a similar fashion toward the same direction,



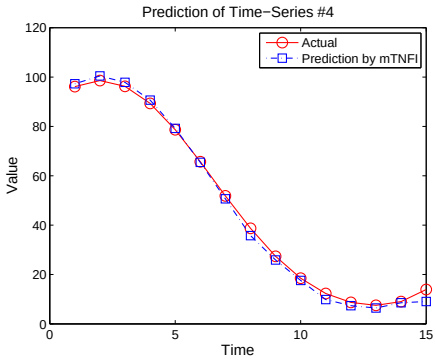
(a) Actual versus estimated trajectories of Series #1.



(b) Actual versus estimated trajectories of Series #2.



(c) Actual versus estimated trajectories of Series #3.



(d) Actual versus estimated trajectories of Series #4.

Figure 5.5. Plot of estimated trajectories of synthetic data set by mTNFI in the testing period.

then the upcoming value of Series #1 is positively dependent to its current value and the current values of Series #2 and #4; and also negatively dependent to current value that is of Series #3. Additionally, the two rules suggest that interdependencies between Series #2, #3 and #4 are more significant compared to their interdependencies with Series #1. This understanding can be acquired by analysing the antecedent and consequent parts of the rules.

The antecedent parts of the rules explain the behaviour of the multiple time-series being observed. It represents the value of ratio of difference be-

Rule1

If RatioOfDiff S1(t)-S2(t) is GaussianMF (-1.44 0.11)
RatioOfDiff S1(t)-S3(t) is GaussianMF (-0.80 0.09)
RatioOfDiff S1(t)-S4(t) is GaussianMF (-1.54 0.12)
RatioOfDiff S2(t)-S3(t) is GaussianMF (0.52 0.11)
RatioOfDiff S2(t)-S4(t) is GaussianMF (1.09 0.09)
RatioOfDiff S3(t)-S4(t) is GaussianMF (1.94 0.13)

Then S1(t+1) = 1.358*S1(t) - 0.163*S2(t) - 0.431*S3(t) + 0.439*S4(t)
S2(t+1) = -0.080*S1(t) + 1.060*S2(t) - 0.022*S3(t) + 0.005*S4(t)
S3(t+1) = 0.234*S1(t) - 0.122*S2(t) + 0.198*S3(t) + 0.719*S4(t)
S4(t+1) = 0.263*S1(t) + 0.075*S2(t) - 1.023*S3(t) + 1.685*S4(t)

Rule2

If RatioOfDiff S1(t)-S2(t) is GaussianMF (-1.47 0.12)
RatioOfDiff S1(t)-S3(t) is GaussianMF (-0.76 0.11)
RatioOfDiff S1(t)-S4(t) is GaussianMF (-1.58 0.14)
RatioOfDiff S2(t)-S3(t) is GaussianMF (0.58 0.09)
RatioOfDiff S2(t)-S4(t) is GaussianMF (1.21 0.11)
RatioOfDiff S3(t)-S4(t) is GaussianMF (1.99 0.13)

Then S1(t+1) = 0.909*S1(t) + 0.263*S2(t) - 0.404*S3(t) + 0.291*S4(t)
S2(t+1) = 0.116*S1(t) + 0.887*S2(t) + 0.247*S3(t) - 0.272*S4(t)
S3(t+1) = 0.153*S1(t) + 0.170*S2(t) + 1.062*S3(t) - 0.494*S4(t)
S4(t+1) = 0.031*S1(t) + 0.420*S2(t) - 0.200*S3(t) + 0.992*S4(t)

Figure 5.6. Created fuzzy rules (first-order Takagi-Sugeno type) by mTNFI when predicting values of synthetic data set at time-point 15 of the testing period.

tween a pair of series (represented by the variable RateOfDiff) which is useful for determining how different the movements of a pair of series are. A positive ratio of difference between a pair of series suggests that the two series are moving toward the same direction (either upward or downward), whilst a negative value indicates that the two series are moving to different directions. Therefore, it can be simply understood from the antecedent parts of the rules that Series #1 is moving towards a different direction while Series #2, #3 and #4 are moving toward the same direction.

The consequent parts of the rules explain the relationship between upcoming value of a series and current values of all series being considered in the observation. Generally, the consequent parts of the rules indicate that the upcoming value of Series #1 is positively dependent to its current value

Table 5.3

Comparison of prediction accuracy of synthetic data set by mTNFI using different numbers of k in the testing period.

No	Variable	$k = 5$	$k = 10$	$k = 15$	$k = 20$	$k = 25$
1	Series #1	8.1371	7.3871	7.5033	7.2361	7.1802
2	Series #2	12.1763	8.8682	7.9151	6.3772	6.3811
3	Series #3	29.4776	13.5841	6.4647	3.0931	3.2351
4	Series #4	26.3551	9.0487	4.4351	1.9668	1.9237

and the current values of Series #2 and #4, while at the same time being considerably affected negatively by current values of Series #3. In addition, the consequent parts also suggest that upcoming values of Series #2, #3 and #4 are being affected insignificantly by the current value of Series #1 and are only mutually dependent to their current values. These analysis of both the antecedent and consequent parts of the rules forms the understanding outlined above.

Throughout the conducted experiment, the number of nearest neighbour k to be found for constructing the fuzzy rules in the mTNFI algorithm is set manually by hand to 20. The selection of k was made based on the prediction accuracy of the testing data set produced by a number of trials using different numbers of k's. Comparison of prediction accuracy when different numbers of k's are selected is outlined in Table 5.3 and illustrated in Figure 5.7.

Having only 20 points to be clustered, it is then appropriate to have only a small number of created clusters and this explains why the number of created fuzzy rules for each test data point lies in the range of 2 to 4 rules (as outlined in Table 5.2). Nevertheless, prediction outcomes of the mTNFI as listed in Table 5.1 and Figure 5.5 confirm that by using only a relatively small number

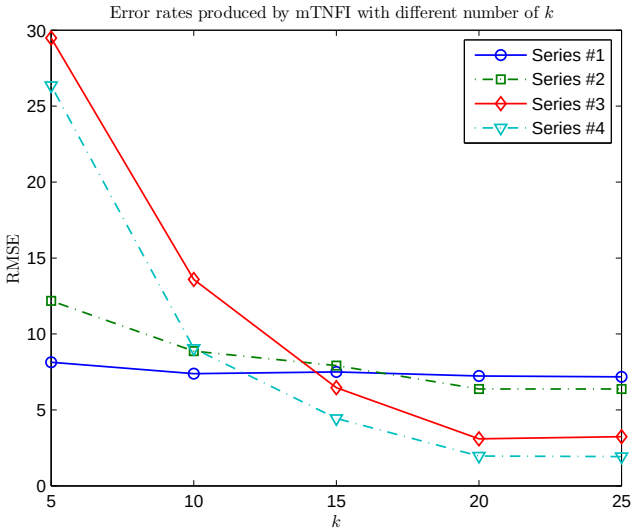
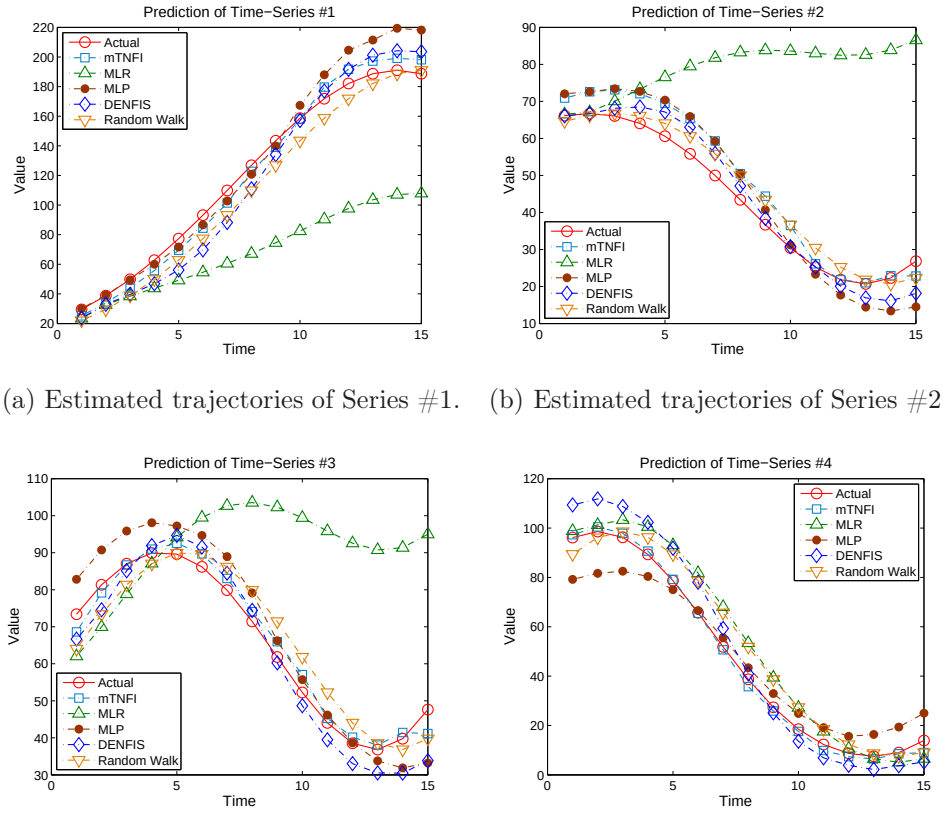


Figure 5.7. Prediction accuracy of synthetic data set by mTNFI using different numbers of k in the testing period.

of rules the methodology is capable of performing simultaneous multiple time-series prediction with a high degree of accuracy.

However, despite its capability to predict movement of multiple time-series with good accuracy, there is a limitation in the proposed methodology. The main weakness of the proposed methodology is that the computational cost of mTNFI increases as the size of the search space (samples) increases over time. This is caused by the nature of the methodology being an instance-based learning algorithm, which searches through all existing samples to find a number of nearest neighbours with the most similarity to the input vector. One approach to prevail over this limitation is to limit the size of the search space.

Another more sophisticated approach that can be implemented (specifically for time-series analysis and modelling) is to develop a searching algorithm that is capable of finding nearest neighbours for current input vector by utilising information of prior input vector's nearest neighbours. Using information about a prior input vector's nearest neighbours, the search space for



(a) Estimated trajectories of Series #1. (b) Estimated trajectories of Series #2.

(c) Estimated trajectories of Series #3. (d) Estimated trajectories of Series #4.

Figure 5.8. Plot of estimated trajectories of synthetic data set by mTNFI versus prediction by other methods applied on single time-series in the testing period.

the current input vector can then be reduced to only those within the relative radius of those nearest neighbours.

A comparative analysis to evaluate the performance of the mTNFI compared to methods applied on single time-series prediction is also conducted in this study. In the experiments, as in Chapters 3 and 4, MLR, MLP and DENFIS are applied as batch models that were trained with the train data set and then tested on the test data set with no incremental learning, whilst random walk model is applied as an incremental model due to its nature. Figure 5.8 depicts the comparison of the actual trajectories compared to the predicted trajectories produced by mTNFI, MLP, MLP, DENFIS and random walk.

Table 5.4

Comparison of mTNFI RMSE prediction error rates against methods applied on single time-series: MLR, MLP, DENFIS and random walk model.

No	Variable	mTNFI	MLR	MLP	DENFIS	Random Walk
1	Series #1	7.2361	62.0580	15.1441	14.3226	12.6634
2	Series #2	6.3772	43.7277	7.5743	4.6464	4.3419
3	Series #3	3.0931	37.0639	7.6572	6.1003	6.4474
4	Series #4	1.9668	10.1890	8.7729	8.8631	8.4156

Plotted trajectories indicate that predictions made by mTNFI match the actual trajectory more closely when compared to the other methods applied on single time-series prediction.

Consequently, the calculated RMSE as outlined in Table 5.4 confirms that mTNFI outperforms other methods applied for a single time-series prediction. This outcome indicates that predicting the movement of multiple time-series simultaneously by constructing an inference system using only samples with similar condition as current input vector results in a better accuracy rather than predicting movement of a single time-series individually.

5.5 Conclusion

The chapter presents a methodology named the Multivariate Transductive Neuro-Fuzzy Inference System denoted as mTNFI for analysis, modelling and prediction of multiple time-series data in which a Takagi-Sugeno type fuzzy model is used to construct a local generalisation over a set of training samples. The proposed algorithm is inspired by the previously proposed method of NFI for transductive reasoning (Q. Song & Kasabov, 2005). When compared to the previously developed inductive inference methods such as MLR, MLP,

etc., the mTNFI has several advantages:

1. As it develops an individual model over the new input vector (by considering the state of relationships between multiple time-series in the new vector to the training samples), mTNFI provides a specific and better local generalisation;

2. The mTNFI model is an adaptive model, in the sense that input-output pairs of data can be added to the data set continuously and immediately made available for transductive inference of local models;

3. By extending the NFI model, the mTNFI model offers the functionality of multiple time-series analysis and modelling. This is the main characteristic of the mTNFI model which differentiates it to other transductive inference methods.

However, as the mTNFI method creates a unique sub-model for each input vector, it usually needs more processing time than inductive models, especially in the case of large data sets. Furthermore, with the existence of new input vectors with exactly the same or very similar condition, the mTNFI model will create the same or similar models repeatedly. It is therefore advantageous to use both incremental inductive reasoning to reveal a global model (the "big picture"), and the mTNFI transductive reasoning for accurate localised inference and decision making.

Time complexity of the method depends mainly on the *search algorithm*, employed for similar data to the new input vector from the complete set of data samples. However, the problem of efficiency of search algorithms is beyond the scope of this work. Further directions for research include: 1) further optimisation of the mTNFI system parameters, such as optimal number of nearest neighbours or samples; 2) investigation on different representations of relationships between multiple time-series at a particular time moment for the

selection of the neighbouring samples; and 3) development of non-linear local models for each cluster from selected samples.

Chapter 6

Integrated Multi-Model Framework for Multiple Time-Series Analysis and Modelling

6.1 Introduction

Global models are capable of capturing global trends in data that are valid for the whole problem space, whereas local models capture local patterns that are valid for subsets of the problem space. In addition, transductive models are capable of constructing local and specific estimation models for every new instance that needs to be classified or predicted. All three approaches are useful for complex modelling tasks and all of them provide complementary information and knowledge, learned from the data. Integrating all of them in a single multi-model system would be a challenging and useful task.

Integrating different types and levels of knowledge about the dynamics of the relationships in a multiple time-series under examination is an imperative step in this study. It is expected that by integrating these knowledge types one should be able to constitute a comprehensive understanding about the underlying behaviour of the dynamics of the system being investigated. This would lead to the possibility of getting better results when predicting the

movement of multiple time-series simultaneously. This idea also relates to previous studies in neural networks that showed that there is no one model that always performs better than the others for all real world problems (Gorr, Nagin, & Szczypula, 1994; S. Ho, Xie, & Goh, 2002; Saad, Prokhorov, & Wunsch, 1998; L. Wang, Liu, & Wang, 2010). For instance, depending on the problem, a simple linear regression model might outperform a neural network model or an SVM, and vice versa.

In relation to this, an integrated scheme to assimilate different types of knowledge has been introduced by Kasabov in the Bioinformatics domain (Kasabov, 2007b). In his study, Kasabov stated that every model has their own power in prediction, and by integrating these models, a powerful model for time-series prediction can be realised. In addition, Kasabov also proposed an integrated multi-model system that includes: a global model, a local model, and a transductive model or so called *personalised* model as introduced by Kasabov in 2007 to increase the accuracy and power of prediction in gene expression data (Kasabov, 2007b).

Nevertheless, no implementation of the proposed integrated framework for multiple time-series analysis and modelling has been made so far. Accordingly, this chapter outlines a methodology to construct a generic integrated framework that not only can be implemented with any kind of time-series data (i.e. financial, biological, weather, etc.), but is also capable of dealing with multiple time-series data. In general, the integrated framework aims to utilise the global and local model for inductive reasoning in recognising and modelling the global trend of multiple time-series interactions along with their profiles of relationships and recurring trends in different time localities. On the other hand, transductive reasoning is put into action to estimate movement of multiple time-series in short and chaotic periods. Ultimately, a generic multi-model framework which integrates complementary knowledge from each model to perform simultaneous multiple time-series prediction is constructed.

The conducted experiments and proposed integrated multi-model framework for multiple time-series analysis and modelling outlined in this chapter intends to address some research questions as follows: (1) Is there any one single model out of the global (DIN), local (LTM) or transductive (mTNFI) models that always performs the best by giving the lowest prediction error over the whole period of examination? If no, (2) can an integrated multi-model framework capable of assigning different level of trust to each model based on their performance in predicting movement of multiple time-series be implemented to increase the overall prediction accuracy?

6.2 Integrated Multi-Model Framework: IMMF

The *Integrated Multi-Model Framework* (IMMF) proposed in this study utilised the three methodologies of multiple time-series analysis and modelling outlined in Chapters 3, 4 and 5 as the main building blocks. Additionally, as explained in the Introductory section of this chapter, the aim of such a framework is to assimilate different types and levels of knowledge extracted by each method to increase the accuracy of predicting movement of multiple time-series simultaneously.

The key idea in constructing the IMMF is to estimate which model out of the global, local and transductive models should be trusted more at any given time-point based on their relative performances in predicting movement of the series under observation. The overall structure of IMMF is illustrated in Figure 6.1.

The main component of the IMMF is the *accumulator* module. The accumulator module will calculate, based on the performance of each model, weight values that will be associated with each model. The output of the accumulator represented by a, is the final prediction formed by the weighted output of

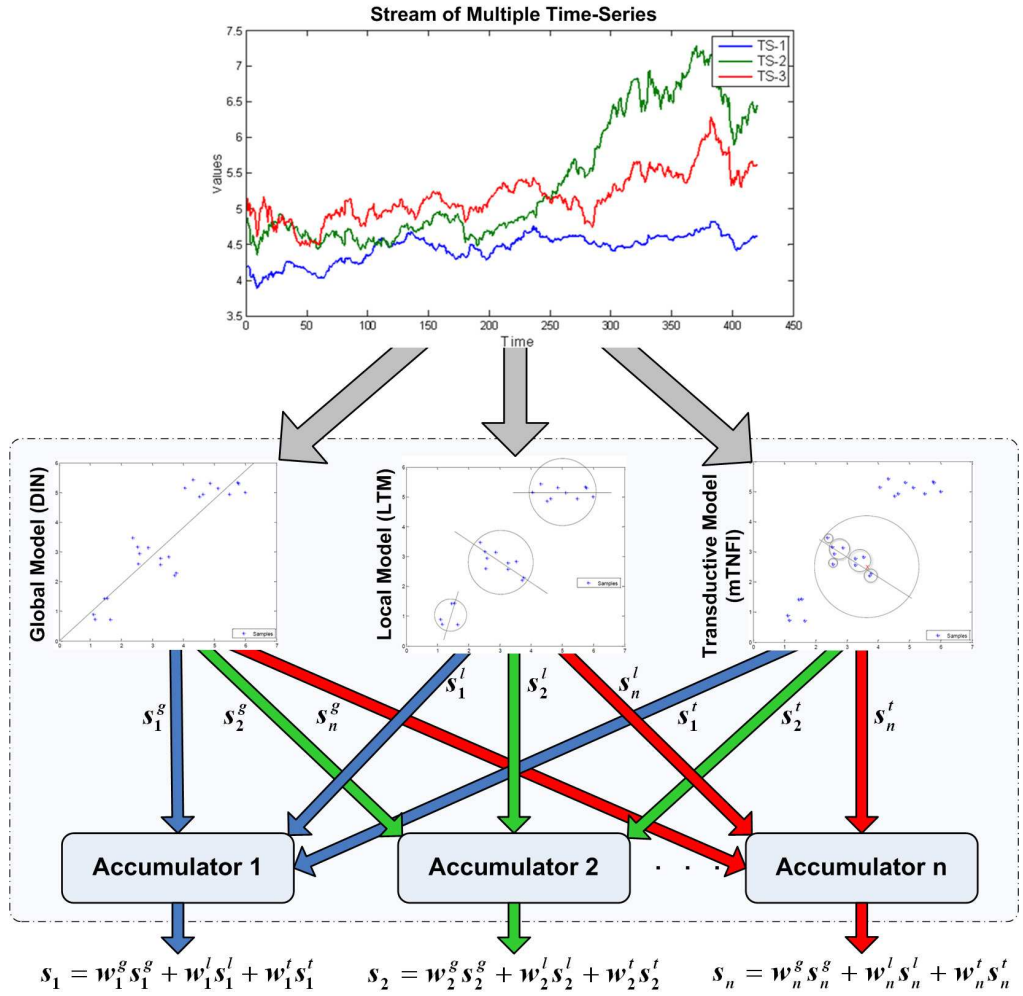


Figure 6.1. The integrated framework of global, local and transductive models for multiple time-series analysis and prediction. n is the number of time-series.

the global, local and transductive models, and is defined by Equation 6.1:

$$s_i = w_i^g s_i^g + w_i^l s_i^l + w_i^t s_i^t, \quad i = 1, \ldots, n \tag{6.1}$$

where s_i^g, s_i^l, s_i^t represent predictions calculated by the DIN as the global model, LTM as the local model and the mTNFI as the transductive model respectively, w_i^g, w_i^l, w_i^t represent the contributing weights assigned to each model and n is the number of time-series. Equation 6.1 describes a linear relationship between the values of the input units and the value of the output unit. Finding the most appropriate weight values to be assigned to each model (rep-

resented by the weight vector w) now amounts to solving a linear optimisation problem.

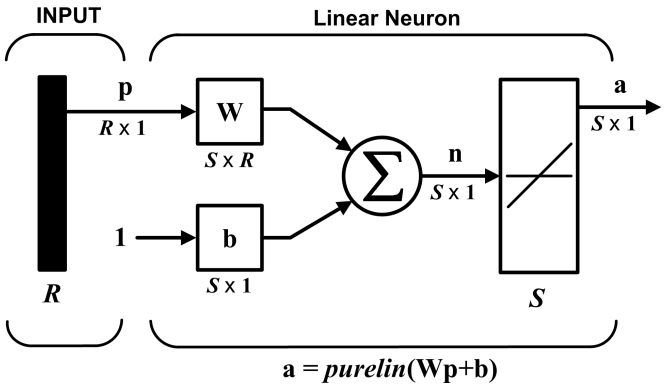
6.2.1 ADALINE Network as the Accumulator Module

The ADALINE network (Frieβ & Harrison, 1999; Labonte, 2002; Shang, 2008; Widrow & Lehr, 1993) is a single layer neural network developed by Bernard Widrow and Ted Hoff at Stanford University in 1960 (Widrow & Hoff, 1960). It is based on the McCulloch-Pitts neuron model (McCulloch & Pitts, 1943). It consists of a connections weight, a bias and a summation function. The key difference between the ADALINE and the standard McCulloch-Pitts perceptron is manifested in the learning process.

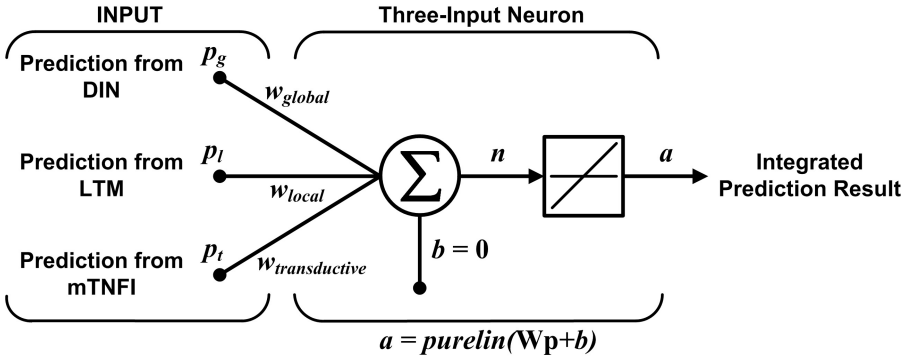
The ADALINE adjusts the connections weight according to the weighted sum of the inputs, whilst in a different way, the standard perceptron adjusts the connection weights according to the output of the activation or transfer function. The structure of the ADALINE network is illustrated in Figure 6.2a, while a simple diagram of its learning process is illustrated in Figure 6.3a.

The accumulator module is the main component of the IMMF and is constructed by utilising the concept of the ADALINE network (its structure and learning process). By using the output from each model (DIN, LTM and mTNFI) as input to the ADALINE network, weight values for each model are based on their relative performances in predicting movements of multiple time-series and can then be estimated through the learning process. Consequently, the weight vector w will represent the trust values given to each model. Additionally, by implementing a recursive learning process these weights can then be recalculated and adjusted based on recent performances of the global (DIN), local (LTM) and transductive model (mTNFI).

Nevertheless, this methodology considers that there is no external interference to the system. In relation to that, the bias value b in the ADALINE



(a) Structure of the ADALINE network (Hagan et al., 1995). *purelin* is the linear transfer function in neural networks where the output is equal to its input with bias (if any is being considered).



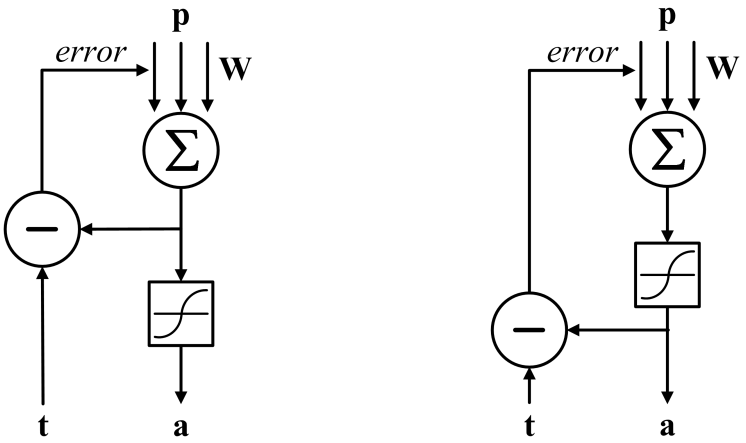
(b) Accumulator module of the IMMF based on the ADALINE network.

Figure 6.2. Illustration of the ADALINE network and the accumulator module of the IMMF.

network structure (as illustrated in Figure 6.2b) is set to 0.

6.2.2 ADALINE Learning Rules

As with the perceptron learning rule, the ADALINE network implements the least mean square (LMS) algorithm which is a supervised learning process (I. Lin & Liou, 2007; T. Lin, Yeh, & Liu, 2010; F. Song & Smith, 2000) in which the learning rule is provided with a set of examples of proper network



(a) Learning process in the ADALINE network.

(b) Learning process in the perceptron neural network.

Figure 6.3. Different learning processes applied to the ADALINE and the perceptron neural network.

behaviour:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}, \tag{6.2}$$

where $\mathbf{p}_q$ is an input to the network, and $\mathbf{t}_q$ is the corresponding target output. As each input is applied to the network, the network output is compared to the target.

In general, the LMS algorithm adjusts the weights and biases (if any is being considered) of the ADALINE network in order to minimise the mean square error, where the error is the difference between the target output and the network output. In relation to constructing the accumulator module of the IMMF, this section of the chapter discusses the performance index of the ADALINE network with just a single-neuron.

For the sake of simplifying the development of the learning rule, all of the parameters to be adjusted (weights and bias) in the ADALINE network are

lumped into one vector (Hagan et al., 1995):

$$\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix}. \tag{6.3}$$

Similarly, the bias input "1" is included as a component of the input vector

$$\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}. \tag{6.4}$$

Consequently the network output, which is usually written in the form

$$a = {}_1\mathbf{w}^T\mathbf{p} + b, \tag{6.5}$$

can now be written as follows:

$$a = \mathbf{x}^T\mathbf{z}. \tag{6.6}$$

This learning rule was introduced by Widrow and Hoff (1960) and therefore is also widely known as the Widrow-Hoff learning algorithm or the *delta rule* (Hui & Zak, 1994; Widrow & Hoff, 1960; X. Zhang, Hang, Tan, & Wang, 1994). The key insight of the learning rule was to estimate the ADALINE network's mean square error $F(\mathbf{x})$ defined by:

$$\hat{F}(\mathbf{x}) = e^2(k) = (t(k) - a(k))^2, \tag{6.7}$$

where the error e in Equation 6.7 is a function of the weights vector w. Consequently, as weights change, the error changes. The objective of the learning process is then to move in "weight space" down the slope of the error function with respect to each weight. Nevertheless, the size of the move should be proportional to the magnitude of the slope. Then at each iteration k a gradient estimate of the form below can be obtained:

$$\nabla\hat{F}(\mathbf{x}) = \nabla e^2(k) \tag{6.8}$$

The first R elements of $\nabla e^2(k)$ are derivatives with respect to the network weights, (where R is the size of input variables) while the $(R+1)^{\text{st}}$ element is the derivative with respect to the bias. Thus we have

$$\left[\nabla e^2(k)\right]_j = \frac{\partial e^2(k)}{\partial w_{1,j}} = -2e(k)p_j(k), \quad \text{for } j = 1, 2, \ldots, R \tag{6.9}$$

and

$$\left[\nabla e^2(k)\right]_{R+1} = \frac{\partial e^2(k)}{\partial b} = -2e(k) \tag{6.10}$$

Therefore the gradient of the square error at iteration k in Equation 6.8 can be written as

$$\nabla \hat{F}(\mathbf{x}) = \nabla e^2(k) = -2e(k)\mathbf{z}(k) \tag{6.11}$$

The term $\mathbf{z}(k)$ is the activation of the unit as only active units contribute to the output and therefore should have their weights adjusted. This differential can now be used to determine the modification of weights. The above approximation to $\nabla F(\mathbf{x})$ can now be used in the steepest descent algorithm, with constant learning rate α as follow,

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k - \alpha \nabla F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k} \\ &= \mathbf{x}_k + 2\alpha e(k) z(k), \end{aligned} \tag{6.12}$$

or

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\alpha e(k)\mathbf{p}(k) \tag{6.13}$$

and

$$b(k+1) = b(k) + 2\alpha e(k) \tag{6.14}$$

Equations 6.13 and 6.14 make up the LMS algorithm which is the learning algorithm for the ADALINE network (also known as the Widrow-Hoff learning algorithm or the delta rule). The preceding equations can be modified to handle the case where we have multiple outputs, and therefore multiple neurons. The equations of the LMS algorithm can then be written conveniently in matrix notation:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha \mathbf{e}(k)\mathbf{p}^T(k) \tag{6.15}$$

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k) \tag{6.16}$$

where the error $\mathbf{e}$ and the bias $\mathbf{b}$ are now vectors.

However, since no bias value is being considered in the integrated framework proposed in this study, what is being estimated through each iteration are only the weights. Additionally, a single accumulator module in the IMMF would produce only a single output (as the integrated prediction results from the global, local and transductive model) and not multiple outputs. Therefore, in the learning algorithm of the IMMF only Equation 6.13 is used.

6.3 Learning Algorithm of the IMMF

To estimate which model out of the global (DIN), local (LTM) and transductive (mTNFI) models should be trusted more at a given time-point t, the IMMF implements a learning algorithm as follows:

- Step 1: for each time-series, by taking previous prediction of each model as the input vector, the first step of the IMMF learning algorithm is simply to calculate the performance of each model, by calculating the absolute error of the predictions;

- Step 2: an initial profile of performance for each time-series is then created based on the absolute errors. This profile of performance ranks each model based on their performance in predicting movement of a particular time-series. The initial profile of performance for time-series i in this methodology is represented by a vector $Vinitial_i$ defined as follows:

  $$Vinitial_i = [m_{i1}, ..., m_{in}],$$

  where m_{i1} is a model that previously has performed the best, that is the one with the smallest absolute error, and m_{in} is a model with the worst previous performance respectively;

- Step 3: a weight vector w_{it} that represents the trust values given to each model based on their relative performances at time moment $t-1$ is then estimated by implementing the ADALINE network and its learning rules as the accumulator module (as described in previous section of the chapter).

  The initial weight vector in the IMMF is defined based on the absolute error of previous prediction of each model. Absolute error of predictions at time-point $t-1$ are taken and mapped using the Gaussian MF where the smallest absolute error is used as the centre of the membership function. Accordingly, a model with the smallest absolute error will then be given the maximum initial weight value of 1, while the other models will be assigned with smaller initial weights.

  As the ADALINE applies supervised learning rule, previous predictions of each model (DIN, LTM and mTNFI) and the actual value at time-point $t-1$ are taken as the input vector and the target output to train the network in finding the best weights with the smallest square error;

- Step 4: a final prediction at time-point t is then made by incorporating the weight vector, prediction of each model and Equation 6.1. The absolute error of the final prediction is then calculated;

- Step 5: as new predictions of the global (DIN), local (LTM) and transductive (mTNFI) models become available for time moment $t, t+1, t+2, ...$, the IMMF re-evaluates performance of each model (as in Step 1) and creates current profile of performance for each time moment defined by $Vcurrenti$ (as in Step 2);

- Step 6: **WHILE** $Vcurrent_i = Vinitial_i$ **AND** absolute error of final prediction $<$ absolute errors that of global, local and transductive models, the IMMF maintains current weight vector as the best solution for

that particular time-series and the algorithm terminates. Otherwise, the algorithm goes to Step 3.

The learning procedure implements two stopping conditions of the ADALINE learning process when finding the best weights with the smallest square error. The first stopping condition is the maximum number of iterations or *epochs*, while the second one is the tolerance level for the square error. The ADALINE learning process will stop estimating the weights if either the maximum number of epochs is reached or if the tolerance level on square error is reached.

6.4 Experiments on Synthetic Data

The same set of synthetic data that was used in previous chapters is again being put in place to evaluate the performance of the proposed IMMF. However, in this experiment predictions (as outcomes from the global (DIN), local (LTM) and transductive (mTNFI) models in previous chapters) are used as input to calculate the final prediction of each time-series.

There are three parameters that need to be set in the proposed IMMF. These three parameters are those of the accumulator module that uses the concept of the ADALINE network. The three parameters are the learning rate α, number of iteration in the learning process or epochs and the tolerance level for the square error. As explained in the previous section, the last two parameters are the stopping condition of the learning process in estimating the weights to be assigned to each of the global, local and transductive model. During conducted experiments the learning rate is set to 0.1, while number of epochs is set to 1000 and the tolerance level is set to $1e^{-5}$.

Figure 6.4 shows the comparison between trajectories of predictions by DIN as the global model, LTM as the local model, mTNFI as the transductive

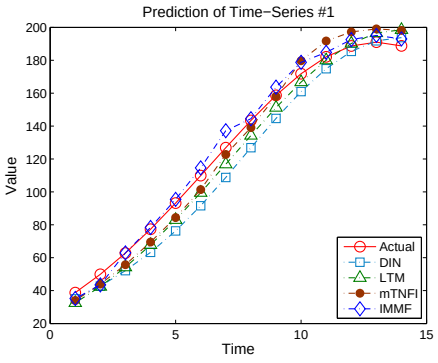
Table 6.1

Comparison of DIN, LTM, mTNFI and IMMF prediction error rates in MAE and RMSE.

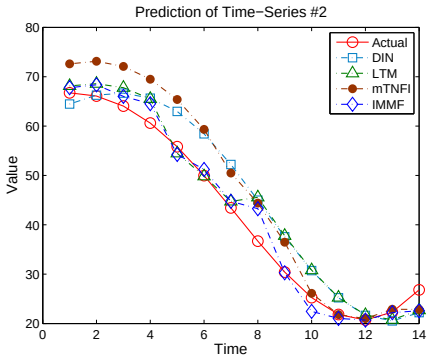
No	Variable	MAE			
		DIN	**LTM**	**mTNFI**	**IMMF**
1	Series #1	10.494	7.3493	6.8040	4.0223
2	Series #2	4.7268	3.3804	5.3793	2.0034
3	Series #3	6.6364	3.9227	2.6342	1.9908
4	Series #4	6.3847	3.5696	1.5934	1.2783
		RMSE			
		DIN	**LTM**	**mTNFI**	**IMMF**
1	Series #1	11.9658	7.8696	7.2361	4.7752
2	Series #2	5.5067	4.1942	6.3772	2.6854
3	Series #3	7.4662	4.8642	3.0931	2.3477
4	Series #4	7.6928	5.4827	1.9668	1.5778

model and the proposed IMMF. Table 6.1 summarises the prediction error rates in Mean Absolute Error (MAE) and RMSE during the testing period for each model. It should be noted that the plots of trajectories have only 14 time moments. This is different compared to the plots of trajectories during the testing period of DIN, LTM and mTNFI in Chapter 3, 4 and 5, which have 15 time moments. The reason for this difference is because the first outcomes of prediction from each model are being utilised to calculate the first weights. Therefore, the first prediction made by the proposed IMMF will start from the second time-point of the testing period resulting in a shorter period then the original one.

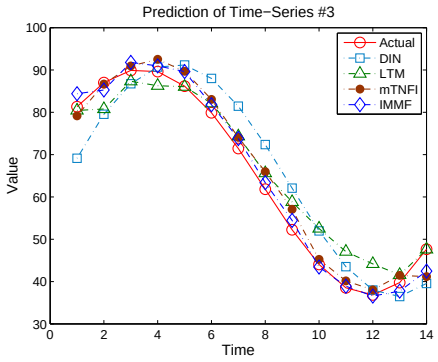
Figure 6.4 shows that the trajectories of predictions calculated by the



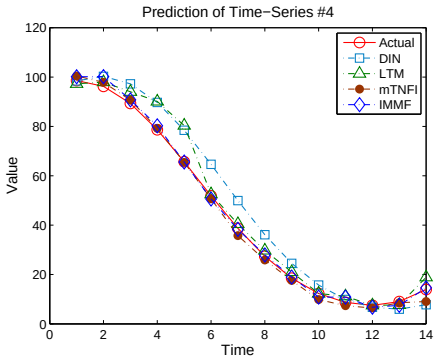
(a) Actual versus estimated trajectories of Series #1.



(b) Actual versus estimated trajectories of Series #2.



(c) Actual versus estimated trajectories of Series #3.



(d) Actual versus estimated trajectories of Series #4.

Figure 6.4. Plot of estimated trajectories of synthetic data set by DIN, LTM, mTNFI and the proposed IMMF in the testing period.

proposed IMMF are in general matching the actual trajectories more closely than those calculated by DIN, LTM and mTNFI. In accordance with these outcomes the calculated error rates as outlined in Table 6.1 reveal equal actuality. This results indicate that integrating outcomes of prediction from various models of time-series analysis might be of help to increase total prediction accuracy.

Figures 6.5 to 6.8 illustrate how the weights assigned to each model are dynamically adjusted over time. In addition, Figures 6.5 to 6.8 depict the

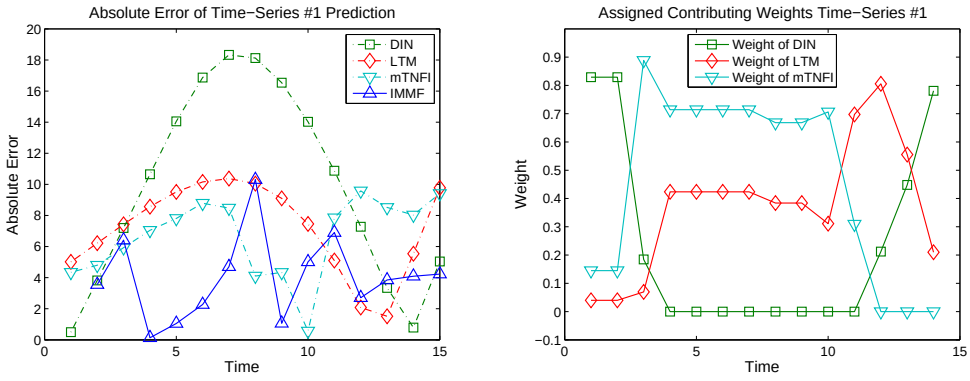


Figure 6.5. Assigned contributing weights to DIN, LTM and mTNFI when predicting movement of Series #1.

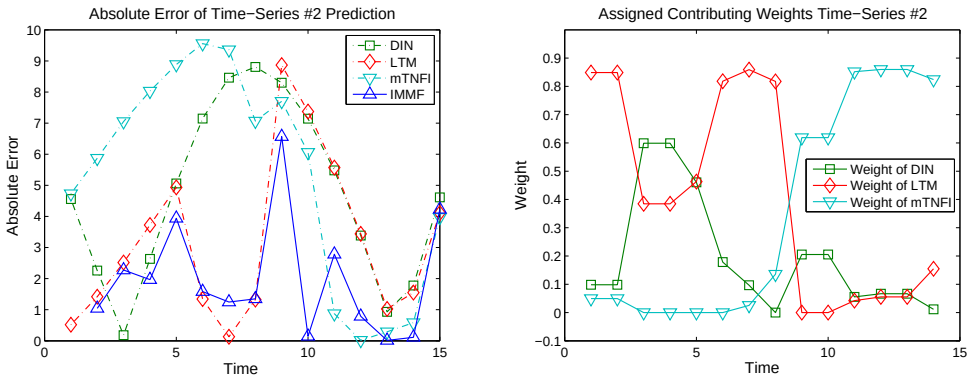


Figure 6.6. Assigned contributing weights to DIN, LTM and mTNFI when predicting movement of Series #2.

calculated absolute error of prediction for each of the global (DIN), local (LTM), transductive (mTNFI) models and the proposed IMMF during the testing period. From these plots it can be seen that the weights assigned to each model at time-point t differs from the weights assigned at time-point $t-1$ if the absolute error of the IMMF is greater than the absolute error of any of the global, local and transductive model, or if the profile of performance between DIN, LTM and mTNFI at time-point $t-1$ has changed from that of $t-2$. This observation confirms the ability of the proposed IMMF to dynamically adjust its level of trust given to each model of multiple time-

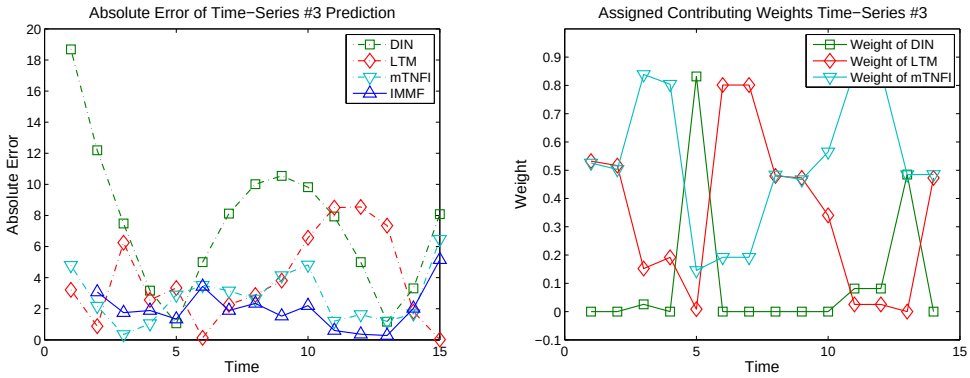


Figure 6.7. Assigned contributing weights to DIN, LTM and mTNFI when predicting movement of Series #3.

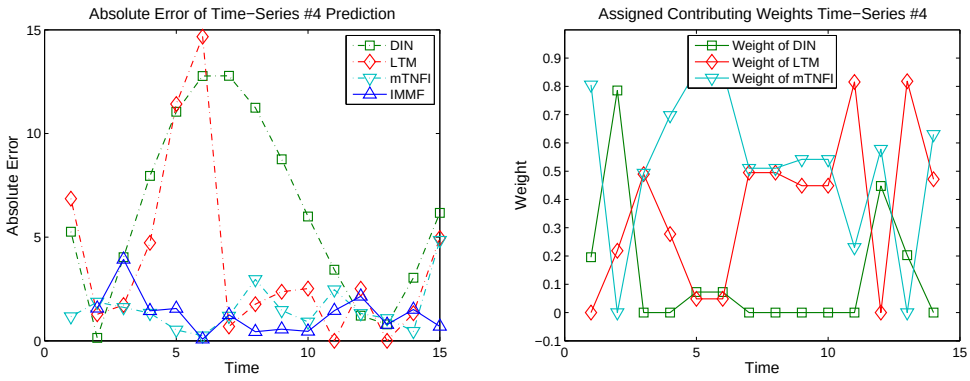


Figure 6.8. Assigned contributing weights to DIN, LTM and mTNFI when predicting movement of Series #4.

series analysis over time.

Figures 6.5 to 6.8 also reveal that most of the time mTNFI performs better than the other two models and therefore was given higher weights more frequently. However, there are also moments when LTM or DIN performs the best and therefore the weights assigned to each model are adjusted accordingly. This finding suggests that having different models for multiple time-series analysis is essential, as each model might perform better than the other at different points in time.

Interestingly, plotted absolute errors of the global (DIN), local (LTM),

transductive (mTNFI) models and the proposed IMMF show that the IMMF does not always give the best prediction results when compared to the other three models. It is noticeable that there are moments when mTNFI, LTM or even DIN gives better results (in terms of absolute error) than the proposed IMMF.

The proposed IMMF makes the final prediction for time-point t by calculating the weighted sum of predictions given by the global, local and transductive models, in which the contributing weights assigned to the three models are determined from their prior performance (time-point $t-1$). This methodology comes with one key limitation, that is: it assumes that current profile of performance between the models is the same as at a previous time moment. However, as the performance of these models varies over time (as illustrated in Figures 6.5 to 6.8), there is always a possibility that the contributing weights based on prior predictions are currently no longer suitable to produce the best prediction. This might lead to the condition where the final predictions of the IMMF are less accurate compared to those produced by the global, local or transductive models individually as found in the conducted experiment.

Nevertheless, it is impractical to eliminate this limitation as it is impossible to assess the current performance of the models prior to having the actual values of the series being analysed. Therefore, to maintain the level of prediction accuracy, what can be done is to apply a learning process that enables the IMMF to dynamically adjust the contributing weights (to be assigned to each model in the next prediction) according to the last profile of prediction accuracy produced by the contributing models. This learning process has been implemented in the proposed IMMF as explained in the previous section.

In addition, Figures 6.5 to 6.8 disclose that in general, the proposed IMMF produces a more stable and lower level of absolute error over the testing period compared to the other models. Furthermore, calculated error rates over the testing period (in MAE and RMSE) as outlined in Table 6.1 indicate that

Table 6.2

Comparison of IMMF prediction error rates against methods applied on single time-series: MLR, MLP, DENFIS and random walk model, in RMSE.

No	Variable	IMMF	MLR	MLP	DENFIS	Random Walk
1	Series #1	4.7752	62.0580	15.1441	14.3226	12.6634
2	Series #2	2.6854	43.7277	7.5743	4.6464	4.3419
3	Series #3	2.3477	37.0639	7.6572	6.1003	6.4474
4	Series #4	1.5778	10.1890	8.7729	8.8631	8.4156

on average the proposed IMMF outperforms the other three models. This observation indicates that the proposed IMMF dynamically adjusts the contributing weights. This dynamic adjustment in turn enables the framework to attach more importance to the model type that is producing more accurate results at the given time points thus enhancing overall accuracy. Having smaller error rates over the testing period also gives the indication that the proposed IMMF is more suitable to be implemented when the objective of the work is to continuously predict movement of dynamic systems over extended period of time and not just for one-off prediction tasks.

Additionally, calculated error rates of the proposed IMMF against other methods applied on single time-series prediction (as performed in previous chapters: Chapters 3, 4 and 5), outlined in Table 6.2, confirms that the proposed IMMF of multiple time-series prediction is superior.

6.5 Conclusion

This chapter proposed an integrated framework named the IMMF that aims to increase the accuracy of multiple time-series prediction by incorporating different models of multiple time-series analysis. In the proposed integrated

framework three different models of multiple time-series analysis (as outlined and proposed in previous chapters) are utilised as the main building blocks. Additionally, an accumulator module to determine the contributing weights of each model in calculating the final prediction result is applied.

The three models of multiple time-series analysis use different type of learning processes to create contrasting views of the problem. The global model (DIN) focuses on the extraction of global trend, the local model (LTM) concentrates on the effects of recurring profiles of relationships and similar trends in the past, and the transductive models (mTNFI) contemplates only recent trends. The proposed IMMF adjusts the contributing weights between the three models using their recent performance in predicting movement of multiple time-series as an indication.

The proposed IMMF applies the concept of the ADALINE network as the accumulator model to estimate the contributing weights assigned to each model in relation to their performance in predicting movement of multiple time-series. The assigned weights represent a given level of trust which will then determine the level of contribution when calculating the final prediction.

Outcomes of conducted experiments with the synthetic data set can be summarised as follows:

1. The absolute error between the global, local and transductive models vary and there is no one single model that always performs better than the others and therefore the contributing weights are dynamically adjusted over time in relation to prior performance (prediction error) of the global, local and transductive model;

2. Prediction trajectories produced by the proposed IMMF (as the sum of weighted contribution of each model) match closely the actual trajectories compared to the prediction trajectories produced individually by each model resulting in smaller error rates;

3. Even though plots of absolute error reveal that there are moments when predictions made by the DIN, LTM or mTNFI are better than the ones made by the proposed IMMF, calculated error rates over the testing period in general indicate the proposed IMMF gives better prediction accuracy compared to the global, local and transductive models.

Point 1 confirms the ability of the proposed IMMF to dynamically adjust the contributing weights between the global, local and transductive model over time. Point 2 confirms that integrating knowledge from different models that treat a problem from different perspectives allows for a more comprehensive understanding of the problem. In the case of multiple time-series prediction this leads to the possibility of achieving better prediction accuracy. In addition, point 3 suggests that the use of the proposed IMMF is more suitable when the objective of the work is to perform continuous time-series analysis and prediction over a relatively longer period of time.

Currently, the proposed IMMF employs an integration method that bases the calculation of the contributing weights only on prior prediction errors. For future direction another integration method that allows a smoother adjustment of the contributing weights can be applied, for instance the use of the average error from each model for the last n time moments in calculating the contributing weights. Furthermore, the use of the ADALINE network to construct the accumulator module in the proposed IMMF can be replaced with other machine learning or parameter optimisation algorithms.

Chapter 7

Case Study 1: Analysis and Modelling of Global Stock Market Indexes in the Asia Pacific Region

7.1 Introduction

The globalised security markets of today are characterised by interdependencies, and often demonstrate contagious behaviour in periods of crisis. Interaction between stock markets have been researched in the past few years. Globalisation has brought interdependence among stock markets around the world, in which a change in one stock market affects other stock markets. An increasing number of studies are addressing the effects of such interrelationships, along with the challenge of relationship identification and modelling within a globalised environment.

A study by Phylaktis and Ravazzolo (2005) found that the relationship between international stock markets is stronger than in the preceding years, due to the relaxation of foreign ownership restrictions. Some other researchers also found that local stock markets are generally influenced by major stock markets in the world (i.e. Europe, Africa, Australia, U.S. and Asia) (Adam & Tweneboah, 2008; Beelders, 2002; Drew & Chong, 2002; Glezakos, Merika, & Kaligosfyris, 2007; Isakov & Parignon, 2000; Y. Liu & Sun, 2008; Psillaki & Margaritis, 2008). Consequently a number of studies that applied the co-

integration analysis method to find the correlation between the stock markets have also been conducted (Brahmasrene & Jiranyakul, 2007; Clout & Willett, 2009; Okunev, Wilson, & Zurbruegg, 2000).

Chiang and Doong (2001) considered stock returns and volatility in their study and found that four out of seven Asian markets present a significant relationship between stock returns and unexpected volatility. In research conducted by the French, German and UK indexes, and the corresponding stock index future markets, Antoniou et al. (2003) signified that the behaviour of a domestic market was influenced by foreign markets. Collins and Biekpe (2003) further studied contagion and interdependence in the African markets and found evidence for contagion from the most traded markets, i.e. Egypt and South Africa, while Serguieva, Kalganova, and Khan (2003); Serguieva and Wu (2008) , who focused their studies on Asia during the time of crisis, suggested that investigating the characteristics of financial contagion will contribute to recognising at an earlier stage financial crises which potentially destabilise cross-market linkages (Serguieva et al., 2003; Serguieva & Wu, 2008).

However, these studies do not appear to simultaneously capture multiple dynamic relationships between interactive markets. This task serves as the main reason to why this study is looking at the possibility of implementing the computational intelligence approaches (global, local and transductive) to extract such relationships. Furthermore, to meet the challenge of analysing and modelling dynamic interactive multiple markets in predicting their future values, it is essential to capture their interactive behaviour in a dynamic fashion.

In this chapter of the thesis, financial data of stock market indexes in the Asia Pacific region is used as a case study to examine the performance of DIN, LTM, mTNFI and IMMF for the analysis, modelling and prediction of multiple time-series data.

7.2 Interactive Stock Markets Analysis and Modelling

The main objective of the work outlined in this chapter is to evaluate performance of DIN, LTM, mTNFI and IMMF in predicting movement of multiple stock market indexes in the Asia Pacific region. Additionally, an analysis of the dynamics of interactions, profiles of relationships in different time localities and the recurring trends of movement between the stock markets under observation is also conducted.

7.3 Data Description

The financial data set used in this study includes time-series indexes of 10 stock markets in the Asia Pacific region (Yahoo! Finance, 2010), spanning 155 weeks from July 2007 to June 2010. The weekly aggregated values of the stock market indexes are considered here. The 10 selected equity markets (in alphabetical order) are: Australia AORD, Hong Kong HSI, Indonesia JSX, Malaysia KLSE, South Korea KOSPI, Japan Nikkei 225, New Zealand NZ50, Shanghai China SSX, Singapore STI and Taiwan TSEC. Figure 7.1 depicts the trajectories of the 10 stock markets in the period of examination.

Throughout the conducted experiment the data set is divided into two different parts: the training part and the testing part. The training part consists of the first 110 points of the weekly stock market indexes. Consequently, the testing part consists of the remaining 45 points from the complete set of 155 points. The experiment employs an incremental testing process, which means that whenever a new instance arrives, a prediction is first made for the new instance before it is added to the training set as an additional training example.

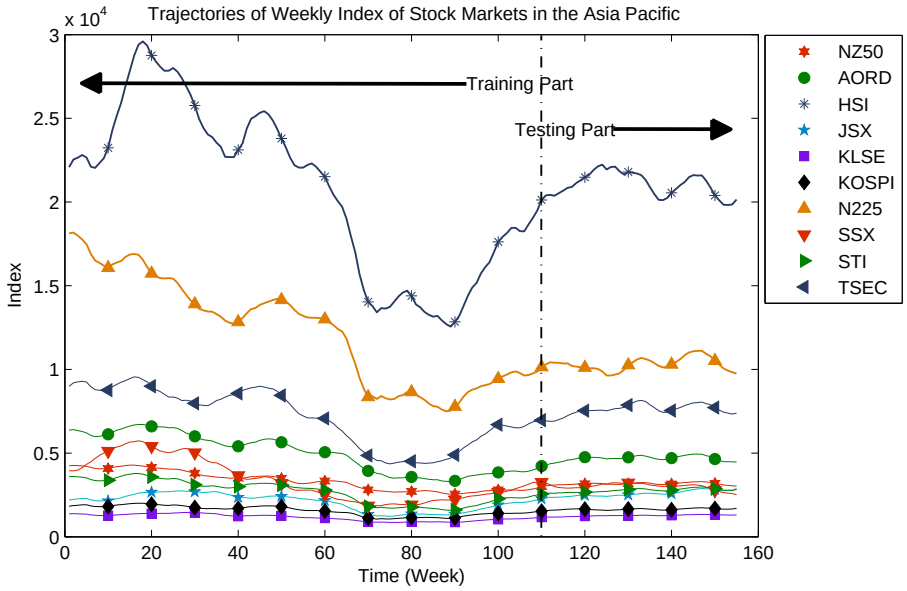
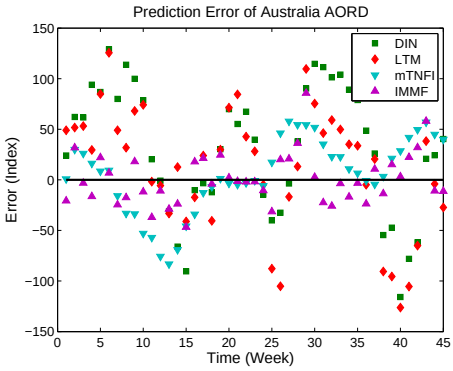


Figure 7.1. Weekly index of 10 stock markets in the Asia Pacific region spanning 155 weeks from July 2007 to June 2010.

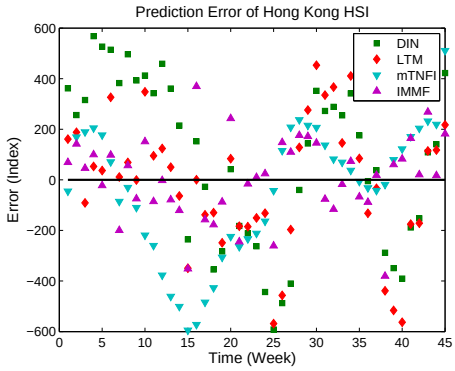
7.4 Experimental Results and Comparison

In order to evaluate the performance of the proposed models in predicting movement of a number of time-series simultaneously, scatter plots of absolute error of prediction from each model are used to illustrate the distribution of error. Additionally the RMSE is used as the error measurement.

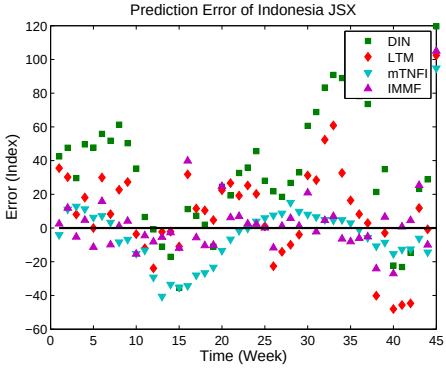
Figures 7.2 and 7.3 are the scatter plots of errors produced by DIN, LTM, mTNFI and IMMF when predicting movement of weekly 10 stock market indexes in the Asia Pacific region. The scatter plots reveal that the prediction accuracy from each model varies during the testing period and there is no one single model that always works better than the others across the testing period. For instance, even though the scatter plots show that in general DIN gives the worst performance compared to the other methods, there are moments when DIN performs the best and outperforms the other models. Similar situations are also found for LTM, mTNFI and IMMF as it is depicted



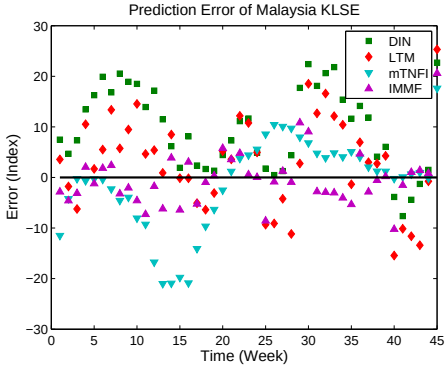
(a) Prediction error of Australia AORD



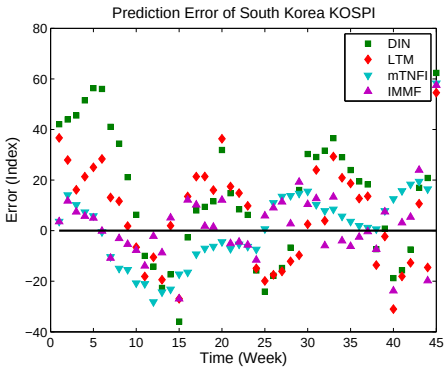
(b) Prediction error of Hong Kong HSI



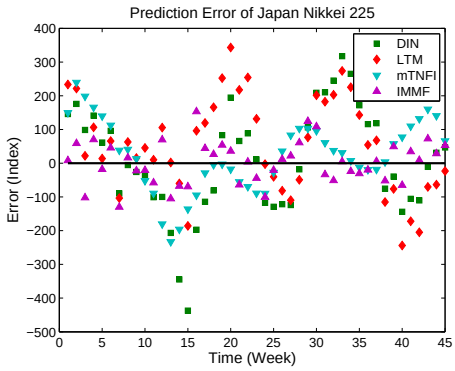
(c) Prediction error of Indonesia JSX



(d) Prediction error of Malaysia KLSE

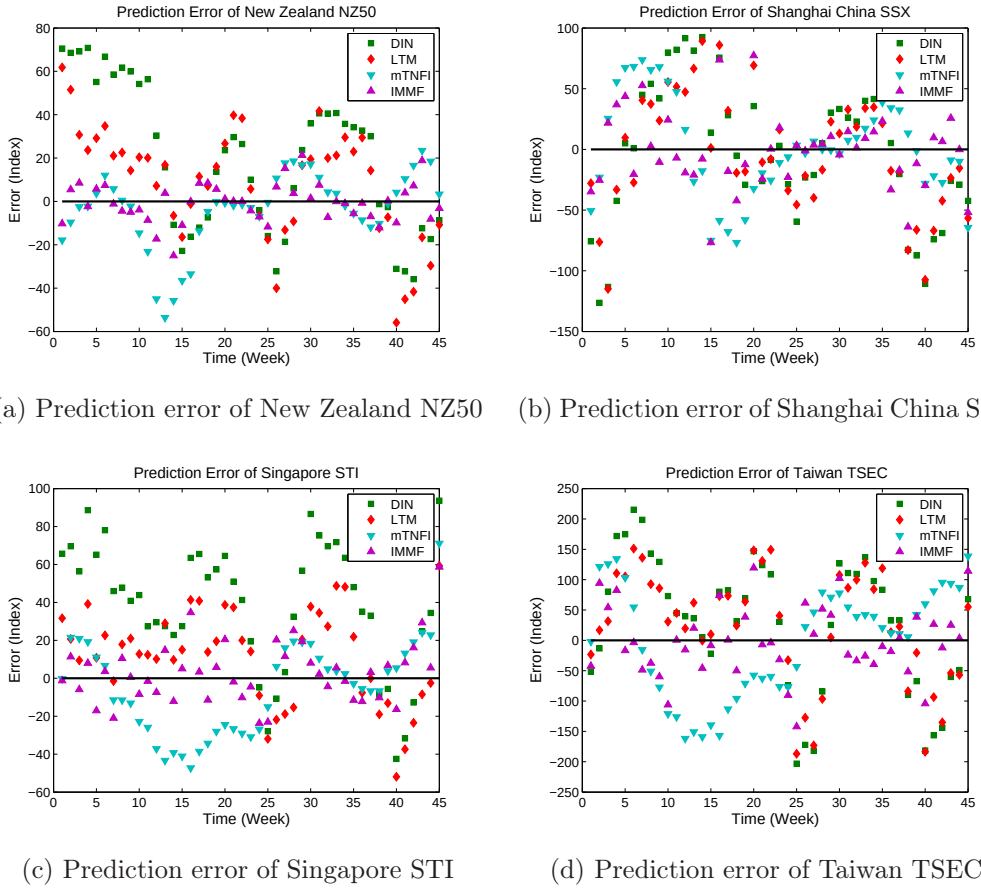


(e) Prediction error of South Korea KOSPI



(f) Prediction error of Japan Nikkei N225

Figure 7.2. Prediction error of Australia, Hong Kong, Indonesia, Malaysia, South Korea and Japan stock market indexes.



(a) Prediction error of New Zealand NZ50

(b) Prediction error of Shanghai China SSX

(c) Prediction error of Singapore STI

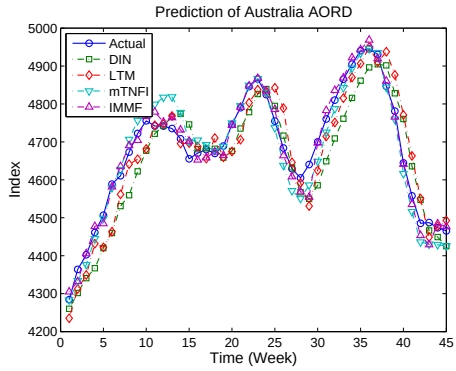
(d) Prediction error of Taiwan TSEC

Figure 7.3. Prediction error of New Zealand, Shanghai China, Singapore and Taiwan stock market indexes.

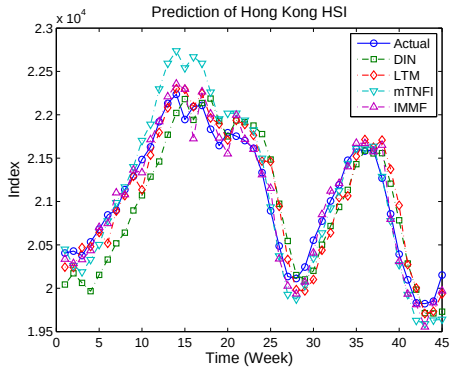
by the plots.

However, Figures 7.4 and 7.5 show that the predictions made by the three models and the integrated framework closely match with the actual trajectories. In addition, plots of prediction error (as illustrated in Figures 7.2 and 7.3) indicate that the integrated framework is often the most accurate model when predicting movement of the stock market indexes across the whole period of examination. Table 7.1, which outlines the RMSE of predictions made by each model, is in compliance with this actuality.

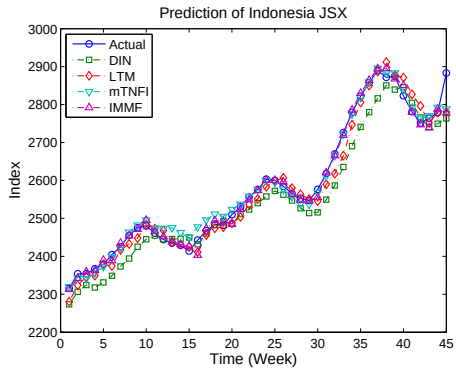
Figures 7.6 to 7.10 illustrate how the contributing weights of DIN, LTM



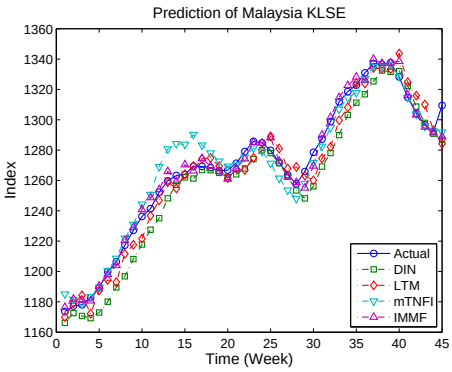
(a) Prediction of Australia AORD



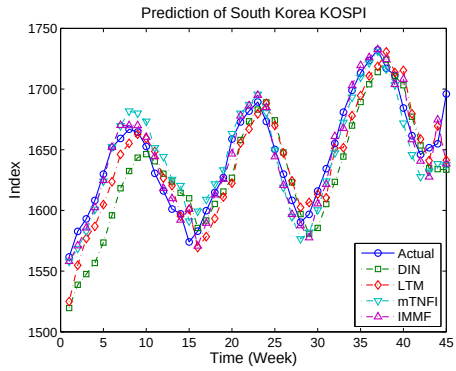
(b) Prediction of Hong Kong HSI



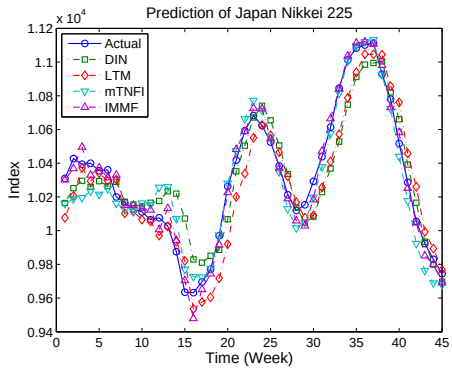
(c) Prediction of Indonesia JSX



(d) Prediction of Malaysia KLSE

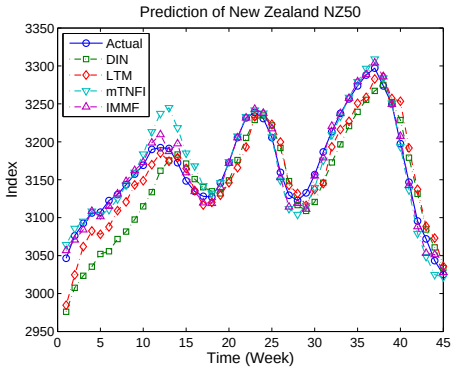


(e) Prediction of South Korea KOSPI

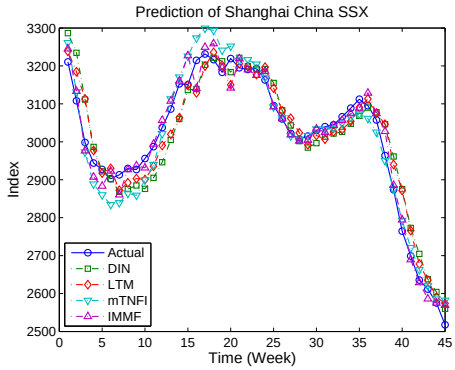


(f) Prediction of Japan Nikkei 225

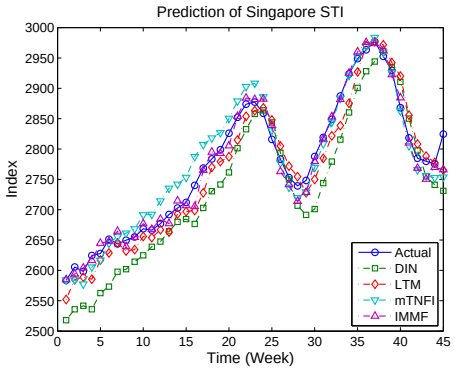
Figure 7.4. Prediction of Australia, Hong Kong, Indonesia, Malaysia, South Korea and Japan stock market indexes.



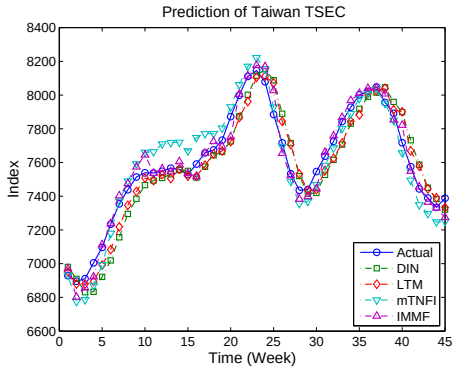
(a) Prediction of New Zealand NZ50



(b) Prediction of Shanghai China SSX



(c) Prediction of Singapore STI



(d) Prediction of Taiwan TSEC

Figure 7.5. Prediction of New Zealand, Shanghai China, Singapore and Taiwan stock market indexes.

and mTNFI defined by IMMF to calculate the final prediction are changing dynamically in relation to movement of the absolute error of each model. These plots indicate that the integrated framework is capable to adjust the contribution weights assigned to each contributing model to maintain its accuracy level of prediction over time.

To validate if forecasting movements of multiple time-series simultaneously offers better prediction accuracy, a comparative analysis with MLR, MLP and random walk methods applied on single time-series is conducted in this study. The random walk model is a time-series analysis that assumes that next value

Table 7.1

Error rates in RMSE of DIN, LTM, mTNFI and IMMF when predicting movement of weekly stock market indexes in the Asia Pacific region.

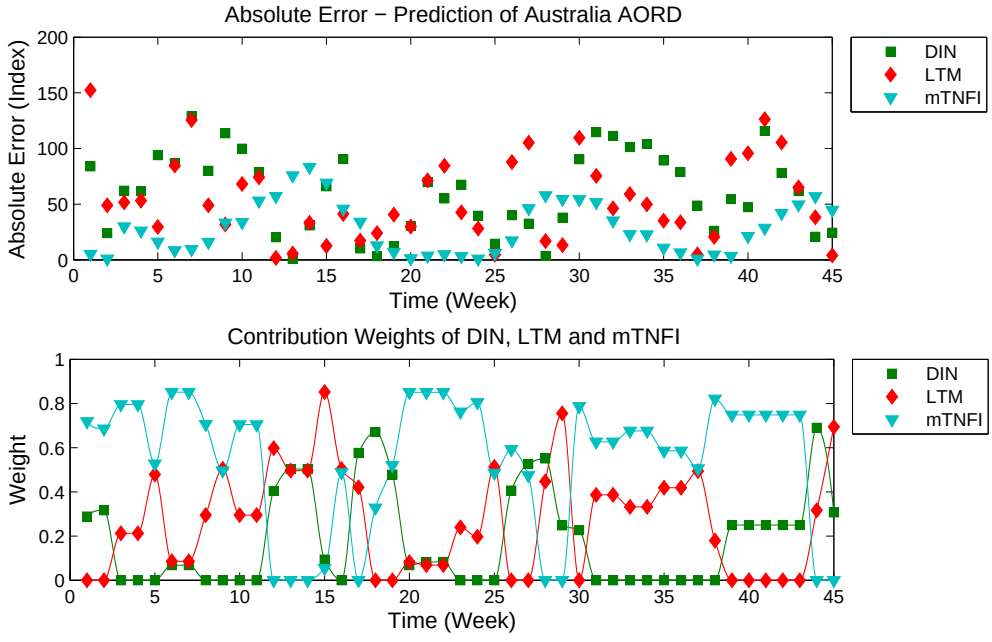
No	Stock Market	DIN	LTM	mTNFI	IMMF
1	Australia AORD	68.8629	60.5921	36.7233	25.1345
2	Hong Kong HSI	330.4391	254.4411	258.9914	156.3394
3	Indonesia JSX	47.6129	29.3084	21.1106	19.8513
4	Malaysia KLSE	12.4014	9.3991	9.0178	5.2856
5	South Korea KOSPI	28.0342	20.2284	15.4878	13.5639
6	Japan Nikkei 225	156.0617	149.4570	105.9573	61.8075
7	New Zealand NZ50	38.1143	27.1600	17.8022	8.8974
8	Shanghai China SSX	56.5567	48.1407	40.4320	30.7801
9	Singapore STI	50.8090	27.2544	24.8119	16.1047
10	Taiwan TSEC	112.3427	94.7727	89.0169	57.3583

of a time-series is equal to current value. Here the random walk without drift model defined simply by Equation 7.1 is used.

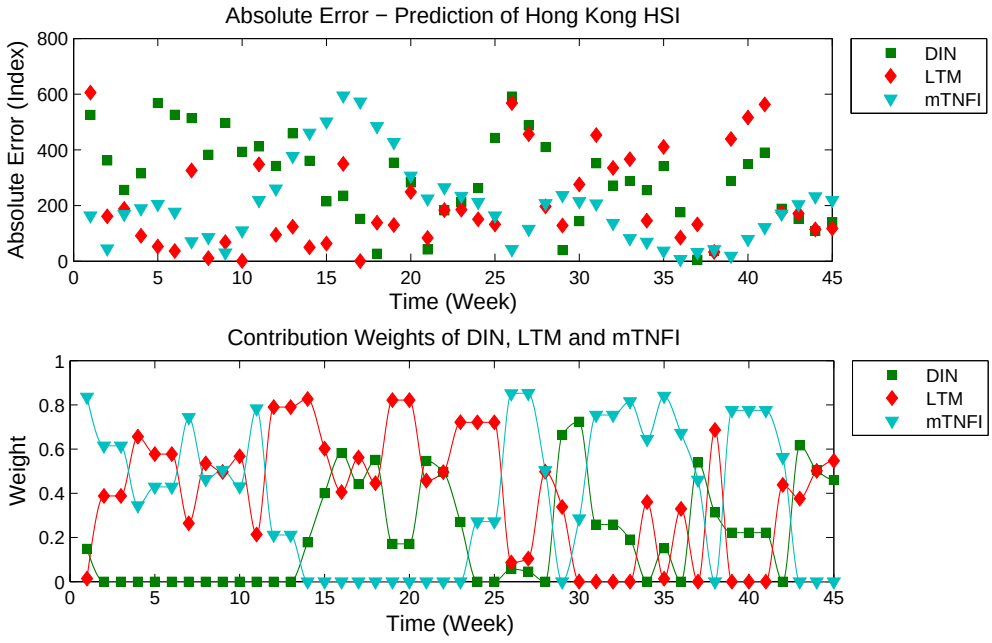
$$x_{t+1} = x_t \tag{7.1}$$

The random walk model in some cases might offer better prediction accuracy (in terms of sum squared residual). However, this model produces a shadow plot of the observed data, lagging exactly one period behind and providing no knowledge on the observed system as it simply assumes that the upcoming value is exactly the same as current value. Therefore, it can be considered that this model is actually have no predictive power.

Table 7.2 shows the much smaller RMSE of the IMMF in comparison to the other methods applied for single time-series prediction. This outcome clearly indicates the value of extracting and exploiting relationships between

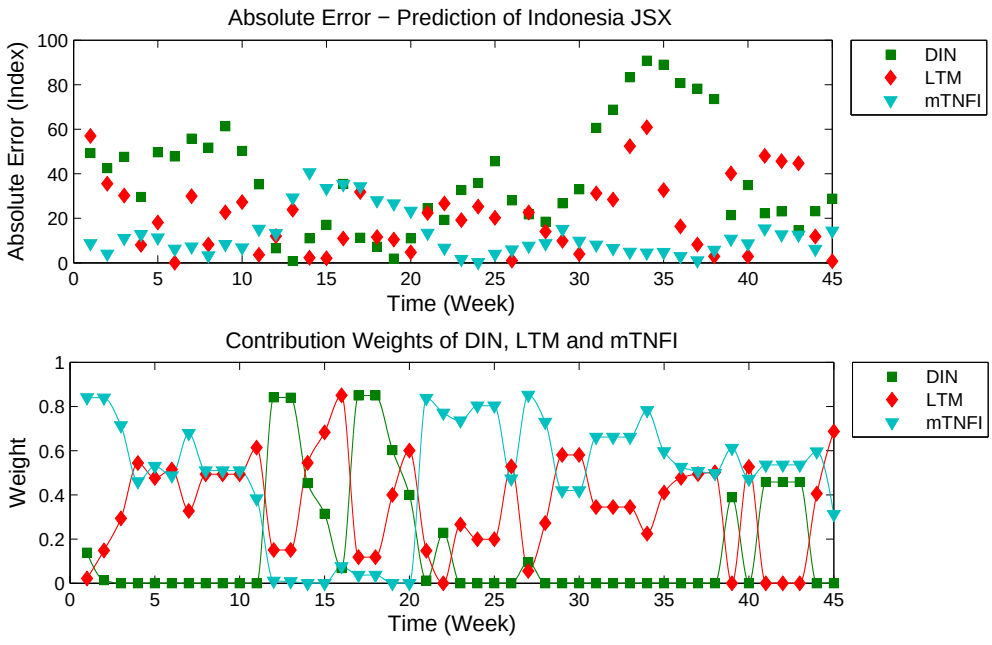


(a) Assigned contributing weights for prediction of AORD

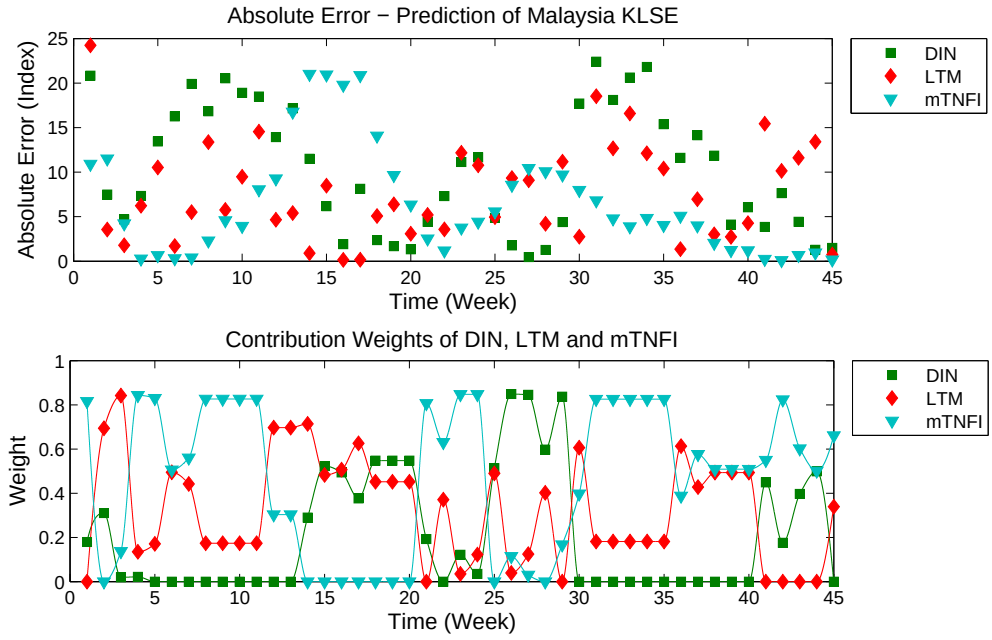


(b) Assigned contributing weights for prediction of HSI

Figure 7.6. Assigned contributing weights for prediction of Australia and Hong Kong stock market indexes.

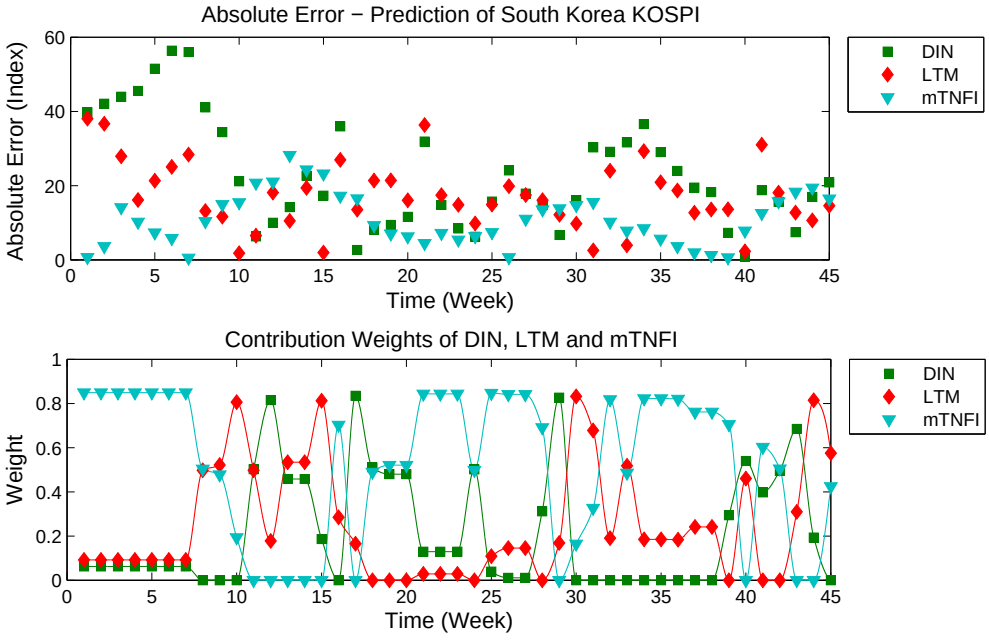


(a) Assigned contributing weights for prediction of JSX

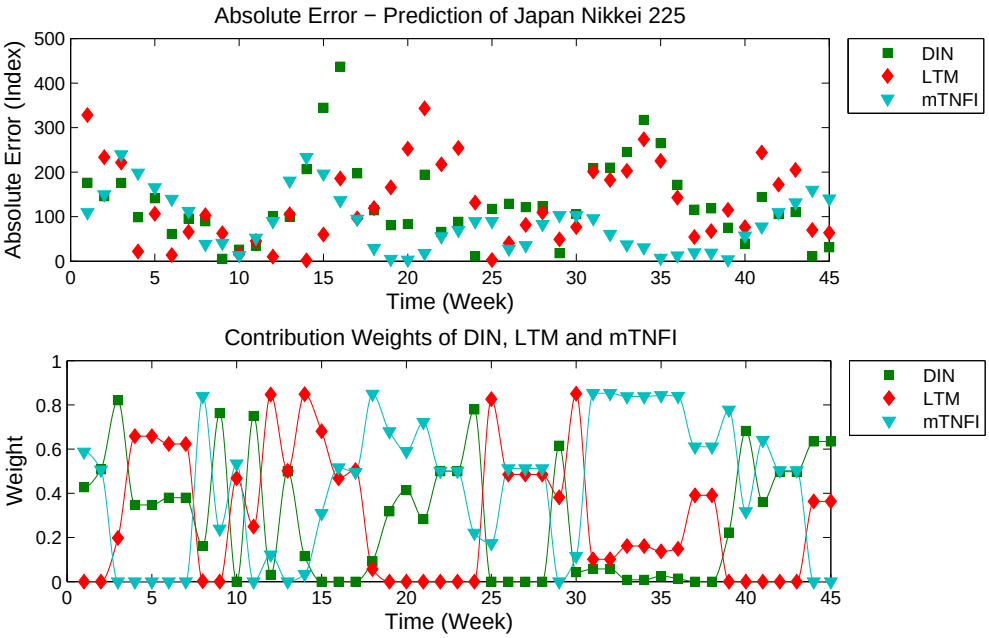


(b) Assigned contributing weights for prediction of KLSE

Figure 7.7. Assigned contributing weights for prediction of Indonesia and Malaysia stock market indexes.

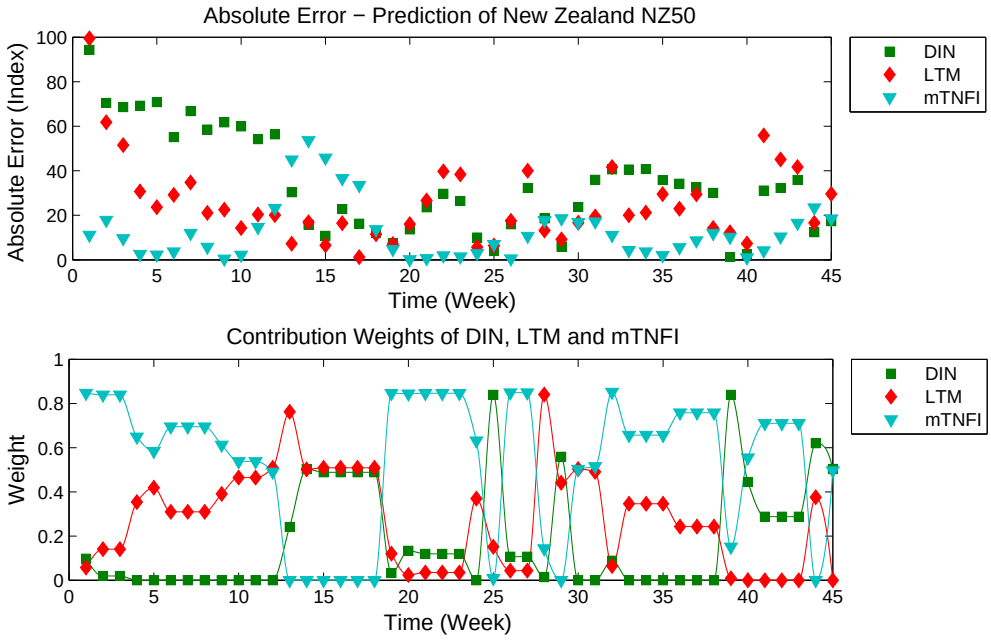


(a) Assigned contributing weights for prediction of KOSPI

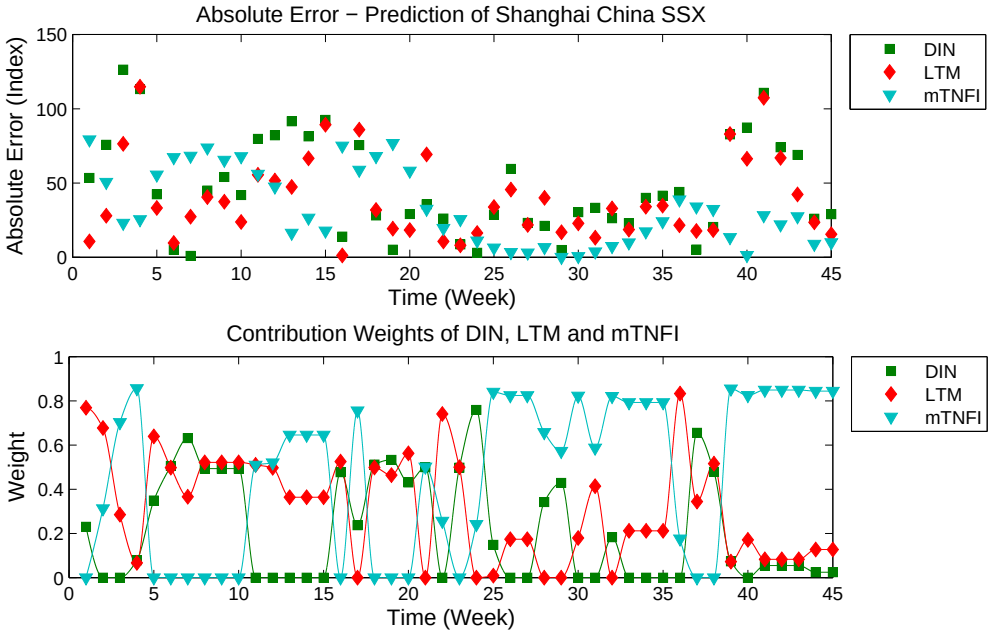


(b) Assigned contributing weights for prediction of Nikkei 225

Figure 7.8. Assigned contributing weights for prediction of South Korea and Japan stock market indexes.

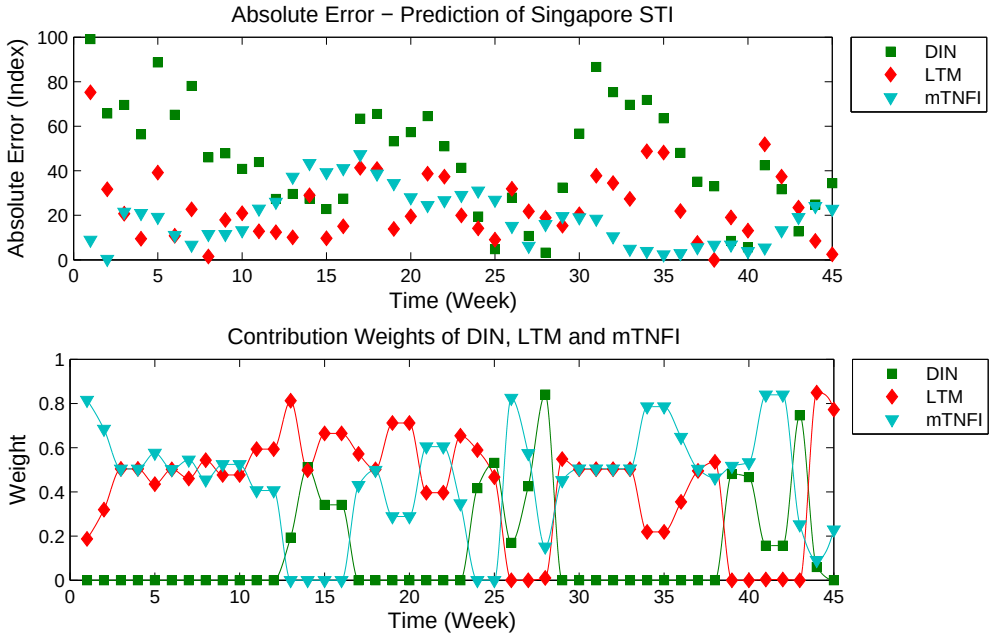


(a) Assigned contributing weights for prediction of NZ50

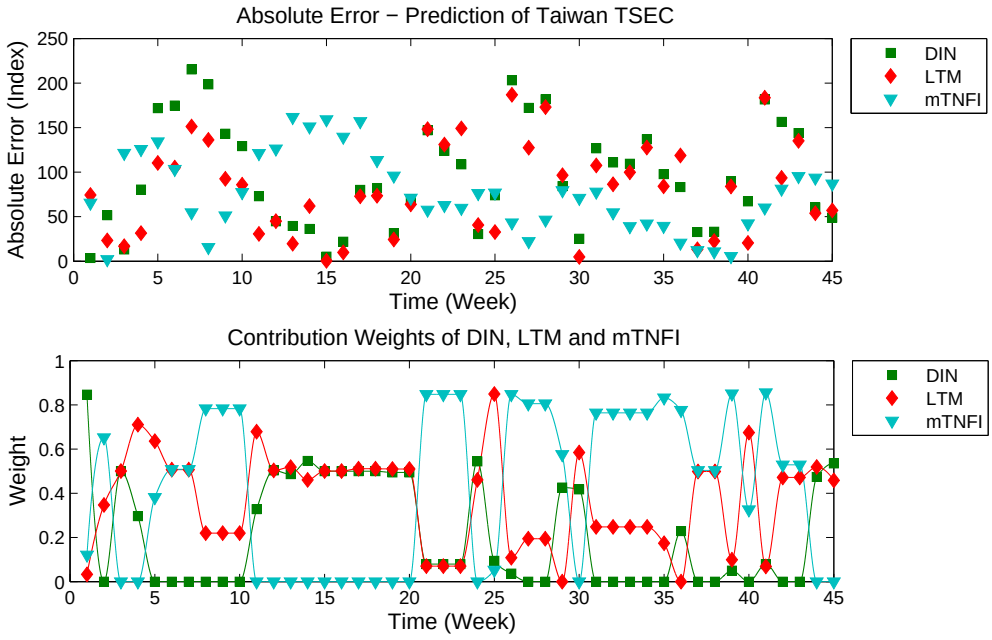


(b) Assigned contributing weights for prediction of SSX

Figure 7.9. Assigned contributing weights for prediction of New Zealand and Shanghai China stock market indexes.



(a) Assigned contributing weights for prediction of STI



(b) Assigned contributing weights for prediction of TSEC

Figure 7.10. Assigned contributing weights for prediction of Singapore and Taiwan stock market index.

Table 7.2

Comparison of the IMMF RMSE prediction error rates against single time-series prediction methods: MLR, MLP, and random walk model.

No	Stock Market	IMMF	MLR	MLP	Random Walk
1	AORD	25.1345	171.7993	88.5213	85.7106
2	HSI	156.3394	460.3831	428.6957	320.0959
3	JSX	19.8513	77.3591	41.7204	47.5735
4	KLSE	5.2856	36.4490	26.6499	22.5799
5	KOSPI	13.5639	47.6204	28.5997	27.3852
6	Nikkei 225	61.8075	311.8844	293.4835	220.9357
7	NZ50	8.8974	126.1689	48.9904	46.8238
8	SSX	30.7801	175.5635	105.7773	113.7639
9	STI	16.1047	75.2957	60.7475	57.2612
10	TSEC	57.3583	198.6822	141.3183	108.3788

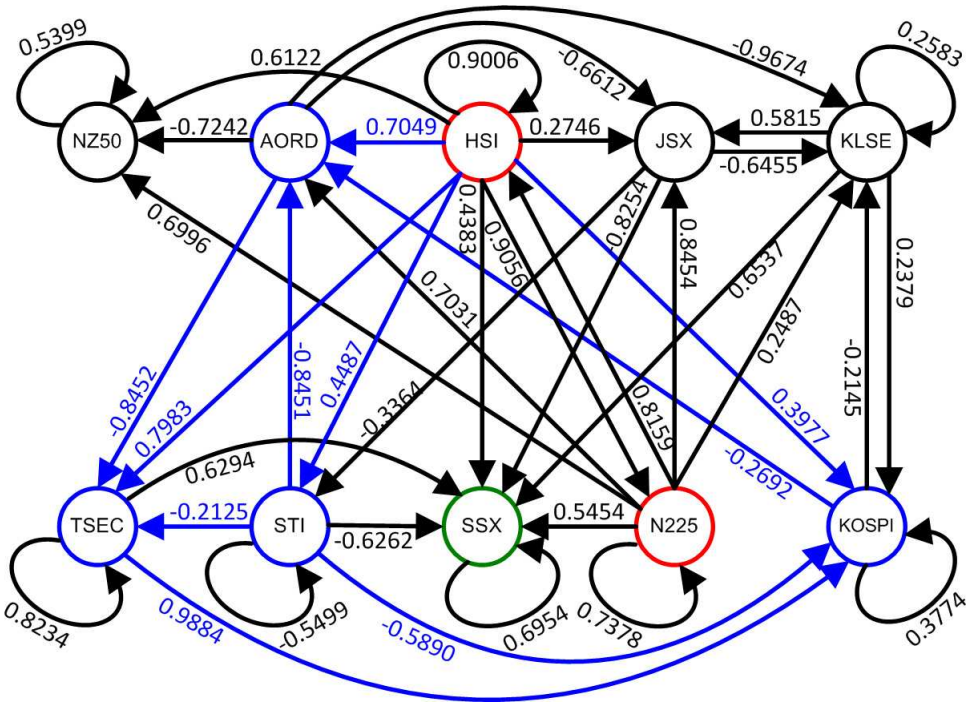
multiple variables in prediction when the variables concerned are influencing each other in a dynamic fashion.

7.5 Knowledge Discovery and Discussion

Figure 7.11a and 7.11b presents the mathematical model of interaction between the 10 stock markets under investigation extracted from the training data set which cover the period of 110 weeks starting from July 2007 and the interaction network model which is the graphical representation of the mathematical model itself. The mathematical model of interaction between the 10 stock markets was constructed from the transition matrix discovered during the learning process of DIN. The same mathematical model of interaction was also used to predict index values of the 10 stock markets for time-point 111,

$$\begin{bmatrix} \text{AORD}_{t+1} \\ \text{HSI}_{t+1} \\ \text{JSX}_{t+1} \\ \text{KLSE}_{t+1} \\ \text{KOSPI}_{t+1} \\ \text{N225}_{t+1} \\ \text{NZ50}_{t+1} \\ \text{SSX}_{t+1} \\ \text{STI}_{t+1} \\ \text{TSEC}_{t+1} \end{bmatrix} = \begin{bmatrix} 0 & 0.70 & 0 & 0 & -0.27 & 0.70 & 0 & 0 & -0.85 & 0 \\ 0 & 0.90 & 0 & 0 & 0 & 0.82 & 0 & 0 & 0 & 0 \\ -0.66 & 0.27 & 0 & 0.58 & 0 & 0.85 & 0 & 0 & 0 & 0 \\ -0.97 & 0 & -0.65 & 0.26 & -0.21 & 0.25 & 0 & 0 & 0 & 0 \\ 0 & 0.40 & 0 & 0.24 & 0.38 & 0 & 0 & 0 & -0.59 & 0.99 \\ 0 & 0.91 & 0 & 0 & 0 & 0.74 & 0 & 0 & 0 & 0 \\ -0.72 & 0.61 & 0 & 0 & 0 & 0.70 & 0.54 & 0 & 0 & 0 \\ 0 & 0.44 & -0.83 & 0.65 & 0 & 0.55 & 0 & 0.70 & -0.63 & 0.63 \\ 0 & 0.45 & -0.34 & 0 & 0 & 0 & 0 & 0 & -0.55 & 0 \\ -0.85 & 0.80 & 0 & 0 & 0 & 0 & 0 & 0 & -0.21 & 0.82 \end{bmatrix} \begin{bmatrix} \text{AORD}_t \\ \text{HSI}_t \\ \text{JSX}_t \\ \text{KLSE}_t \\ \text{KOSPI}_t \\ \text{N225}_t \\ \text{NZ50}_t \\ \text{SSX}_t \\ \text{STI}_t \\ \text{TSEC}_t \end{bmatrix}$$

(a) Mathematical model of interaction between multiple time-series



(b) Interaction network model

Figure 7.11. Mathematical model of interaction and its interaction network model between the 10 stock markets obtained from the training data set for the period of 110 weeks.

which is the first point of the test data set.

The interaction network model (as the graphical representation of the mathematical model of interaction between the 10 stock markets) extracted from the training data (as illustrated in Figure 7.11) shows how the 10 stock markets in the Asia Pacific are connected to each other exposing both posi-

tive and negative influences. This network model indicates that there is not a single market that moves by itself, not being affected by the conditions of any other markets in the region. The network model suggests that index movement of a stock market is always either affecting or being affected by the other stock markets. This discovery supports findings from previous studies that analysed the existence of interdependencies in global stock markets (Chowdhury, 1994; Lucey & Muckley, 2010; Lukmanto, Widiputra, & Lukas, 2009; Shabri, Kameel, & Azmi, 2008).

The mathematical model of interaction between the 10 stock markets and its interaction network model in Figure 7.11 clearly shows how the two leading markets in the Asia Pacific, the Hong Kong HSI and Japan Nikkei 225, are affecting the other markets significantly. Interestingly, the interaction network model also shows that movement of the Shanghai China SSX is affected by most of the other markets in the region as shown by this equation below which is derived from the mathematical model in Figure 7.11a:

$$\begin{aligned} \mathrm{SSX}_{t+1} = \; & 0.44\mathrm{HSI}_t - 0.83\mathrm{JSX}_t + 0.65\mathrm{KLSE}_t + 0.55\mathrm{N225}_t + 0.70\mathrm{SSX}_t \\ & -0.63\mathrm{STI}_t + 0.63\mathrm{TSEC}_t \end{aligned}$$

This finding is in agreement with results from a previous study by Widiputra et al. which extracted the dynamics of interaction of stock markets in the Asia Pacific region during the period of February 2006 to November 2008 (Widiputra, Pears, Serguieva, & Kasabov, 2009). The study outlined that the explanation to the discovery has a strong relation with China's strategic approach to its Asian neighbours as follows:

1. China has become one of the largest traders and investors with many Asian countries;

2. China exports primarily consumer goods to most countries in the Asia Pacific;

3. China is more than just a trading partner, as it also invests extensively in the region (e.g. China recently became one of the largest investors in Indonesia buying into oil and gas interests);

4. China is the largest foreign investor in some of the smaller economies in the South East Asia.

By proposing to negotiate a free trade agreement with the ASEAN (Association of South East Asia Nations) countries, China offered to share the benefits of its economic growth, while reminding the region of their growing reliance on China. All these contribute to the unique position of China, and to the number of vertices involved in the interaction network diagram.

Furthermore, the network model also identifies the existence of interaction between Australia AORD, Hong Kong HSI, South Korea KOSPI, Singapore STI, and Taiwan TSEC, which is in agreement with previous findings by Masih and Masih (2001) in their research on the dynamics of stock market interdependency in 1998 which found that the 5 stock markets are interdependent with each other. These equations, derived from the mathematical model of interaction between the 10 stock markets as shown in Figure 7.11a, support this finding as well:

$$\begin{aligned}
\mathrm{AORD}_{t+1} &= 0.70\mathrm{HSI}_t - 0.27\mathrm{KOSPI}_t + 0.70\mathrm{N225}_t - 0.85\mathrm{STI}_t \\
\mathrm{HSI}_{t+1} &= 0.90\mathrm{HSI}_t + 0.82\mathrm{N225}_t \\
\mathrm{KOSPI}_{t+1} &= 0.40\mathrm{HSI}_t + 0.24\mathrm{KLSE}_t + 0.38\mathrm{KOSPI}_t - 0.59\mathrm{STI}_t \\
&\quad +0.99\mathrm{TSEC}_t \\
\mathrm{STI}_t &= 0.45\mathrm{HSI}_t - 0.34\mathrm{JSX}_t 0.55\mathrm{STI}_t \\
\mathrm{TSEC}_t &= -0.85\mathrm{AORD}_t + 0.80\mathrm{HSI}_t - 0.21\mathrm{STI}_t + 0.82\mathrm{TSEC}_t
\end{aligned}$$

Though their results are based on the period from 1982 to 1994, the outcome of the DIN modelling advises that the relationships among these countries persist in the more recent period. Therefore, for some of the leading economies with relatively stable economic infrastructure, consistent relationships exist

between their stock markets (Widiputra et al., 2011a; Widiputra, Pears, Serguieva, & Kasabov, 2009).

Throughout the testing period of 45 weeks, the interaction network evolves as new observations on new behaviour of interaction become available over time. Consequently, a new mathematical model of interaction between the 10 stock markets and a new structure of the interaction network were obtained at the end of the testing period as illustrated in Figure 7.12a and 7.12b. The mathematical model of the interaction and the interaction network model obtained at the end of the testing period reveals that the structure of interdependencies between markets has changed.

However, it is also observable that some markets retain their connections with the other markets even though the level of interaction (denoted by the weight of the connection between vertices) is changing. For instance, it is clear that Hong Kong HSI and Japan Nikkei 225 are still the most influential markets in the region, whilst Shangai China SSX is still the most affected market as shown by this equation below,

$$\begin{aligned} \mathrm{SSX}_{t+1} = \; & 0.31\mathrm{HSI}_t - 0.30\mathrm{JSX}_t + 0.59\mathrm{KLSE}_t - 0.30\mathrm{KOSPI}_t - 0.44\mathrm{N225}_t \\ & +0.31\mathrm{NZ50}_t + 0.58\mathrm{SSX}_t - 0.40\mathrm{STI}_t + 0.25\mathrm{TSEC}_t, \end{aligned}$$

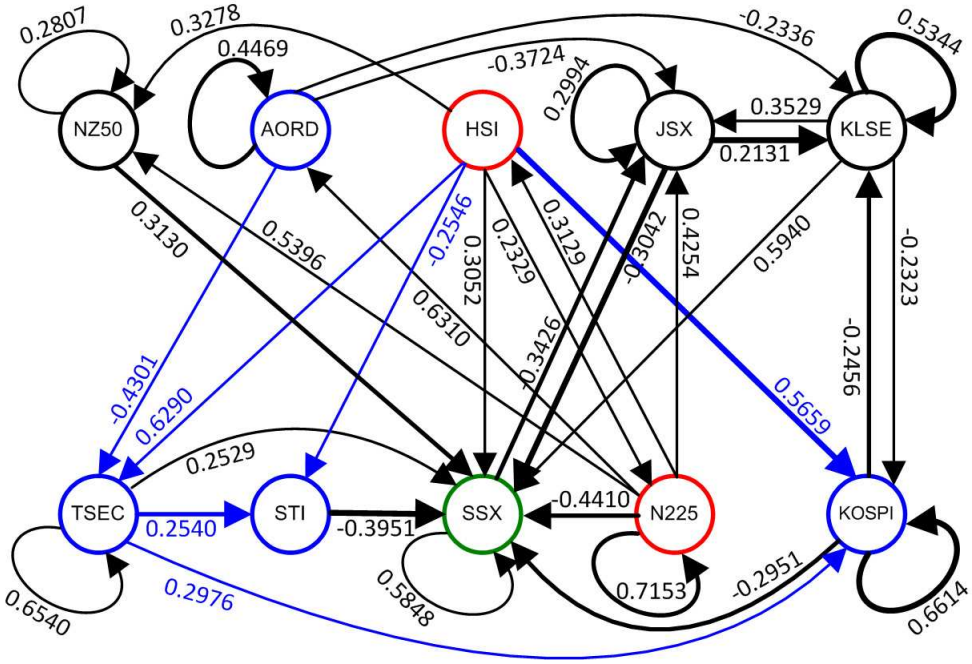
and that Australia, Hong Kong, South Korea, Singapore and Taiwan maintain the existence of interaction between them as it is confirmed by these equations:

$$\begin{aligned} \mathrm{AORD}_{t+1} &= 0.45\mathrm{AORD}_t + 0.63\mathrm{N225}_t \\ \mathrm{HSI}_{t+1} &= 0.31\mathrm{N225}_t \\ \mathrm{KOSPI}_{t+1} &= 0.57\mathrm{HSI}_t - 0.23\mathrm{KLSE}_t + 0.66\mathrm{KOSPI}_t + 0.30\mathrm{TSEC}_t \\ \mathrm{STI}_t &= -0.25\mathrm{HSI}_t + 0.25\mathrm{TSEC}_t \\ \mathrm{TSEC}_t &= -0.43\mathrm{AORD}_t + 0.63\mathrm{HSI}_t + 0.65\mathrm{TSEC}_t \end{aligned}$$

As it has been stated previously, this finding yet again suggests that even though the pattern of interaction between globalised stock markets is changing dynamically over time, consistent relationships exist between the stock

$$\begin{bmatrix} \text{AORD}_{t+1} \\ \text{HSI}_{t+1} \\ \text{JSX}_{t+1} \\ \text{KLSE}_{t+1} \\ \text{KOSPI}_{t+1} \\ \text{N225}_{t+1} \\ \text{NZ50}_{t+1} \\ \text{SSX}_{t+1} \\ \text{STI}_{t+1} \\ \text{TSEC}_{t+1} \end{bmatrix} = \begin{bmatrix} 0.45 & 0 & 0 & 0 & 0 & 0.63 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.31 & 0 & 0 & 0 & 0 \\ -0.37 & 0 & 0.30 & 0.35 & 0 & 0.43 & 0 & -0.30 & 0 & 0 \\ -0.23 & 0 & 0.21 & 0.53 & -0.25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.57 & 0 & -0.23 & 0.66 & 0 & 0 & 0 & 0 & 0.30 \\ 0 & 0.23 & 0 & 0 & 0 & 0.72 & 0 & 0 & 0 & 0 \\ 0 & 0.33 & 0 & 0 & 0 & 0.54 & 0.28 & 0 & 0 & 0 \\ 0 & 0.31 & -0.30 & 0.59 & -0.30 & -0.44 & 0.31 & 0.58 & -0.40 & 0.25 \\ 0 & -0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 \\ -0.43 & 0.63 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.65 \end{bmatrix} \begin{bmatrix} \text{AORD}_t \\ \text{HSI}_t \\ \text{JSX}_t \\ \text{KLSE}_t \\ \text{KOSPI}_t \\ \text{N225}_t \\ \text{NZ50}_t \\ \text{SSX}_t \\ \text{STI}_t \\ \text{TSEC}_t \end{bmatrix}$$

(a) Mathematical model of interaction between multiple time-series



(b) Interaction network model

Figure 7.12. Mathematical model of interaction and its interaction network model between the 10 stock markets obtained from the training data set for the period of 110 weeks.

markets of countries with relatively stable economic conditions.

Figure 7.13 illustrates the state of the LTM knowledge repository after 110 points (that is 110 weeks) of stock market indexes of the 10 selected stock markets in the Asia Pacific region are conferred to LTM. Figure 7.13 shows the existence of strong relationship between Australia AORD, Hong

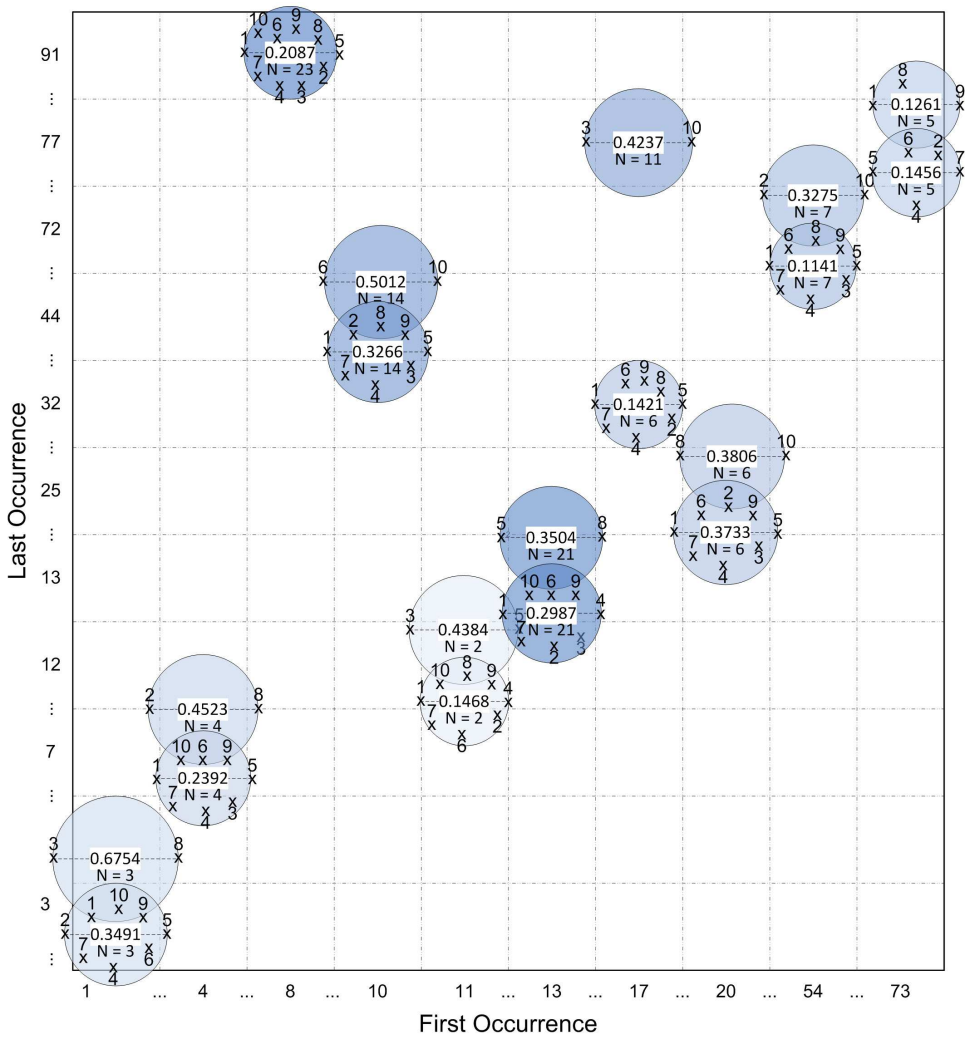


Figure 7.13. Profiles of relationships of LTM after 110 weekly indexes of 10 stock markets. The 10 stock markets are denoted as follows: NZ50 (1), AORD (2), HSI (3), JSX (4), KLSE (5), KOSPI (6), N225 (7), SSX (8), STI (9) and TSEC (10).

Kong HSI, South Korea KOSPI, Singapore STI, and Taiwan TSEC. Extracted profiles of relationships in the knowledge repository reveal that these five stock markets are grouped together frequently. This outcome is in agreement with the interaction network model produced by DIN (as explained previously) and again with a previous study (Masih & Masih, 2001).

In addition, extracted profiles of relationships of LTM reveal that the sig-

nificant interaction of Shanghai China SSX with the other markets is also being recognised. LTM confirms that SSX exists in almost all profiles of relationships with substantial number of participants. This outcome is comparable to results from previous work by Widiputra et al. (2011a). The discovery suggests that generally SSX has always been progressing with most of the other markets in the region. This result indicates that SSX maintained its level of interaction with the other markets in different periods of time localities. Even more, this finding might suggest that Shanghai China SSX has actually become an important "market player" in the Asia Pacific region, and that its index movement could highly influence the movement of the other stock markets in the region.

Additionally, in Figure 7.13 cluster diameter represents the farthest correlation between time-series in the same cluster while relative positioning of the labels indicates the degree of similarity in behaviour. For instance, in the cluster of NZ50, HSI, JSX, KLX, KOSPI, N225, STI, TSEC; South Korea KOSPI is positioned closer to New Zealand NZ50. This indicates that similarity between KOSPI and NZ50 is higher compared to similarity between KOSPI and the other markets. This initial result confirms LTM's capability to capture the existence of diverse profiles of relationships that exist in the globalised security markets.

Figure 7.14 illustrates the state of the knowledge repository after the whole data set of 155 weekly indexes of the 10 stock markets was entered in the system. It is observable that some new profiles of relationships have emerged in the repository and the existing profiles have been updated (in terms of the cluster diameter). This result confirms that LTM is able to perform dynamic learning of multiple time-series by capturing the dynamics of relationships between the series in different time localities.

As mTNFI constructs a specific solution for every input vector, different fuzzy inference systems are then constructed for input vectors corresponding

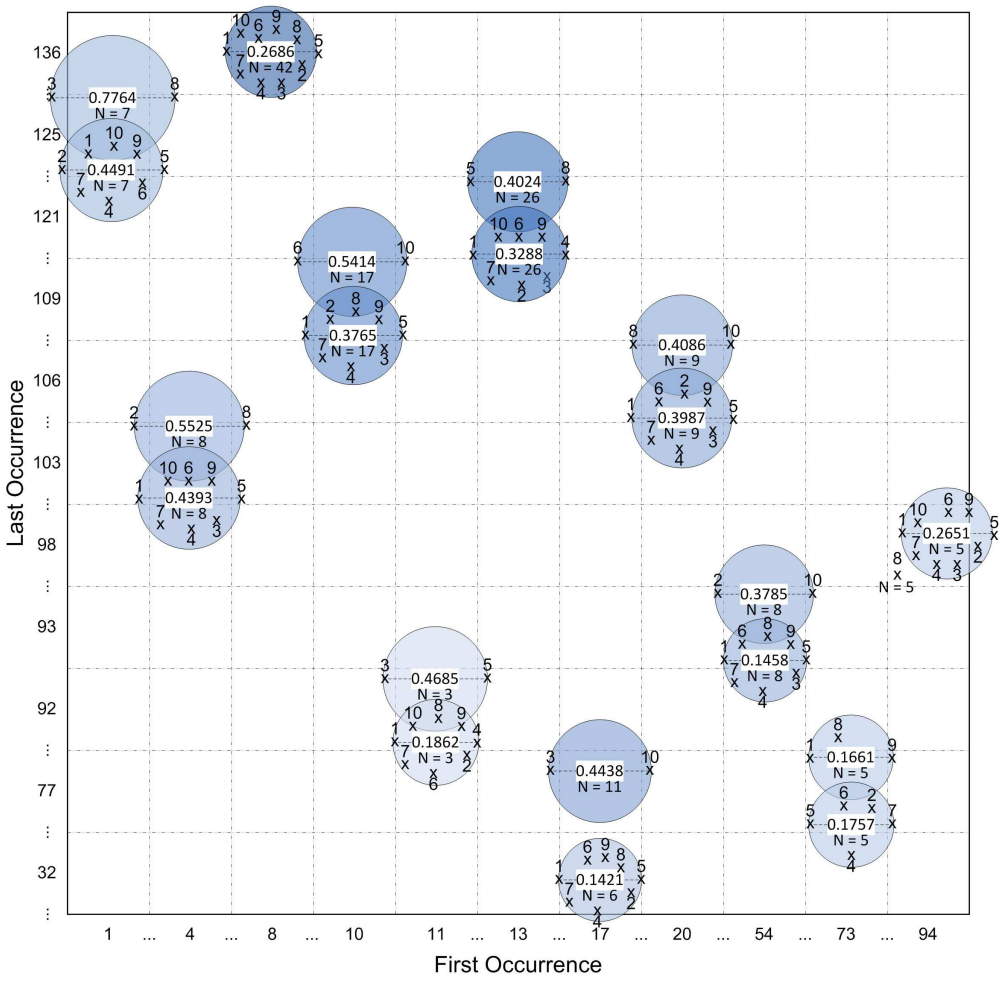


Figure 7.14. Profiles of relationships of LTM after 155 weekly indexes of 10 stock markets. The 10 stock markets are denoted as follows: NZ50 (1), AORD (2), HSI (3), JSX (4), KLSE (5), KOSPI (6), N225 (7), SSX (8), STI (9) and TSEC (10).

to different time points. Figure 7.15 outlines the fuzzy rules created by mTNFI using the 80 nearest samples from the 110 weeks training data set when calculating output for input vector at time-point 111. Throughout the experimentation, the number of selected nearest samples in mTNFI is set by hand (based on experience) to a fixed value of 80 samples. Figure 7.15 outlines only 3 fuzzy rules from the complete set of 6 created rules, that help to build the inference system for the prediction of stock market indexes at time-point 111 of the evaluation period.

Rule1

If RatioOfDiff NZ50(t)-AORD(t) is GaussianMF (1.74 0.23)
RatioOfDiff NZ50(t)-HSI(t) is GaussianMF (2.33 0.21)
. . .
RatioOfDiff TSEC(t)-STI(t) is GaussianMF (0.59 0.19)

Then NZ50(t+1) = 1.14*NZ50(t) – 0.20*AORD(t) + 0.03*HSI(t) – 0.04*JSX(t) – 0.01*KLSE(t)
–0.28*KOSPI(t) + 0.04*N225(t) + 0.01*SSX(t) – 0.16*STI(t) – 0.05*TSEC(t)
AORD(t+1) = 0.32*NZ50(t) + 0.63*AORD(t) + 0.07*HSI(t) – 0.10*JSX(t) – 0.42*KLSE(t)
–0.40*KOSPI(t) + 0.06*N225(t) + 0.06*SSX(t) – 0.40*STI(t) + 0.12*TSEC(t)
. . .
TSEC(t+1) = 0.40*NZ50(t) – 0.39*AORD(t) + 0.09*HSI(t) + 0.04*JSX(t) – 0.49*KLSE(t)
–0.36*KOSPI(t) + 0.08*N225(t) + 0.17*SSX(t) – 1.00*STI(t) + 1.14*TSEC(t)

Rule 2

If RatioOfDiff NZ50(t)-AORD(t) is GaussianMF (2.46 0.19)
RatioOfDiff NZ50(t)-HSI(t) is GaussianMF (2.47 0.24)
. . .
RatioOfDiff TSEC(t)-STI(t) is GaussianMF (0.70 0.13)

Then NZ50(t+1) = 1.12*NZ50(t) – 0.10*AORD(t) – 0.01*HSI(t) – 0.01*JSX(t) – 0.00*KLSE(t)
–0.46*KOSPI(t) + 0.05*N225(t) – 0.01*SSX(t) + 0.52*STI(t) + 0.02*TSEC(t)
AORD(t+1) = 0.25*NZ50(t) + 0.80*AORD(t) + 0.02*HSI(t) – 0.15*JSX(t) + 0.03*KLSE(t)
–0.90*KOSPI(t) – 0.07*N225(t) – 0.01*SSX(t) + 0.61*STI(t) + 0.10*TSEC(t)
. . .
TSEC(t+1) = 0.15*NZ50(t) – 0.04*AORD(t) – 0.05*HSI(t) – 0.25*JSX(t) + 1.16*KLSE(t)
–1.87*KOSPI(t) – 0.19*N225(t) – 0.00*SSX(t) + 1.48*STI(t) + 1.14*TSEC(t)

Rule 3

If RatioOfDiff NZ50(t)-AORD(t) is GaussianMF (1.58 0.14)
RatioOfDiff NZ50(t)-HSI(t) is GaussianMF (2.74 0.27)
. . .
RatioOfDiff TSEC-STI is GaussianMF (0.70 0.19)

Then NZ50(t+1) = 1.14*NZ50(t) – 0.20*AORD(t) + 0.03*HSI(t) – 0.08*JSX(t) – 0.03*KLSE(t)
–0.24*KOSPI(t) + 0.02*N225(t) + 0.01*SSX(t) – 0.03*STI(t) + 0.04*TSEC(t)
AORD(t+1) = 0.28*NZ50(t) + 0.61*AORD(t) + 0.07*HSI(t) – 0.23*JSX(t) – 0.08*KLSE(t)
–0.64*KOSPI(t) + 0.05*N225(t) + 0.03*SSX(t) – 0.20*STI(t) + 0.13*TSEC(t)
. . .
TSEC(t+1) = 0.14*NZ50(t) – 0.35*AORD(t) + 0.05*HSI(t) – 0.38*JSX(t) + 1.11*KLSE(t)
–1.07*KOSPI(t) + 0.06*N225(t) + 0.08*SSX(t) – 0.37*STI(t) + 1.19*TSEC(t)

Figure 7.15. Extracted fuzzy rules (first-order Takagi-Sugeno type) from the Multivariate Transductive Neuro-Fuzzy Inference system (mTNFI) when predicting upcoming indexes of 10 stock markets in the Asia Pacific based on data from week 111.

However, when estimating the output for input vectors at different time moments mTNFI reconstructs new fuzzy inference systems consisting of different set of fuzzy rules. For instance, Figure 7.16 represents 2 fuzzy rules (from the complete set of another 6 created rules) that took part in constructing the new fuzzy inference system when estimating the output for input vector

Rule1

If RatioOfDiff NZ50(t)-AORD(t) is GaussianMF (1.24 0.19)
RatioOfDiff NZ50(t)-HSI(t) is GaussianMF (0.51 0.14)
. . .
RatioOfDiff TSEC(t)-STI(t) is GaussianMF (4.87 0.23)

Then NZ50(t+1) = 1.15*NZ50(t) – 0.16*AORD(t) + 0.04*HSI(t) + 0.05*JSX(t) – 0.24*KLSE(t)
–0.12*KOSPI(t) + 0.01*N225(t) + 0.00*SSX(t) – 0.21*STI(t) + 0.03*TSEC(t)
AORD(t+1) = 0.36*NZ50(t) + 0.74*AORD(t) + 0.09*HSI(t) + 0.15*JSX(t) – 0.02*KLSE(t)
–0.01*KOSPI(t) – 0.01*N225(t) + 0.03*SSX(t) – 0.52*STI(t) + 0.06*TSEC(t)
. . .
TSEC(t+1) = 0.44*NZ50(t) – 0.26*AORD(t) + 0.07*HSI(t) + 0.31*JSX(t) – 1.55*KLSE(t)
+0.40*KOSPI(t) + 0.01*N225(t) + 0.20*SSX(t) – 0.87*STI(t) + 1.05*TSEC(t)

Rule 2

If RatioOfDiff NZ50(t)-AORD(t) is GaussianMF (1.45 0.12)
RatioOfDiff NZ50(t)-HSI(t) is GaussianMF (0.55 0.27)
. . .
RatioOfDiff TSEC(t)-STI(t) is GaussianMF (4.48 0.19)

Then NZ50(t+1) = 1.12*NZ50(t) – 0.10*AORD(t) – 0.00*HSI(t) – 0.01*JSX(t) + 0.03*KLSE(t)
–0.60*KOSPI(t) – 0.06*N225(t) – 0.02*SSX(t) + 0.59*STI(t) + 0.03*TSEC(t)
AORD(t+1) = 0.26*NZ50(t) + 0.85*AORD(t) – 0.00*HSI(t) – 0.13*JSX(t) – 0.03*KLSE(t)
–0.81*KOSPI(t) – 0.12*N225(t) – 0.02*SSX(t) + 0.91*STI(t) + 0.07*TSEC(t)
. . .
TSEC(t+1) = 0.10*NZ50(t) + 0.16*AORD(t) – 0.09*HSI(t) – 0.14*JSX(t) + 0.95*KLSE(t)
–2.24*KOSPI(t) – 0.30*N225(t) – 0.03*SSX(t) + 2.33*STI(t) + 1.11*TSEC(t)

Figure 7.16. Extracted fuzzy rules (first-order Takagi-Sugeno type) from the Multivariate Transductive Neuro-Fuzzy Inference system (mTNFI) when predicting upcoming indexes of 10 stock markets in the Asia Pacific on week 120.

at time-point 120. These observations indicate mTNFI's capability to dynamically construct specific individual local models for each new input vector produced by the stream of multiple time-series under evaluation.

By constructing different inference systems for every new input vector, the mTNFI is expected to be able to predict simultaneously the movement of multiple time-series of a non-stationary environment with a reasonable degree of accuracy. This expectation has been realised in this study based on outcomes of conducted experiments outlined in Section 7.4 (see Table 7.1).

7.6 Conclusion

This chapter presents an application of the global (DIN), local (LTM), transductive model (mTNFI) and the integrated framework of the three models (IMMF) for analysis, modelling and prediction of 10 interactive stock markets in the Asia Pacific region. The selected stock markets are: AORD of Australia, HSI of Hong Kong, JSX of Indonesia, KLSE of Malaysia, KOSPI of South Korea, Nikkei 225 of Japan, NZ50 of New Zealand, SSX of Shanghai China, STI of Singapore and TSEC of Taiwan. The data used in the experiments conducted in the study are the aggregate weekly index of the stock markets, spanning 155 weeks from July 2007 to July 2010.

Results of conducted experiments show that the three models of multiple time-series prediction produced good prediction accuracies. It is also found that there is no single model out of the global, local and transductive model that always performs the best throughout the period of examination. Outcomes of experiments also reveal that generally, the integrated framework which assimilates predictions from the global (DIN), local (LTM) and transductive model (mTNFI) has effectively delivered the best accuracy across the whole period of examination. This was accomplished by dynamically changing the contributing weights assigned to each model based on their prior performance in predicting movement of the stock market indexes.

A comparative analysis with methods applied on single time-series prediction suggests that predicting movement of multiple interactive stock markets by considering the interactions and profiles of relationships between them results in a more accurate prediction. Overall, the proposed global (DIN), local (LTM), and transductive (mTNFI) models of multiple time-series analysis and the proposed integrated framework (IMMF) have several advantages compared to MLR, MLP and random walk model, that are outlined below:

- DIN, LTM and mTNFI predict movement of multiple time-series si-

multaneously by taking into account the interactions between variables globally (DIN), in different localities of time (LTM) or at a specific time moment (mtNFI);

- The three models adapt to new data and are capable to evolve their structure (DIN and LTM) or to construct a new local model for every new input vector (mTNFI). Additionally, the integrated framework of DIN, LTM and mTNFI is able to dynamically assign contributing weights to each model based on their prior prediction performance;

- Based on the two main characteristics listed above, DIN, LTM, mTNFI and IMMF are more accurate time-series prediction methods in comparison with MLR, MLP and random walk.

Chapter 8

Case Study 2: Analysis and Modelling of New Zealand's Weather Condition

8.1 Introduction

The word "weather" can be defined as the state of the atmosphere when measured on a scale of hot or cold, wet or dry, calm or stormy, clear or cloudy. Different from climate, which is the term for the average atmospheric conditions over longer periods of time, weather generally refers to day-to-day temperature, air pressure, wind movement (speed and direction) and precipitation activities. Weather occurs due to density (in terms of temperature and moisture) differences between one place and another. These differences can occur due to the sun angle at any particular spot, which varies by latitude from the tropics. The strong temperature contrast between polar and tropical air gives rise to the jet stream.

One of the most important factors that governs or has the greatest influence on circulating the weather systems is the wind, as it directs movement of the air in a certain velocity. Wind is caused by differences in air pressure levels between locations. When a difference in pressure exists, the air is accelerated from higher to lower pressure. On a rotating planet the air will be deflected by the *Coriolis effect* (Barry & Chorley, 2003; Lofting, Dougherty, &

Southwood, 1997; Trefil, 2003), except exactly on the equator. Globally, the two major driving factors of large scale winds (the atmospheric circulation) are the differential heating between the equator and the poles (difference in absorption of solar energy leading to buoyancy forces) and the rotation of the planet.

The atmosphere or the weather system is a dynamic system, therefore small changes to one part of the system can grow to have large effects on the system as a whole. This makes it difficult to accurately predict weather more than a few days in advance, though weather forecasters are continually working to extend this limit through the scientific study of weather, meteorology. It is theoretically impossible to make useful day-to-day predictions more than about two weeks ahead, imposing an upper limit to the potential for improved prediction (Barry & Chorley, 2003; Mailier, 2010).

Based on the fact outlined above, it would be of interest to observe variables of the atmosphere in order to extract and model how they are related to each other and how they will behave as a system. Therefore, in this chapter a weather data set collected from various sites in New Zealand is utilised as another case study to examine the performance of DIN, LTM, mTNFI and IMMF for the analysis, modelling and prediction of multiple time-series data.

8.2 Analysis and Modelling of Air Pressure Level

The main objective of this chapter is to present the application of DIN, LTM, mTNFI and IMMF for the analysis and modelling or air pressure levels in different locations collected on a daily basis. Additionally, the chapter evaluates the performance of DIN, LTM, mTNFI and IMMF when utilised for the prediction of movement of air pressure levels in different geographical locations. Finally, an analysis is carried out of the dynamics of interaction, profiles of

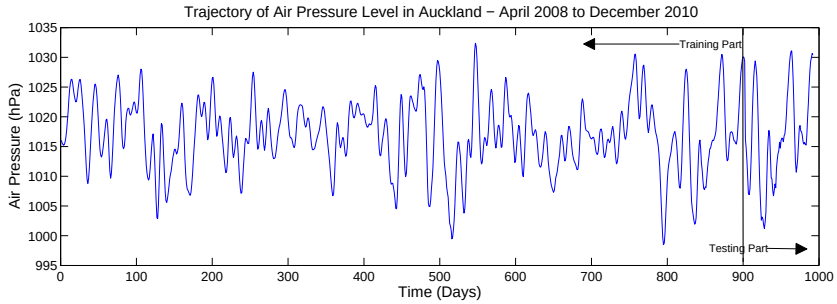
relationships in different time localities and the recurring trends of movement between air pressure levels collected at various sites.

8.3 Data Description

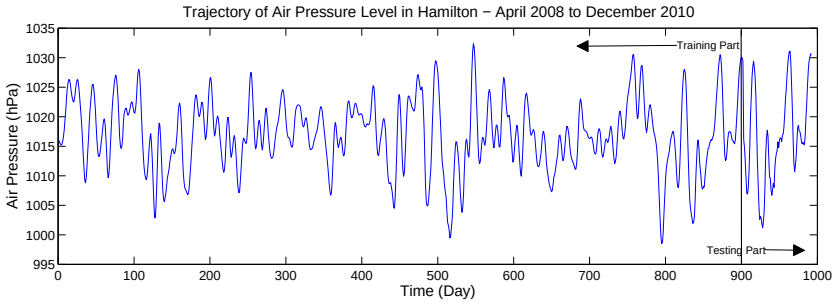
Daily air pressure data collected from various locations in New Zealand by the National Institute of Weather and Atmosphere (National Insitute of Weather and Atmosphere, 2010) constitutes the data set used in this study. The data covers a period of almost three years, ranging from the beginning of April 2008 to the end of December 2010. Findings from a previous study on global weather systems that argued that small changes to one part of the system led to a complete change in the weather system as a whole (Vitousek, 1992, 1994)) was the key motivation for experimenting with such data. The spatial coordinate was used to define the multiple variables, with the air pressure at four different locations in New Zealand (Auckland, Hamilton, Paeroa and Reefton) comprising the multiple variables.

Trajectories of observed air pressure data are illustrated in Figure 8.1. It should be noted that even though in general the four trajectories of air pressure level look alike, a closer look would reveal that the first three trajectories are exposing a more similar shape compared to the last one. This is expected as the first three trajectories were collected by three observation stations that are considerably close to each other: Auckland, Hamilton and Paeroa (within a diameter of 100 kilometres, part of the North Island of New Zealand), while the last trajectory of air pressure data was collected in Reefton (in the South Island of New Zealand).

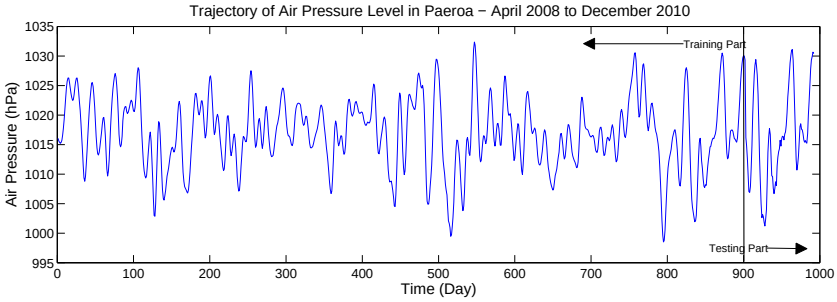
Throughout the conducted experiment the data set is divided into two different parts: the training part and the testing part as depicted in Figure 8.1. The experiment conducted in this work employs an incremental testing process, which means that whenever a new instance arrives a prediction is first



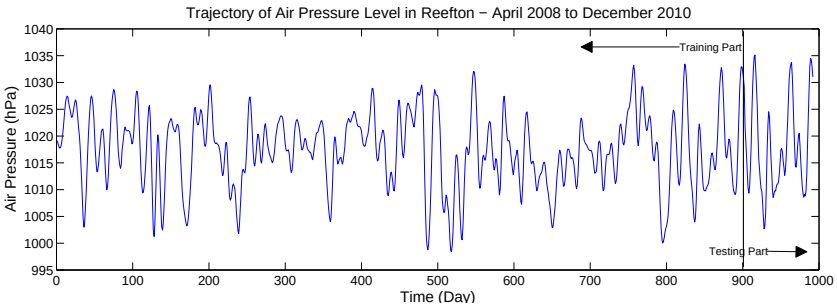
(a) Auckland



(b) Hamilton

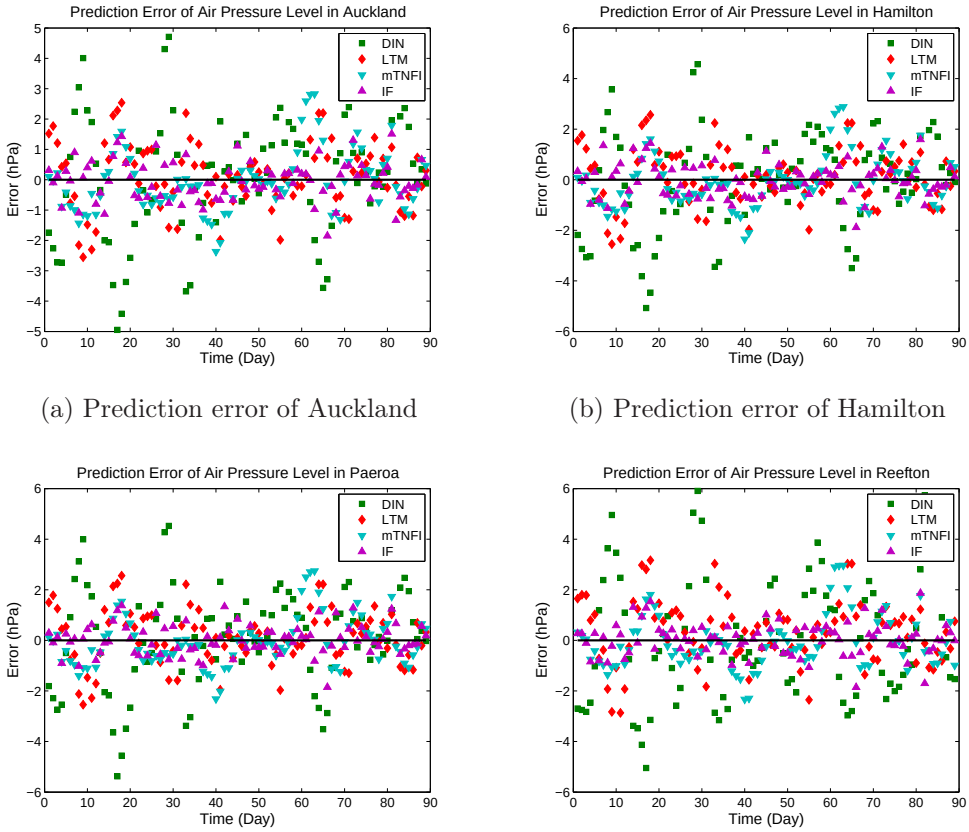


(c) Paeroa



(d) Reefton

Figure 8.1. Trajectories of air pressure level collected on a daily basis at four different locations in New Zealand.



(a) Prediction error of Auckland

(b) Prediction error of Hamilton

(c) Prediction error of Paeroa

(d) Prediction error of Reefton

Figure 8.2. Prediction error of Auckland, Paeroa, Hamilton and Reefton air pressure level.

made for that new instance before it is then added to the training set as additional training example.

8.4 Experimental Results and Comparison

Similarly to Chapter 7, where DIN, LTM, mTNFI and IMMF are applied for the analysis, modelling and prediction of multiple interactive stock markets in the Asia Pacific region, in this chapter scatter plots of error of prediction and RMSE are used to measure the quantitative performance of the proposed methods.

Table 8.1

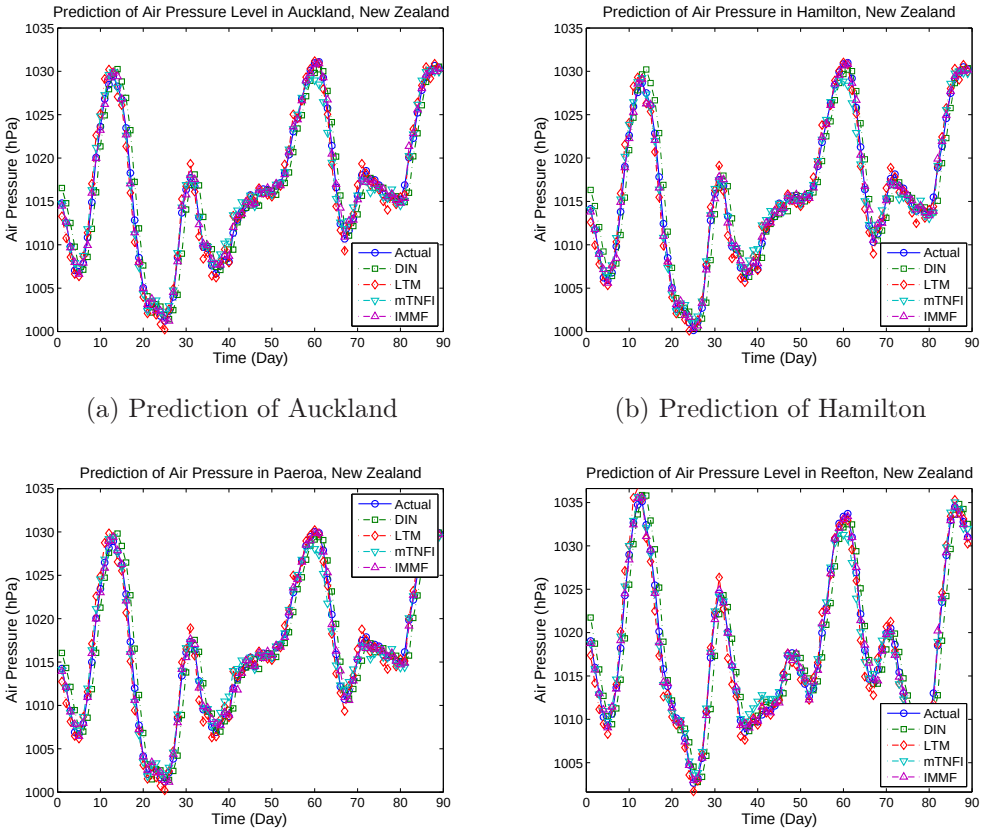
Error rates in RMSE of DIN, LTM, mTNFI and the IMMF when predicting movement of air pressure level in Auckland, Paeroa, Hamilton and Reefton.

No	Location	DIN	LTM	mTNFI	IMMF
1	Auckland	1.9874	1.1011	1.0077	0.6411
2	Paeroa	1.9723	1.0987	0.9758	0.5965
3	Hamilton	1.9602	1.1073	1.0164	0.6501
4	Reefton	2.4644	1.3063	1.0482	0.6453

The scatter plots clearly show that the prediction error of each model varies over time and there is no one single model that works the best all the time across the period of examination. However, the plots also show that the prediction errors produced by the IMMF during the testing period are generally distributed closer to 0 compared to the other models. This fact indicates that by assimilating predictions from DIN, LTM and mTNFI, the IMMF is capable of producing more accurate results in predicting movements of multiple time-series.

Plots of prediction trajectories produced by DIN, LTM, mTNFI and the IMMF as illustrated in Figure 8.3 show that the proposed global, local, transductive model and the integrated framework closely match the actual trajectories. This outcome is in agreement with results of the analysis and modelling of the multiple interactive stock markets outlined in Chapter 7, and therefore suggests that including the nature and strength of relationships into a prediction model leads to improving the accuracy of prediction of given variables involved in the system under observation.

Figures 8.4 and 8.5 illustrate how the contributing weights of DIN, LTM and mTNFI that are defined by the IMMF to calculate the final prediction of



(a) Prediction of Auckland

(b) Prediction of Hamilton

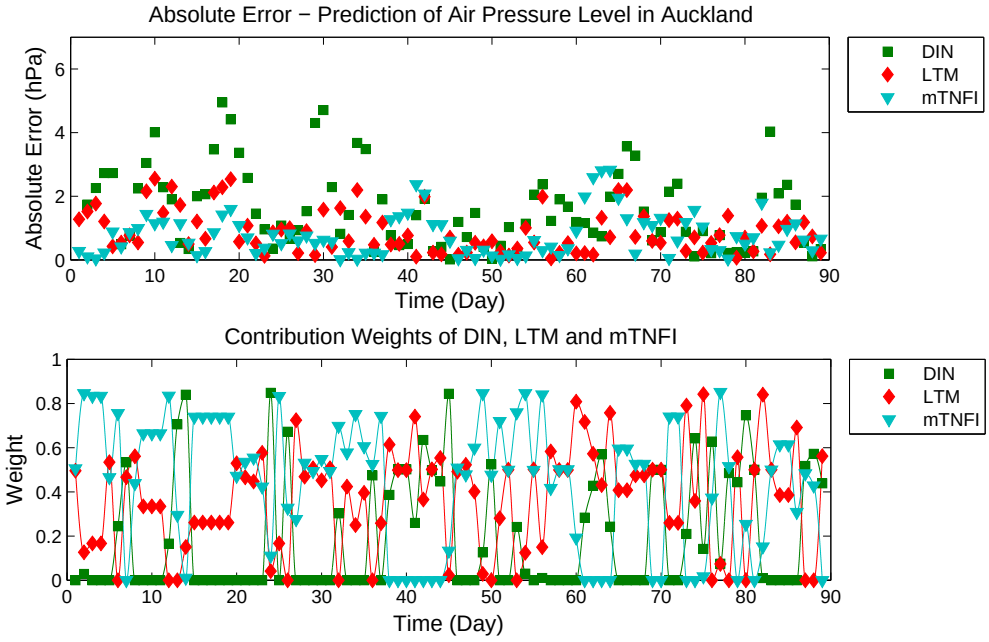
(c) Prediction of Paeroa

(d) Prediction of Reefton

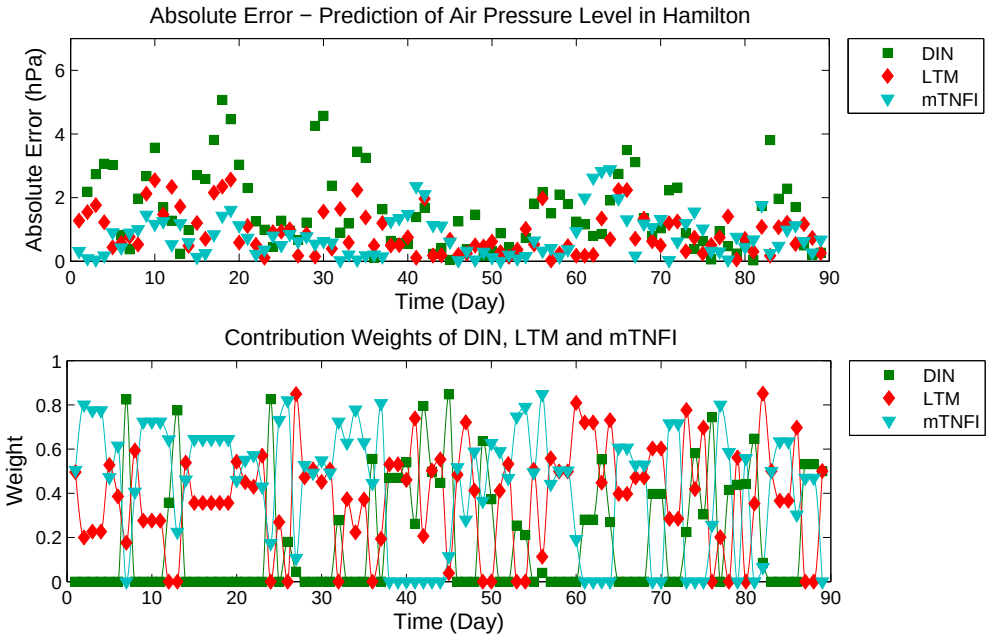
Figure 8.3. Prediction of Auckland, Hamilton, Paeroa and Reefton air pressure level.

air pressure levels in the four locations are changing dynamically in relation to the movement of the absolute error of each model. These plots indicate that the IMMF is able to adjust the level of trust given to each model (in the form of contributing weights) to maintain the level of its accuracy of prediction over time.

As a quantitative performance comparison, prediction of air pressure levels in the four observation locations using methods applied on single time-series prediction such as MLR, MLP and the random walk model is conducted and the outcomes are compared to results from the IMMF. Generally, error rates

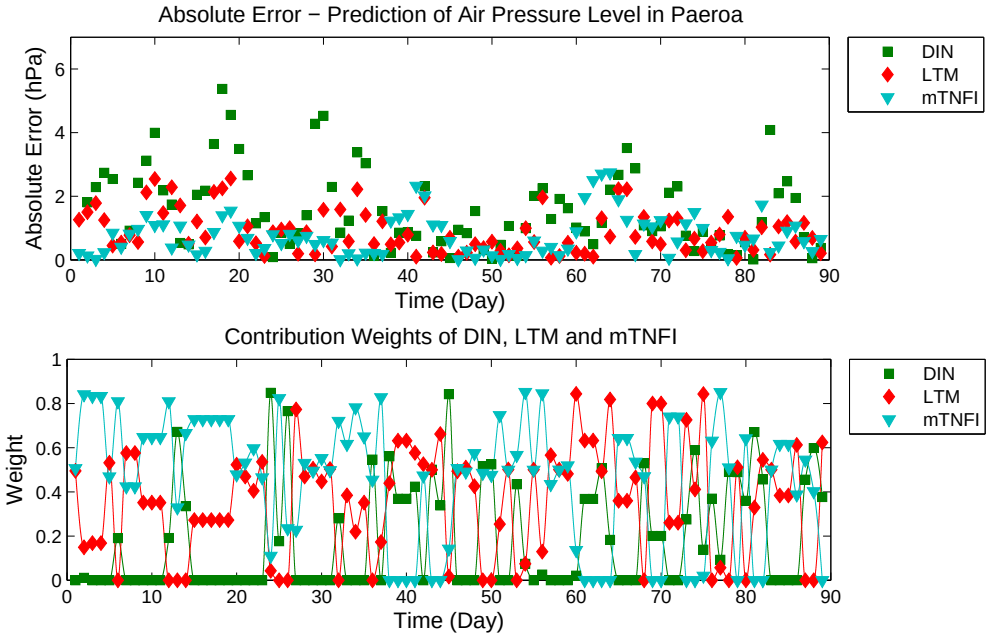


(a) Assigned contributing weights for prediction of Auckland

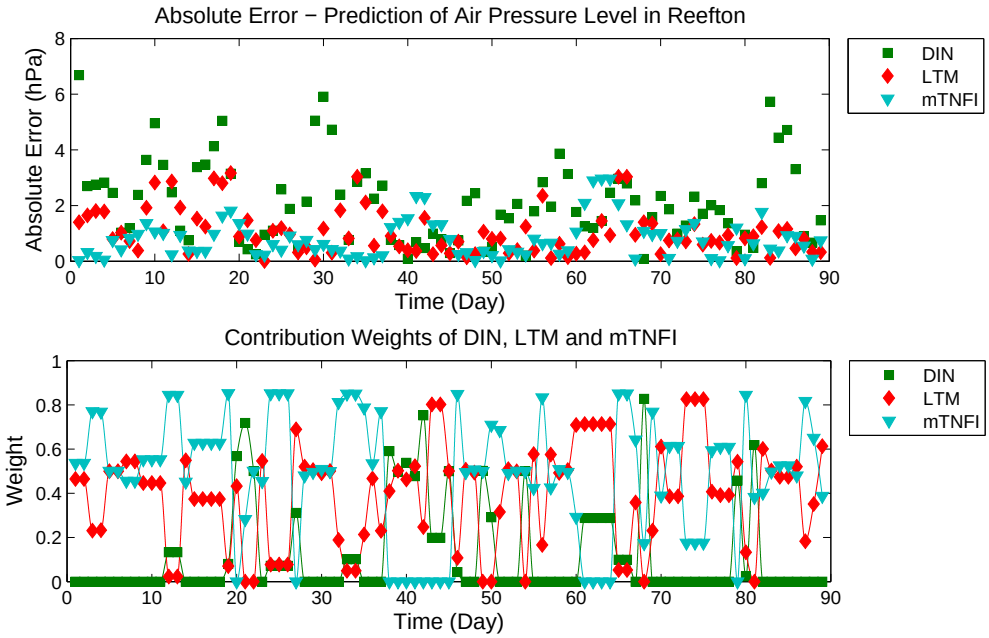


(b) Assigned contributing weights for prediction of Hamilton

Figure 8.4. Assigned contributing weights for prediction of Auckland and Hamilton air pressure level.



(a) Assigned contributing weights for prediction of Paeroa



(b) Assigned contributing weights for prediction of Reefton

Figure 8.5. Assigned contributing weights for prediction of Paeroa and Reefton air pressure.

Table 8.2

Comparison of the IMMF RMSE prediction error rates against methods applied on single time-series: MLR, MLP, and random walk model.

No	Location	IMMF	MLR	MLP	Random Walk
1	Auckland	0.6411	3.5236	3.0371	1.6340
2	Paeroa	0.5965	3.4257	3.1592	1.6868
3	Hamilton	0.6501	3.7263	3.4958	1.8316
4	Reefton	0.6453	4.1725	3.9125	2.4708

of prediction in RMSE outlined in Table 8.2 reveal that the proposed IMMF is superior to MLR, MLP and random walk model. Interestingly, random walk being the simplest model of all also outperforms MLR and MLP. The explanation to this outcome was discussed previously.

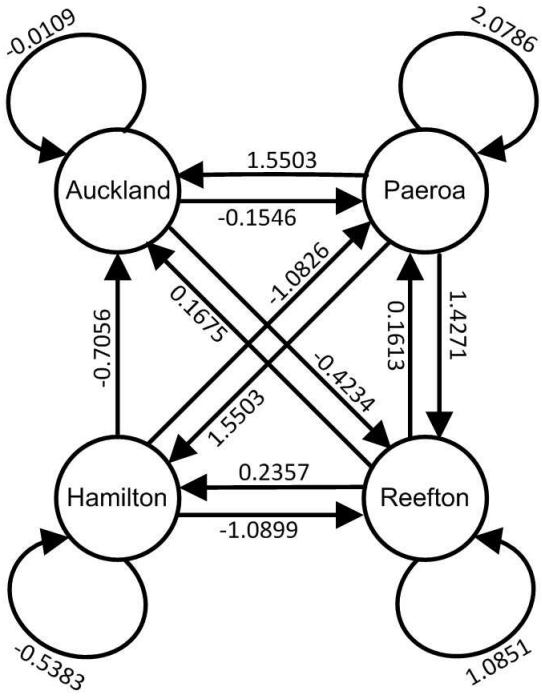
In Chapters 3 and 7 it was explained that the random walk model works simply by assuming that next value of a time-series is equal to its current value and therefore this model often gives better statistical results (in terms of sum squared of residual). However, this quantitative superiority is deceptive as the random walk model produces a shadow plot effect, and therefore despite of its capability to produce small RMSE in time-series prediction, it actually provides no predictive power.

8.5 Knowledge Discovery and Discussion

The initially extracted DIN model from the training data as illustrated in Figure 8.6 reveals that in the period of observation, there are significant interactions between air pressure levels in Auckland, Hamilton, Paeroa and Reefton. This outcome is in fact in agreement with the trajectories of air pressure level in these four locations (as depicted in Figure 8). The plots

$$\begin{bmatrix} \text{Auckland}_{t+1} \\ \text{Paeroa}_{t+1} \\ \text{Hamilton}_{t+1} \\ \text{Reefton}_{t+1} \end{bmatrix} = \begin{bmatrix} -0.0109 & 1.5503 & -0.7056 & 0.1675 \\ -0.1546 & 2.0786 & -1.0826 & 0.1613 \\ 0 & 1.5503 & -0.5383 & 0.2357 \\ -0.4234 & 1.4271 & -1.0899 & 1.0851 \end{bmatrix} \begin{bmatrix} \text{Auckland}_{t} \\ \text{Paeroa}_{t} \\ \text{Hamilton}_{t} \\ \text{Reefton}_{t} \end{bmatrix}$$

(a) Mathematical model of interaction between multiple time-series



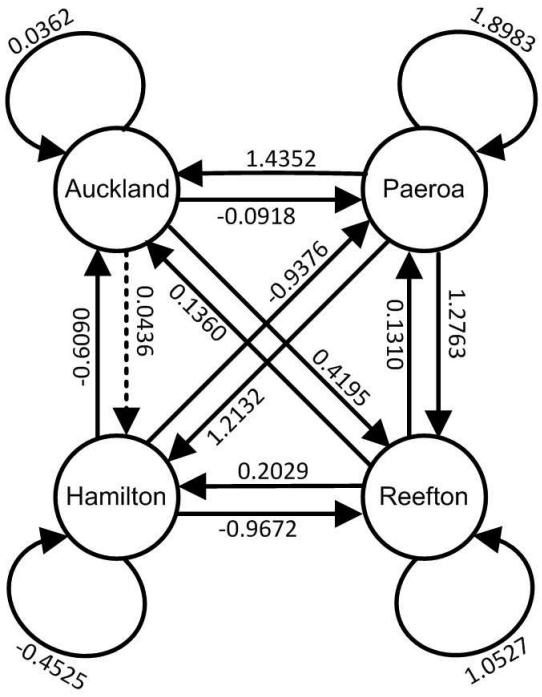
(b) Interaction network model

Figure 8.6. Mathematical model of interaction and its interaction network model between air pressure levels in Auckland, Paeroa, Hamilton and Reefton from the training data set.

of trajectories show that there is a similar shape of air pressure level movement in Auckland, Hamilton, Paeroa and Reefton during the period of evaluation. Therefore, having an interaction network model which describes the relationships between the four locations in a form of fully-connected graph is expected. Additionally, this finding also suggests that even though Reefton is located distantly from Auckland, Hamilton and Paeroa, in general movement

$$\begin{bmatrix} \text{Auckland}_{t+1} \\ \text{Paeroa}_{t+1} \\ \text{Hamilton}_{t+1} \\ \text{Reefton}_{t+1} \end{bmatrix} = \begin{bmatrix} 0.0362 & 1.4352 & -0.6090 & 0.1360 \\ -0.0918 & 1.8983 & -0.9376 & 0.1310 \\ 0.0436 & 1.2132 & -0.4525 & 0.2029 \\ 0.4195 & 1.2763 & -0.9672 & 1.0527 \end{bmatrix} \begin{bmatrix} \text{Auckland}_{t} \\ \text{Paeroa}_{t} \\ \text{Hamilton}_{t} \\ \text{Reefton}_{t} \end{bmatrix}$$

(a) Mathematical model of interaction between multiple time-series



(b) Interaction network model

Figure 8.7. Mathematical model of interaction and its interaction network model between air pressure levels in Auckland, Paeroa, Hamilton and Reefton at the end of the testing period.

of air pressure levels in these four locations exhibit similar manner and are influencing each other. Based on this discovery, it can be proposed that one should consider taking into account information about weather conditions in distant places when forecasting weather condition in a particular location as there might exist direct on indirect interactions between variables of interest in those places.

When more data ranging from the beginning of October to the end of December 2010 (the test data set) is added to the DIN model, the interaction network model in general remains stable, as depicted in Figure 8.7. It is observable that the interactions between Auckland, Hamilton, Paeroa and Reefton generally stay in a similar manner. This outcome suggests that patterns of interactions between air pressure level at different locations in New Zealand are considerably stable over the period of evaluation.

Nevertheless, the adapted interaction network model (extracted at the end of the testing period) also shows that the strength of interactions between variables of interest has changed when compared to the ones from the training period. This finding suggest that the state of the weather system is actually changing over time and therefore the state of existing interactions are also changing dynamically, i.e. new interactions are emerging or existing interactions are disappearing.

Extracted profiles of relationships as shown in Figure 8.8 reveals that in the period of observation there is a significant number of episodes when Auckland, Hamilton and Paeroa are moving together ($N = 437$; where N is the exact number of occurrences of a particular profile) while Reefton moves independently ($N = 516$). This discovery indicates that the relationships between Auckland, Hamilton and Paeroa are stronger compared to their relationship with Reefton. In regards to this finding, DIN model (see Figure 8.6) is in general agreement with the LTM.

Additionally, Figure 8.8 also reveals that in the same period air pressure trajectories in Auckland, Paeroa, Hamilton and Reefton were frequently correlated with each other. This can be observed from the number of occurrences N, and the cluster diameter of the cluster to which these four locations were allocated together. The four sites were clustered together 365 times which was a considerably large number of co-occurrences during the examination period. The cluster diameter has a value of 0.0542, confirming that the air pressure

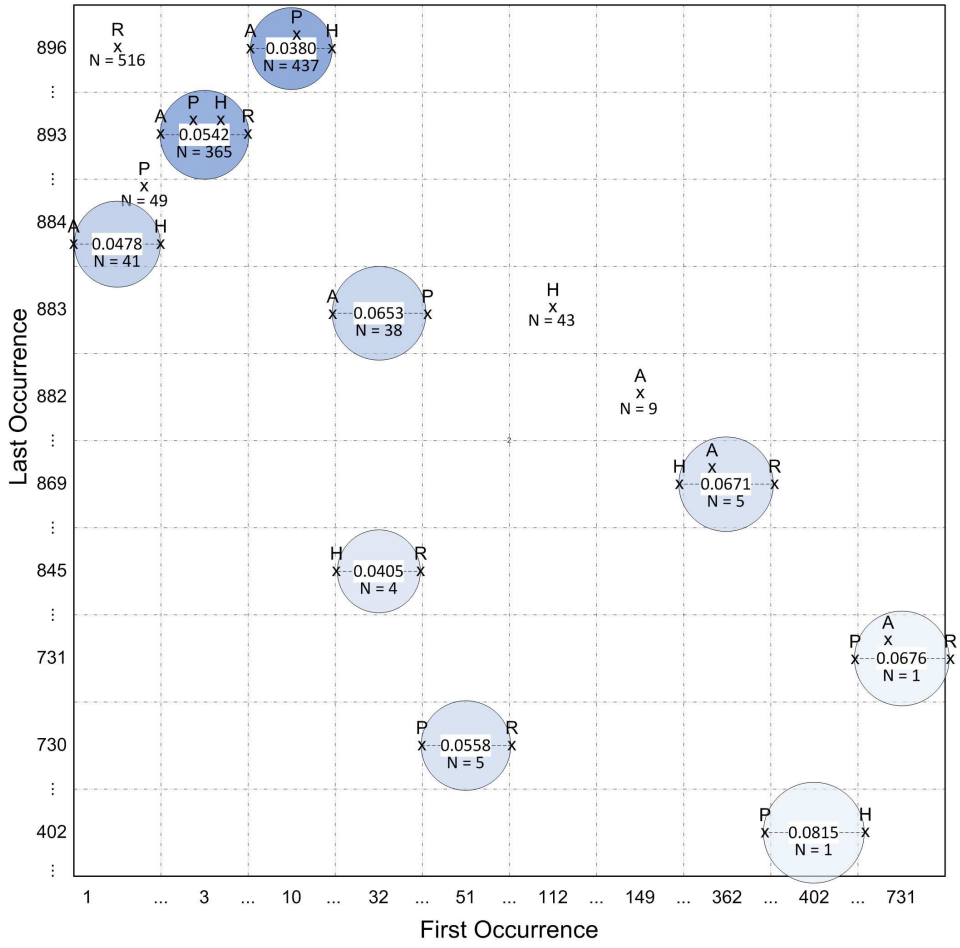


Figure 8.8. Profiles of relationships of LTM in the period of April 2008 to September 2010 of air pressure levels in Auckland (A), Paeroa (P), Hamilton (H) and Reefton (R).

level in these four observed locations progresses in a comparable fashion at certain periods of time. In LTM, cluster diameter represents maximum normalised correlation coefficients between time-series that belong to the same cluster. It ranges from 0 to 1, where 0 denotes high similarity and 1 signifies the opposite condition.

Consequently, the interaction network model as illustrated in Figure 8.6 indicates comparable actuality as it reveals the existence of fully interdependent relationships between air pressure level at the four locations.

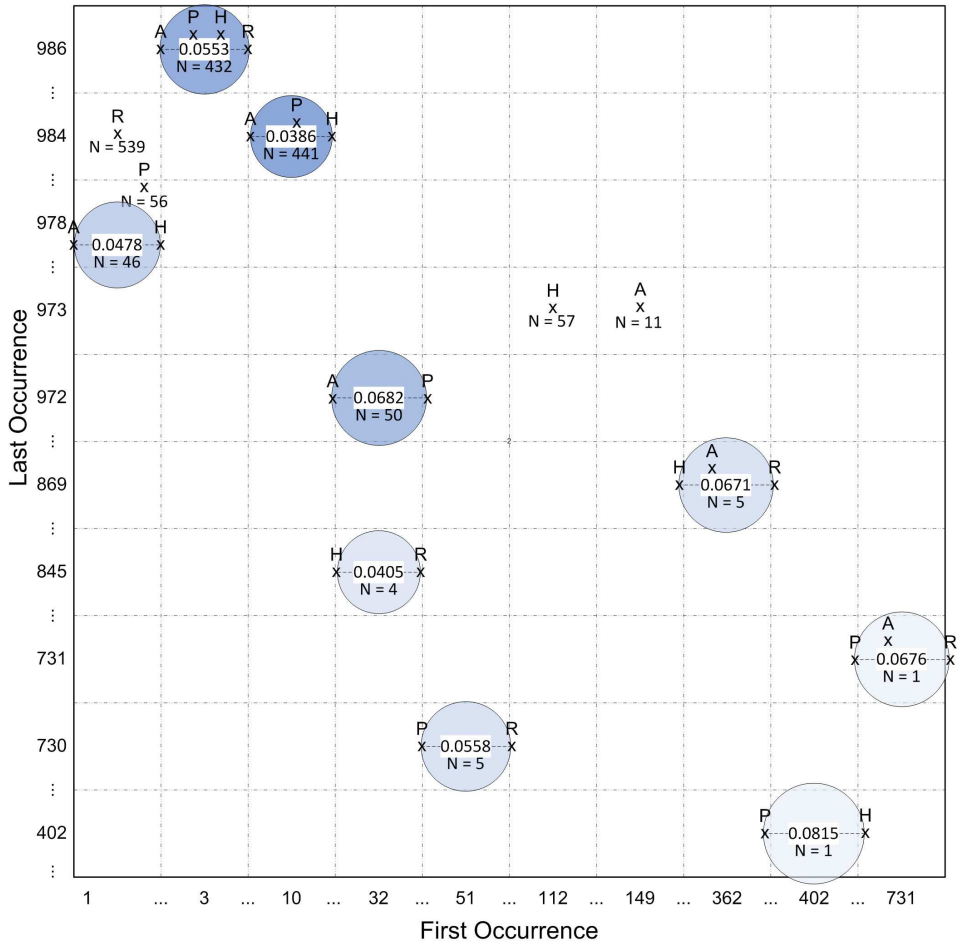


Figure 8.9. Profiles of relationships of LTM in the period of October to December 2010 of air pressure levels in Auckland (A), Paeroa (P), Hamilton (H) and Reefton (R).

The existence of a strong relationship between Auckland, Hamilton, Paeroa and Reefton was also being recognised by DIN after the data ranging from the beginning of October 2010 to the end of December 2010 is conferred (as illustrated in Figure 8.7). This result is again in agreement with the outcome from the LTM as during this period of examination it also recognised that Auckland, Hamilton, Paeroa and Reefton are progressing together in a significant number of episodes as depicted in Figure 8.9.

Figures 8.10 and 8.11 outline two sets of fuzzy rules that constructed the

Rule1

If RatioOfDiff Auckland(t)-Paeroa(t) is GaussianMF (0.33 0.27)
RatioOfDiff Auckland(t)-Hamilton(t) is GaussianMF (0.88 0.12)
RatioOfDiff Auckland(t)-Reefton(t) is GaussianMF (0.54 0.31)
RatioOfDiff Paeroa(t)-Hamilton(t) is GaussianMF (0.75 0.23)
RatioOfDiff Paeroa(t)-Reefton(t) is GaussianMF (0.40 0.28)
RatioOfDiff Hamilton(t)-Reefton(t) is GaussianMF (0.54 0.31)

Then Auckland(t+1) = 1.98*Auckland(t) + 0.35*Paeroa(t) – 1.74*Hamilton(t) + 0.41*Reefton(t)
Paeroa(t+1) = 1.11*Auckland(t) + 1.25*Paeroa(t) – 1.74*Hamilton(t) + 0.38*Reefton(t)
Hamilton(t+1) = 0.95*Auckland(t) + 0.42*Paeroa(t) – 0.76*Hamilton(t) + 0.40*Reefton(t)
Reefton(t+1) = 0.01*Auckland(t) + 1.16*Paeroa(t) – 1.50*Hamilton(t) + 1.33*Reefton(t)

Rule 2

If RatioOfDiff Auckland(t)-Paeroa(t) is GaussianMF (0.37 0.31)
RatioOfDiff Auckland(t)-Hamilton(t) is GaussianMF (0.77 0.11)
RatioOfDiff Auckland(t)-Reefton(t) is GaussianMF (0.61 0.24)
RatioOfDiff Paeroa(t)-Hamilton(t) is GaussianMF (0.67 0.19)
RatioOfDiff Paeroa(t)-Reefton(t) is GaussianMF (0.31 0.27)
RatioOfDiff Hamilton(t)-Reefton(t) is GaussianMF (0.56 0.29)

Then Auckland(t+1) = – 2.75*Auckland(t) + 3.50*Paeroa(t) + 0.01*Hamilton(t) + 0.24*Reefton(t)
Paeroa(t+1) = – 3.55*Auckland(t) + 4.39*Paeroa(t) + 0.07*Hamilton(t) + 0.23*Reefton(t)
Hamilton(t+1) = – 3.73*Auckland(t) + 3.54*Paeroa(t) + 0.94*Hamilton(t) + 0.25*Reefton(t)
Reefton(t+1) = – 3.90*Auckland(t) + 3.77*Paeroa(t) + 0.00*Hamilton(t) + 1.13*Reefton(t)

Rule 3

If RatioOfDiff Auckland(t)-Paeroa(t) is GaussianMF (0.30 0.27)
RatioOfDiff Auckland(t)-Hamilton(t) is GaussianMF (0.67 0.21)
RatioOfDiff Auckland(t)-Reefton(t) is GaussianMF (0.55 0.31)
RatioOfDiff Paeroa(t)-Hamilton(t) is GaussianMF (0.64 0.28)
RatioOfDiff Paeroa(t)-Reefton(t) is GaussianMF (0.36 0.23)
RatioOfDiff Hamilton(t)-Reefton(t) is GaussianMF (0.53 0.30)

Then Auckland(t+1) = 0.14*Auckland(t) + 1.12*Paeroa(t) – 0.43*Hamilton(t) + 0.16*Reefton(t)
Paeroa(t+1) = – 0.93*Auckland(t) + 2.08*Paeroa(t) – 0.30*Hamilton(t) + 0.15*Reefton(t)
Hamilton(t+1) = – 0.93*Auckland(t) + 1.17*Paeroa(t) + 0.62*Hamilton(t) + 0.15*Reefton(t)
Reefton(t+1) = – 2.17*Auckland(t) + 1.66*Paeroa(t) + 0.34*Hamilton(t) + 1.17*Reefton(t)

Figure 8.10. Extracted fuzzy rules (first-order Takagi-Sugeno type) from mTNFI when predicting last time point of test data. (Rule 1 - Rule 3)

inference system when mTNFI predicted the air pressure level at time-point 992 of the whole data set (the last point of the testing data set). As in previous experiment with the multiple stock market indexes, here the number of selected nearest samples used by mTNFI to build the inference system is again set to a fixed value of 80 samples. Figures 8.10 and 8.11 show that for the 90^{th}-point of the test data, 5 rules were created, indicating that the 80 nearest samples were grouped into 5 clusters in the modelling process.

The consequent parts of the fuzzy rules outlined in Figures 8.10 and 8.11,

Rule4

If RatioOfDiff Auckland(t)-Paeroa(t) is GaussianMF (0.32 0.32)
RatioOfDiff Auckland(t)-Hamilton(t) is GaussianMF (0.74 0.16)
RatioOfDiff Auckland(t)-Reefton(t) is GaussianMF (0.57 0.24)
RatioOfDiff Paeroa(t)-Hamilton(t) is GaussianMF (0.71 0.19)
RatioOfDiff Paeroa(t)-Reefton(t) is GaussianMF (0.48 0.31)
RatioOfDiff Hamilton(t)-Reefton(t) is GaussianMF (0.49 0.29)

Then Auckland(t+1) = – 0.10*Auckland(t) + 2.56*Paeroa(t) – 1.57*Hamilton(t) + 0.12*Reefton(t)
Paeroa(t+1) = – 1.11*Auckland(t) + 3.50*Paeroa(t) – 1.51*Hamilton(t) + 0.12*Reefton(t)
Hamilton(t+1) = – 0.93*Auckland(t) + 2.51*Paeroa(t) – 0.70*Hamilton(t) + 0.12*Reefton(t)
Reefton(t+1) = – 0.01*Auckland(t) + 2.13*Paeroa(t) – 2.22*Hamilton(t) + 1.10*Reefton(t)

Rule 5

If RatioOfDiff Auckland(t)-Paeroa(t) is GaussianMF (0.37 0.28)
RatioOfDiff Auckland(t)-Hamilton(t) is GaussianMF (0.89 0.11)
RatioOfDiff Auckland(t)-Reefton(t) is GaussianMF (0.53 0.24)
RatioOfDiff Paeroa(t)-Hamilton(t) is GaussianMF (0.84 0.17)
RatioOfDiff Paeroa(t)-Reefton(t) is GaussianMF (0.37 0.26)
RatioOfDiff Hamilton(t)-Reefton(t) is GaussianMF (0.53 0.31)

Then Auckland(t+1) = 1.02*Auckland(t) + 1.92*Paeroa(t) – 2.14*Hamilton(t) + 0.20*Reefton(t)
Paeroa(t+1) = – 0.05*Auckland(t) + 2.93*Paeroa(t) – 2.08*Hamilton(t) + 0.19*Reefton(t)
Hamilton(t+1) = 0.26*Auckland(t) + 1.75*Paeroa(t) – 1.20*Hamilton(t) + 0.19*Reefton(t)
Reefton(t+1) = 0.56*Auckland(t) + 1.12*Paeroa(t) – 1.76*Hamilton(t) + 1.09*Reefton(t)

Figure 8.11. Extracted fuzzy rules (first-order Takagi-Sugeno type) from mTNFI when predicting last time point of test data. (Rule 4 - Rule 5)

indicate that the upcoming air pressure level in a particular location is dependent on the air pressure level in other observed locations. In addition, the fuzzy rules also signify that there is a complex form of interdependencies between air pressure levels in different locations. For instance, each created rule indicates that the upcoming air pressure level in Auckland is being significantly influenced by Paeroa and Hamilton, while Reefton gave only a relatively small contribution compared to those of Paeroa and Hamilton. In addition, the rules also indicate that Reefton is being influenced more by Auckland, Paeroa and Hamilton rather than by itself, and so forth.

Furthermore, all created fuzzy rules are in line with findings from the DIN model and LTM, suggesting that movements of air pressure levels in Auckland, Hamilton, Paeroa and Reefton are frequently inter-correlated and influencing each other.

8.6 Conclusion

This chapter presented an application of DIN, LTM, mTNFI and IMMF for the analysis, modelling and prediction of a weather data set from different locations. As a case study daily air pressure levels collected at four different sites in New Zealand are used. Results from conducted experiments show that the DIN, LTM, mTNFI and the IMMF are capable to perform simultaneous multiple time-series prediction with a considerably high accuracy.

It was also found that during the period of examination, the performance of DIN, LTM, mTNFI and IMMF varies and no one single model was found to always perform the best across the whole testing period. However, the calculated RMSE suggests that the IMMF is a better and more stable prediction model. This finding yet again confirms that every model has their own power in prediction, and by integrating these models a powerful model for multiple time-series prediction can be realised.

Conclusively, outcomes from the comparative analysis of the proposed IMMF against MLR, MLP and random walk again clearly indicate the value of extracting and exploiting relationships between multiple variables, when the variables concerned are influencing each other in a dynamic fashion.

Chapter 9

Conclusion and Further Research

9.1 Dynamic Interaction Network: DIN

The first method for multiple time-series analysis and modelling introduced in this thesis is the Dynamic Interaction Network denoted as DIN, which was developed based on a method proposed by Kasabov et al. (2004) to identifying interdependencies between genes. DIN is a global modelling technique for multiple time-series that utilises the Kalman filter and the EM algorithm to perform a state-space estimation modelling by extracting a transition matrix from the inter-related multiple time-series data. Extending the method proposed by Kasabov et al. (2004), DIN employs an on-line learning process with the ability to capture time varying patterns of inter-relationships between multiple time-series. Therefore it is able to evolve its structure as new forms of relationship between variables emerge over time.

Results from experiments with synthetic data show that DIN as a global model for multiple time-series analysis and modelling is useful in capturing global trends of dynamic interactions between variables. Additionally, outcomes from experiments using multiple interactive stock markets in the Asia Pacific region and weather data from New Zealand suggest that extraction of DIN models (interaction network) reveals important and complex interdependencies among observed variables. Furthermore, it was also confirmed that interaction network model was not just limited to exposing essential relationships; it could also be employed to predict future values of observed variables

with an acceptable degree of accuracy.

9.2 Localised Trend Model: LTM

There is an understanding that the trajectories produced by global models often fail to track localised changes that take place at discrete points in time due to the fact that trajectories tend to smooth localised deviations by averaging the effects of such deviations over a long period of time. It is then logical to develop localised models that are built only on data that define the phenomenon under consideration and are not contaminated by data outside the underlying phenomenon.

A local model for multiple time-series analysis and modelling was then introduced in this research. The principal objective of the methodology, named Localised Trend Model or LTM, is to construct a repository of profiles and recurring trends whose structure will dynamically evolve as changes take place in the observed non-stationary environment over time. Results of conducted experiments with synthetic data, stock market indexes in the Asia Pacific region and the New Zealand's weather data suggests that LTM is capable of capturing more detail and localised patterns of relationship between variables and therefore in general offers better prediction accuracy compared to DIN.

9.3 Multivariate Transductive Neuro-Fuzzy Inference System: mTNFI

Both global and local models are realisations of the inductive reasoning approach, in which they are concerned with the creation of models (functions) from all available data or a number of local models, representing the entire problem space. Transductive inference, introduced in Vapnik (1998), is de-

fined in contrast as a method used to estimate the value of a potential model (function) only for a single point of space (that is, a new data vector) by utilising additional information related to that vector. A key advantage of this type of learning method is that this method is capable of constructing local and specific estimation model(s) for each new instance that needs to be classified or predicted (Kasabov, 2007b; Mitchell, 1997).

As multiple data streams consist of various variables producing examples continuously over time, the basic idea behind the proposed transductive model for multiple time-series analysis and modelling, named the Multivariate Transductive Neuro-Fuzzy Inference System (mTNFI), is simply to find and model relationships between variables that produce these streams of data at a particular time moment and then to search for similar forms of relationships from the past. Found instances will then be utilised to constitute a specific model of inference system to predict future values of the variables being observed.

The mTNFI is an extension of the NFI model for transductive reasoning (Q. Song & Kasabov, 2005). Modifications were made so that the new methodology is now capable of performing multiple time-series data analysis and modelling. As it develops an individual model over the new input vector, mTNFI provides a specific and better local generalisation and therefore offers a better prediction accuracy compared to DIN and LTM.

This conclusion is made based on results from experiments using synthetic data, multiple stock market indexes in the Asia Pacific region and the New Zealand's weather data.

9.4 Integrated Multi-Model Framework: IMMF

DIN is capable of capturing global trends of interaction in multiple time-series, while LTM captures local patterns of relationship between variables from different time localities, and the mTNFI is capable of constructing local and

specific fuzzy inference system for each new input vector. All three models are useful for complex modelling tasks, providing complementary information and knowledge and have their own predictive power. To allow multiple models with different approaches to the problem, and to extract knowledge with different predictive power to be integrated, a mixture of models in an integrated multi-model framework named the IMMF for multiple time-series analysis and modelling is proposed in this thesis.

The IMMF means to utilise the global and local model of inductive reasoning in recognising and modelling global trends of multiple time-series interactions along with their profiles of relationships and recurring trends in different time localities. On the other hand, the transductive reasoning is put into action to estimate movement of multiple time-series in short and chaotic periods. Integrating the three models through a weights adjustment module that regulates the contributing weight or level of trust given to each model based on its prior prediction error is expected to allow better prediction to be made than any single model alone.

The proposed integrated framework was evaluated using synthetic data and applied to two different case studies, which are the prediction of multiple stock market indexes in the Asia Pacific region and the prediction of air pressure level at different locations in New Zealand. Outcomes of conducted experiments with synthetic data and the two case studies reveal that the integrated multi-model framework is capable of dynamically adjusting the contributing weight assigned to each model and generally produces predictions with better accuracy compared to those of the global (DIN), local (LTM) and transductive model (mTNFI). Additionally, comparison with single time-series prediction methods such as MLR, MLP and random walk model confirmed that the proposed IMMF is superior.

9.5 General Discussion

The PhD study suggests that the dynamics of relationship between a set of related time-series variables from a specific setting can be modelled and utilised for simultaneous prediction of multiple time-series with a considerably high accuracy. In addition, comparison with outcomes from single-time series prediction models clearly indicates the value of extracting and exploiting relationships between multiple variables in prediction, when the variables concerned are influencing each other in a dynamic fashion.

Results from the two case studies indicate that the extraction of the dynamics of relationship between variables can be of help to predict their future values. Additionally, utilising the proposed methods the existence of important and complex interdependency in both multiple interactive stock markets in the Asia Pacific region and the state of air pressure level at different locations in New Zealand was revealed.

It can then be summarised that the idea of including the nature and strength of relationships between variables into a prediction model appears to be beneficial to the multiple time-series problem of modelling and predicting simultaneous movements of a collection of time sensitive variables which are related to each other. It allows the identification of complex dynamic interactions between variables that may lead to further studies, and allows for better accuracy in multiple time-series prediction.

Nevertheless, for some methods i.e. the LTM, the granularity of data snapshots is critical to the accuracy of the model, as different size of snapshot window might cause the extraction of varied information and understanding. Therefore, a careful selection or further analysis of how this granularity of data snapshots should be determined needs to be conducted.

9.6 Future Research

The work and research conducted during this PhD study is limited by time, yet there is much work to be done to extend this research and improve the performance of all proposed methods. Some possible future directions for this topic of research are proposed and discussed in the following three sections.

9.6.1 Modelling Non-Linear Relationships

Currently interaction between time-series is modelled in a linear form which is a simplified representation of more complex relationships between variables related to real world phenomena. In relation to this, one of the future directions of the study is to extend the proposed methods to allow the modelling of non-linear relationships between contributing variables in multiple time-series data.

To realise this, the extended Kalman filter (Welch & Bishop, 1995) which is capable of modelling non-linear relationships between variables can be used in place of the discrete Kalman filter in DIN. Also, in order to capture the existence of functional correlation between variables LTM can employ the correlation ratio (Roche, Malandain, Pennec, & Ayache, 1998) to replace the currently used Pearson's correlation. Additionally, a high-order Takagi-Sugeno fuzzy inference system that models relationships between variables as non-linear functions can be used in mTNFI to model more complex and non-linear relationships between contributing variables rather than the first-order Takagi-Sugeno fuzzy inference system.

9.6.2 Incorporating a Forgetting Function in LTM

As multiple data streams produce large amount of samples over time, the growth of the knowledge repository of the LTM will be immense. Nevertheless,

there is always some possibilities that certain profiles of relationships in the knowledge repository might became obsolete after sometimes. In relation to this possibility, a *forgetting function* that removes those configurations which have not been used for a certain period should be incorporated into the LTM. This functionality will help to eliminate the pressure of the repository growth of the LTM.

9.6.3 Parameter Optimisation

One of the challenges in the proposed local, transductive model and the integrated framework is parameter optimisation. These algorithms have several parameters that have significant impact on the performance. For instance, in both LTM and mTNFI the distance threshold affects the number of created clusters, which in turn sets the number of clusters of recurring trends in LTM or fuzzy rules in mTNFI used for prediction. Another example of a parameter that needs to be optimised in mTNFI is the number of nearest neighbours, which will affect the reliability of constructed models/rules. Additionally, in the integrated framework the learning rate and the number of training iterations are also important factors that have an effect on the prediction accuracy and the convergence of the learning process.

The parameters in LTM, mTNFI and IMMF are currently being set by hand based on the prediction accuracy of the training data without any optimisation. The logical next step of the study is optimisation of parameters with the widely-used evolutionary optimisation algorithm, known as genetic algorithm (GA) (Chafekar, Shi, Rasheed, & Xuan, 2005; Medeiros, Amaral, & Campello, 2006; Shoenauer & Xanthakis, 1993), or with the state-of-the-art optimisation algorithms developed in the Knowledge Engineering and Discovery Research Institute of AUT (`http://kedri.info`) known as the *Versatile Quantum Inspired Evolutionary Algorithm* (vQEA)(Platelt, Schliebs, &

Kasabov, 2007; Schliebs, Platel, Worner, & Kasabov, 2009).

9.6.4 Computational Load Assessment of the Proposed Methods

Currently the main objective of the PhD study is to capture and model dynamic patterns of interactions between multiple time-series to discover new knowledge and improve our understanding about the behaviour of a dynamic system related to the real world problems. Additionally, the study also aims to perform simultaneous multiple time-series prediction by utilising knowledge about the dynamic of interactions between variables over time. Therefore, no assessment of computational load of the proposed methods i.e. DIN, LTM, mTNFI and IMMF has been done.

However, prior to implementing the proposed methods in a production stage, assessments of the computational performance of the proposed methods need to be conducted as a future task of this study.

9.6.5 peSNN Reservoir for Multiple Time-Series Analysis

In recent years, triggered by the need to better understand and mimic the brain's capability to process information, the development of more complex and biologically plausible neural network or connectionist models named *spiking neural networks* (SNN) became known (Gerstner & Kistler, 2002; Maass & Bishop, 2001; Vreeken, 2003). Different to the widely-known and used neural network models, i.e. perceptron or MLP, SNN models use trains of spikes to represent the input signals rather than continuous variables. Many studies have attempted to use these models to solve complex real world problems, and some of them demonstrated very promising results. As an extension to

SNN models, an *evolving spiking neural network* (eSNN) architecture which is capable of modifying the structure and connection weights of the network during the training process was proposed in Kasabov (2009). The eSNN employs a supervised learning methodology and is mainly used as a classifier that learns the mapping from a single data vector to a specified class label and is therefore very suitable for the classification of time-invariant data (Sichtig, Schaffer, & Riva, 2010; Soltic, Wysoski, & Kasabov, 2008; Wysoski, Benuskova, & Kasabov, 2010).

Nevertheless, most of today's data volumes are continuously updated over time, adding an additional time dimension to the data sets resulting in a spatio/spectro-temporal data which the eSNN is not capable of dealing with. In relation to this, an extension of eSNN that enables the processing of spatio/spectro-temporal information is proposed in Schliebs et al. (2010). The main idea of the proposed architecture is to add an additional layer to the network architecture that transforms the spatio/spectro-temporal input pattern into a single high-dimensional network state (see Figure 9.1). For a classification task, the mapping of this intermediate state into a desired class label can then be learned by any of the classifiers in the output layer, for instance the eSNN.

In their study, Schliebs et al. (2010) employed the concepts of the reservoir computing paradigm (Verstraeten, Schrauwen, D'Haene, & Stroobandt, 2007) to realise the spatio/spectro-temporal filter, in which the reservoir is represented by a large recurrent neural network whose topology and connection weight matrix is fixed. The main functionality of the reservoir in the proposed architecture is to project the network inputs into a high-dimensional space in order to enhance their separability. Here, Schliebs et al. (2010) introduced the use of *probabilistic evolving spiking neural network* (peSNN) (Kasabov, 2009, 2010) when constructing the spatio/spectro-temporal filter. Based on their experimental results in Schliebs et al. (2010), it was found that proba-

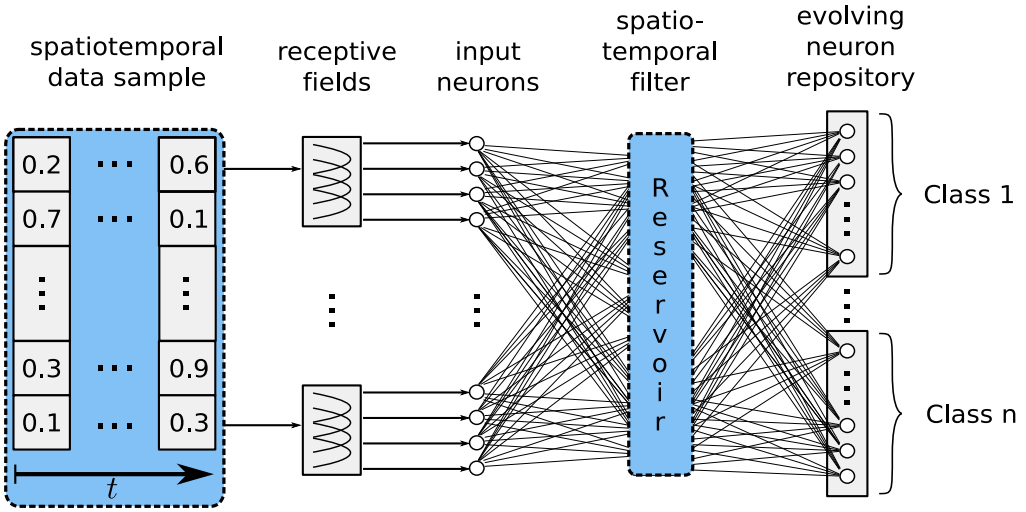


Figure 9.1. Extended evolving spiking neural network (Schliebs et al., 2010).

bilistic neural models are principally suitable reservoirs that have furthermore the potential to enhance the separation ability of spatio-temporal data. Additionally, due to the use of recurrent networks, the state of the reservoir can incorporate temporal information present in the input signals. Thus, the reservoir approach is very suitable to process spatio/spectro-temporal data.

As streams of multiple time-series data relating to real world phenomena can be considered as spatio/spectro-temporal data, it then would be interesting to investigate whether the reservoir computing based architecture proposed in Schliebs et al. (2010) can be implemented for multiple time-series analysis and modelling. The nature of the spiking neurons used to construct the reservoir, which is capable of capturing not only the spatio/spectro but also the temporal information of the input signals is the main appealing feature of the framework. Being able to store the information of both spatio/spectro data and the temporal aspect, it is expected that the peSNN reservoir computing is a suitable approach for multiple time-series analysis and modelling. Furthermore, as the spatio/spectro-temporal filter transforms the input signals to a single high-dimensional network state it would also be interesting to study if this high-dimensional network state actually represents new infor-

mation or knowledge related to the system under examination, such as the interactions between series over time.

However, as the initial aim of the proposed architecture in (Schliebs et al., 2010) is to solve classification problems, some alterations need to be made before it can be implemented for multiple time-series analysis and modelling. The main modification that needs to be made to the proposed architecture (as shown in Figure 9.1) would be the replacement of the classification module with multivariate time-series analysis models.

Appendix A

Parameter Estimation

This appendix derives various results for least squares estimation of the multiple linear regression model using matrix notation and matrix algebra.

A.1 Ordinary Least-Square Estimator

Throughout this appendix, the t subscript is used to index observations and an n to denote the sample size. It is useful to write the multiple linear regression model with k parameters as follows:

$$y_t = \beta_0 + \beta_1 x_{t1} + \beta_2 x_{t2} + ... + \beta_k x_{tk} + \varepsilon_t, \ t = 1, 2, ..., n, \tag{A.1}$$

where y_t is the dependent variable for observation t, and x_{tj}, $j = 1, 2, ..., k$, are the independent variables. As usual, β_0 is the intercept and $\beta_1, ..., \beta_k$ denote the slope parameters.

For each t, define a $1 \times (k+1)$ vector, $\mathbf{x}_t = (1, x_{t1}, ..., x_{tk})$, and let $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_k)$ be the $(k+1) \times 1$ vector of all parameters. Then, we can write Equation (A.1) as

$$y_t = \mathbf{x}_t \boldsymbol{\beta} + \varepsilon_t, \ t = 1, 2, ..., n. \tag{A.2}$$

Equation (A.2) can be written in full matrix notation by appropriately defining data vectors and matrices. Let $\mathbf{y}$ denote the $n \times 1$ vector of observations on y: the t^{th} element of $\mathbf{y}$ is y_t. Let $\mathbf{X}$ be the $n \times (k+1)$ vector of observations on the explanatory variables. In other words, the t^{th} row of $\mathbf{X}$ consists of vector

$\mathbf{x}_t$ as it is written out in detail:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & & & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

Finally, let $\boldsymbol{\varepsilon}$ be the $n \times 1$ vector of unobservable errors or disturbances. Then, Equation (A.2) for all n observations can be written in matrix notation:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}. \tag{A.3}$$

Estimation of $\boldsymbol{\beta}$ proceeds by minimising the sum of squared residual. Define the sum of squared residuals fuction for any possible $(k+1) \times 1$ parameter vector $\mathbf{b}$ as

$$\text{SSR}(\mathbf{b}) \equiv \sum_{t=1}^{n} (y_t - \mathbf{x}_t \mathbf{b})^2. \tag{A.4}$$

The $(k+1) \times 1$ vector of ordinary least squares estimates, $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_k)'$, minimises SSR($\mathbf{b}$) over all possible $(k+1) \times 1$ vectors $\mathbf{b}$. This is a problem in multivariable calculus. For $\hat{\boldsymbol{\beta}}$ to minimise the sum of squared residuals, it must solve the first order condition

$$\partial \text{SSR}(\hat{\boldsymbol{\beta}}) / \partial \mathbf{b} \equiv 0 \tag{A.5}$$

Using the fact that the derivative of $(y_t - \mathbf{x}_t \mathbf{b})^2$ with respect to $\mathbf{b}$ is the $1 \times (k+1)$ vector $-2(y_t - \mathbf{x}_t \mathbf{b})\mathbf{x}_t$, (A.5) is equivalent to

$$\sum_{t=1}^{n} \mathbf{x}_t' (y_t - \mathbf{x}_t \hat{\boldsymbol{\beta}}) \equiv 0. \tag{A.6}$$

This first order condition can be written as

$$\begin{aligned} \sum_{t=1}^{n} (y_t - \hat{\beta}_0 - \hat{\beta}_1 x_{t1} - ... - \hat{\beta}_k x_{tk}) &= 0 \\ \sum_{t=1}^{n} x_{t1} (y_t - \hat{\beta}_0 - \hat{\beta}_1 x_{t1} - ... - \hat{\beta}_k x_{tk}) &= 0 \\ \vdots & \\ \sum_{t=1}^{n} x_{tk} (y_t - \hat{\beta}_0 - \hat{\beta}_1 x_{t1} - ... - \hat{\beta}_k x_{tk}) &= 0 \end{aligned}$$

which are often called ordinary least-square (OLS) (Wooldridge, 2006) first order conditions. It is meaningful to write these in matrix form to make them easier to manipulate. It is observable that Equation (A.6) is equivalent to

$$\mathbf{X}'(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = 0 \tag{A.7}$$

or

$$(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y} \tag{A.8}$$

It can be shown that Equation (A.8) always has at least one solution. Multiple solutions provide no help, as what is being looked for is a unique set of OLS estimates given a data set. Assuming that the $(k+1) \times (k+1)$ symmetric matrix $\mathbf{X}'\mathbf{X}$ is nonsingular, both sides of (A.8) can be premultiplied by $(\mathbf{X}'\mathbf{X})^{-1}$ to solve estimator $\hat{\boldsymbol{\beta}}$:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{A.9}$$

This is the critical formula for matrix analysis of the multiple linear regression model. The assumptions that $\mathbf{X}'\mathbf{X}$ is invertible, is equivalent to the assumption that $\text{rank}(\mathbf{X}) = (k+1)$, which means that the columns of $\mathbf{X}$ must be linearly independent.

The $n \times 1$ vectors of OLS fitted values and residuals are given by

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{X}\hat{\boldsymbol{\beta}}, \\ \hat{\boldsymbol{\varepsilon}} &= \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}. \end{aligned} \tag{A.10}$$

The matrix approach to multiple regression can be used as the basis for a geometrical interpretation of regression.

A.2 Weighted Least-Square Estimator

The method of least square can be extended by assigning different weights to the data points. This methodology is known as the weighted least-square

(Weisberg, 2005). If $w_1, w_2, ..., w_n$ denote the weighted assigned, the weighted sum of squares (WSS) is minimised, instead of the sum of squares residual:

$$\text{WSS}(\mathbf{b}) \equiv \sum_{t=1}^{n} w_t(y_t - \mathbf{x}_t\mathbf{b})^2. \tag{A.11}$$

This includes ordinary least squares as the special case where all the weights $w_t = 1$.

The $(k+1)\times 1$ vector of weighted least squares estimates, $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_k)$, minimises WSS($\mathbf{b}$) over all possible $(k+1) \times 1$ vectors $\mathbf{b}$. For $\hat{\boldsymbol{\beta}}$ to minimise WSS, it must solve the first order condition

$$\partial \text{WSS}(\hat{\boldsymbol{\beta}})/\partial \mathbf{b} \equiv 0 \tag{A.12}$$

The derivative of $w_t(y_t - \mathbf{x}_t\mathbf{b})^2$ with respect to $\mathbf{b}$ is the $1 \times (k+1)$ vector $-2w_t(y_t - \mathbf{x}_t\mathbf{b})\mathbf{x}_t$, Equation (A.12) is equivalent to

$$\sum_{t=1}^{n} \mathbf{x}_t' w_t(y_t - \mathbf{x}_t\hat{\boldsymbol{\beta}}) \equiv 0. \tag{A.13}$$

Let $\mathbf{W}$ denote the $n \times n$ diagonal matrix with elements $w_1, w_2, ..., w_n$ which are the weights for the unobservable errors or disturbances. The weighted sum of squared (A.13) is equivalent to

$$\mathbf{X}'\mathbf{W}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = 0 \tag{A.14}$$

or

$$(\mathbf{X}'\mathbf{W}\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{W}\mathbf{y} \tag{A.15}$$

It can be shown that Equation (A.15) always has at least one solution. Again, multiple solutions should provide no help, as what is being looked for is a unique set of OLS estimates given a data set. Assuming that the $(k+1)\times(k+1)$ symmetric matrix $\mathbf{X}'\mathbf{W}\mathbf{X}$ is nonsingular, both sides of Equation (A.15) are premultiplied by $(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}$ to solve estimator $\hat{\boldsymbol{\beta}}$:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y} \tag{A.16}$$

Application of the weighted least square method requires the knowledge of the weights, w_t. Sometimes prior knowledge, experience, or information from the theoretical model can be used to determine the weights. In some cases it is necessary to guess the weights, perform the analysis, and then re-estimate the weights based on the results. Several iterations may be necessary.

A.3 Recursive Least-Square Estimator

This section explains a recursive method to compute least squares estimation solution known as the recursive least-square (Vu, 2007). The previous section shows that the least squares estimation has the solution as in Equation A.9. Since we have t observations, we can attach the parameter vector with the subscript t to indicate that the parameter vector has been estimated with t observations, or it has been estimated at the time labelled t. We can write Equation A.9 as follows

$$\hat{\boldsymbol{\beta}}_t = \mathbf{P}_t \mathbf{X}_t' \mathbf{y}_t \tag{A.17}$$

where

$$\mathbf{P}_t = (\mathbf{X}_t' \mathbf{X}_t)^{-1}$$

The initial value of $\hat{\boldsymbol{\beta}}_n$ can be calculated from Equation A.17 by using the first n observations from the learning data set.

Now, when the observation $t+1$ is available, we want to find the relation between $\hat{\boldsymbol{\beta}}_{t+1}$ and $\hat{\boldsymbol{\beta}}_t$. By using a matrix identity of the inverse of the sum of a matrix and an outer product of two vectors, known as the Sherman-Morrison

formula, we can write

$$\begin{aligned}
\mathbf{P}_{t+1} &= \left(\mathbf{X}'_{t+1}\mathbf{X}_{t+1}\right)^{-1} \\
&= \left(\mathbf{X}'_{t}\mathbf{X}_{t} + \mathbf{x}'_{t+1}\mathbf{x}_{t+1}\right)^{-1} \\
&= \left(\mathbf{X}'_{t}\mathbf{X}_{t}\right)^{-1} - \frac{\left(\mathbf{X}'_{t}\mathbf{X}_{t}\right)^{-1}\mathbf{x}'_{t+1}\mathbf{x}_{t+1}\left(\mathbf{X}'_{t}\mathbf{X}_{t}\right)^{-1}}{1+\mathbf{x}_{t+1}\left(\mathbf{X}'_{t}\mathbf{X}_{t}\right)^{-1}\mathbf{x}'_{t+1}} \\
&= \mathbf{P}_t - \frac{\mathbf{P}_t\mathbf{x}'_{t+1}\mathbf{x}_{t+1}\mathbf{P}_t}{1+\mathbf{x}_{t+1}\mathbf{P}_t\mathbf{x}'_{t+1}}
\end{aligned} \tag{A.18}$$

and therefore we can write

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_{t+1} &= \left(\mathbf{X}'_{t+1}\mathbf{X}_{t+1}\right)^{-1}\mathbf{X}'_{t+1}\mathbf{y}_{t+1} \\
&= \left(\mathbf{P}_t - \frac{\mathbf{P}_t\mathbf{x}'_{t+1}\mathbf{x}_{t+1}\mathbf{P}_t}{1+\mathbf{x}_{t+1}\mathbf{P}_t\mathbf{x}'_{t+1}}\right)\left(\mathbf{X}'_t\mathbf{y}_t + \mathbf{x}'_{t+1}y_{t+1}\right) \\
&= \left(\mathbf{I}_t - \frac{\mathbf{P}_t\mathbf{x}'_{t+1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}\mathbf{P}_t\mathbf{x}'_{t+1}}\right)\mathbf{P}_t\left(\mathbf{X}'_t\mathbf{y}_t + \mathbf{x}'_{t+1}y_{t+1}\right) \\
&= \left(\mathbf{I}_t - \frac{\mathbf{P}_t\mathbf{x}'_{t+1}\mathbf{x}_{t+1}}{1+\mathbf{x}_{t+1}\mathbf{P}_t\mathbf{x}'_{t+1}}\right)\left(\hat{\boldsymbol{\beta}}_t + \mathbf{P}_t\mathbf{x}'_{t+1}y_{t+1}\right) \\
&= \hat{\boldsymbol{\beta}}_t + \frac{\mathbf{P}_t\mathbf{x}'_{t+1}}{1+\mathbf{x}_{t+1}\mathbf{P}_t\mathbf{x}'_{t+1}}\left(y_{t+1} - \mathbf{x}_{t+1}\hat{\boldsymbol{\beta}}_t\right) \\
&= \hat{\boldsymbol{\beta}}_t + \mathbf{P}_{t+1}\mathbf{x}'_{t+1}\left(y_{t+1} - \mathbf{x}_{t+1}\hat{\boldsymbol{\beta}}_t\right)
\end{aligned} \tag{A.19}$$

The estimated parameter $\hat{\boldsymbol{\beta}}_{t+1}$ in recursive form as in the above equation has many advantages. We can see right away that it makes the problem of matrix inversion in successive calculation disappear. Also, we can see the effect on the parameter vector of the new observation. This is so convenient for process control because the parameter vector is updated in real time.

A.4 Weighted Recursive Least-Square Estimator

The previous recursive least square estimation weighs each observation equally. The method can be extended by assigning weights to each observation. Since we have t observations, we can write Equation A.16 as follows

$$\hat{\boldsymbol{\beta}}_t = \mathbf{P}_t\mathbf{X}'_t\mathbf{W}_t\mathbf{y}_t \tag{A.20}$$

where

$$\mathbf{P}_t = (\mathbf{X}_t' \mathbf{W}_t \mathbf{X}_t)^{-1}$$

The matrix $\mathbf{W}_t$ is the weighting matrix

$$\mathbf{W}_t = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & & \vdots & \\ 0 & 0 & \cdots & w_t \end{bmatrix} \tag{A.21}$$

The solution for weighted recursive least squares estimator with forgetting factor is defined as follows (Kasabov, 2007a; Vu, 2007):

$$\hat{\boldsymbol{\beta}}_{t+1} = \hat{\boldsymbol{\beta}}_t + w_{t+1} \mathbf{P}_{t+1} \mathbf{x}_{t+1}' \left(y_{t+1} - \mathbf{x}_{t+1} \hat{\boldsymbol{\beta}}_t \right) \tag{A.22}$$

$$\mathbf{P}_{t+1} = \frac{1}{\lambda} \left(\mathbf{P}_t - \frac{w_{t+1} \mathbf{P}_t \mathbf{x}_{t+1}' \mathbf{x}_{t+1} \mathbf{P}_t}{\lambda + \mathbf{x}_{t+1} \mathbf{P}_t \mathbf{x}_{t+1}'} \right) \tag{A.23}$$

where λ is a forgetting factor, its value is usually chosen between 0.8 and 1. This means the most recent data have the greatest weight.

Appendix B

Optimisation Algorithm

The objective of this Appendix is to outline an algorithm to optimise a performance index $F(\mathbf{x})$. This means to find the value of $\mathbf{x}$ that minimises $F(\mathbf{x})$. The optimisation algorithm that will be discussed is iterative. We begin from some initial guess, $\mathbf{x}_0$, and then update our guess in stages according to an equation of the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{B.1}$$

or

$$\Delta \mathbf{x}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k) = \alpha_k \mathbf{p}_k \tag{B.2}$$

where the vector $\mathbf{p}_k$ represents a search direction, and positive scalar α_k is the learning rate, which determines the length of the step.

B.1 Gradient Descent

One of the simplest network training algorithms is *gradient descent*, sometimes also known as *steepest descent* (Boyd & Vandenberghe, 2004; Hagan et al., 1995). Nevertheless, it remains one of the more popular deterministic approaches.

When we update our guess of the optimum (minimum) point using Equation (B.1), we would like to have the function decrease at each iteration, as follows

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k) \tag{B.3}$$

Algorithm B.1 Steepest descent method.

Input: a guess point $\mathbf{x} = \mathbf{x}_0$.
  repeat
    Compute the steepest descent direction $\mathbf{p}$
    Line search. Choose a step, α.
    Update. $\mathbf{x} := \mathbf{x} + \alpha\mathbf{p}$.
  until stopping criterion is satisfied.

Consider the first order of Taylor series expansion of $F(\mathbf{x})$ about the old guess $\mathbf{x}_k$:

$$F(\mathbf{x}_{k+1}) = F(\mathbf{x}_k + \delta\mathbf{x}_k) = F(\mathbf{x}_k) + \mathbf{g}_k^{\mathrm{T}}\Delta\mathbf{x}_k \tag{B.4}$$

where $\mathbf{g}_k$ is the gradient evaluated at the old guess $\mathbf{x}_k$:

$$\mathbf{g}_k = \nabla F(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_k} \tag{B.5}$$

To achieve the objective of Equation (B.3), the second term on the right-hand side of Equation (B.4) must be negative:

$$\mathbf{g}_k^{\mathrm{T}}\Delta\mathbf{x}_k = \alpha_k\mathbf{g}_k^{\mathrm{T}}\mathbf{p}_k < 0 \tag{B.6}$$

Any vector $\mathbf{p}_k$ that satisfies this equation is called a *descent direction.* If we take a small enough step (α_k that is small, but greater than zero) in this direction, the function will decrease most rapidly when $\mathbf{g}_k^{\mathrm{T}}\mathbf{p}_k$ is most negative. This is an inner product between the gradient and the direction vector. It will be most negative when the direction vector is the negative of the gradient. Therefore a vector that points in the steepest descent direction is

$$\mathbf{p}_k = -\mathbf{g}_k \tag{B.7}$$

Using this in the iteration of Equation (B.1) produces the method of steepest descent as follows

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\mathbf{g}_k \tag{B.8}$$

For steepest descent there are two general methods for determining the learning rate, α_k. One approach is to minimise the performance index $F(\mathbf{x})$ with

respect to α_k at each iteration. In this case we are minimising along the line

$$\mathbf{x}_k - \alpha_k \mathbf{g}_k \tag{B.9}$$

The other method for selecting α_k is to use a fixed value (e.g., $\alpha_k = 0.02$), or to use variable, but predetermined values (e.g., $\alpha_k = 1/k$).

Appendix C

Lag Operator

In time series analysis, the *lag operator* (Hamilton, 1994; Verbeek, 2008) operates on an element of a time series to produce the previous element. For example, given some time series $X = \{X_1, X2, ...\}$ then

$$LX_t = X_{t-1}, \text{for all } t > 1 \tag{C.1}$$

where L is the lag operator. Note that the lag operator can be raised to arbitrary integer powers so that

$$L^{-1}X_t = X_{t+1}$$

and

$$L^k X_t = X_{t-k}$$

C.1 Lag Polynomials

Also polynomials of the lag operator can be used. For example,

$$\varepsilon_t = X_t - \sum_{i=1}^{p} \phi_i X_{t-i} = (1 - \sum_{i=1}^{p} \phi_i L^i) X_t$$

specifies an AR(p) model.

A polynomial of lag operators is called a lag polynomial so that, for example, the ARMA model can be concisely specified as

$$\phi X_t + \theta \varepsilon_t$$

where ϕ and θ respectively represent the lag polynomials,

$$\phi = 1 - \sum_{i=1}^{p} \phi_i L^i$$

and

$$\theta = 1 + \sum_{i=1}^{q} \theta_i L^i$$

C.2 Difference Operator

In time series analysis, the first difference operator Δ is a special case of lag polynomial.

$$\begin{aligned} \Delta X_t &= X_t - X_{t-1} \\ \Delta X_t &= (1 - L)X_t \end{aligned} \tag{C.2}$$

Similarly, the second difference operator

$$\begin{aligned} \Delta(\Delta X_t) &= \Delta X_t - \Delta X_{t-1} \\ \Delta^2 X_t &= (1 - L)\Delta X_t \\ \Delta^2 X_t &= (1 - L)(1 - L)X_t \\ \Delta^2 X_t &= (1 - L)^2 X_t \end{aligned} \tag{C.3}$$

The above approach generalises to the i^{th} difference operator

$$\Delta^i X_t = (1 - L)^i X_t \tag{C.4}$$

References

Adam, A., & Tweneboah, G. (2008). Macroeconomics factors and stock market movement: Evidence from ghana [MPRA Paper]. *Munich Personal RePEc Archive*(11256).

Amari, S. (1990). Mathematical foundations of neuro-computing. , *78*.

Angelov, P. (2002). *Evolving rule-based models: a tool for design of flexible adaptive systems*. London, UK: Springer-Verlag.

Angelov, P. (2006, October). Evolving fuzzy rulebased systems for modelling of non-linear non-stationary processes. In *Ifac workshop energy efficient control* (pp. 43–50).

Angelov, P., & Buswell, R. (2002, October). Identification of evolving fuzzy rule-based models. *IEEE Transactions on Fuzzy Systems*, *10*, 667–677.

Angelov, P., & Filev, D. (2004a). An approach to online identification of takagi-sugeno fuzzy models. *IEEE Transactions on Cybernetics*, *34*, 484–498.

Angelov, P., & Filev, D. (2004b, February). An approach to online identification of takagi-sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *34*, 484–498.

Angelov, P., & Filev, D. (2004c). Flexible models with evolving structure. *International Journal of Intelligent Systems*, *19*(4), 327–340.

Ankenbrand, T., & Tomassini, M. (1996, March). Predicting multivariate financial time series using neural networks: the swiss bond case. In *Proceedings of conference on computational intelligence for financial engineering, IEEE/IAFE 1996* (pp. 27–33).

Antoniou, A., Pescetto, G., & Violaris, A. (2003, June). Modelling international price relationships and interdependencies between the stock index

and stock index futures markets of three EU countries: A multivariate analysis. *Journal of Business Finance & Accounting*, *30*, 645–667.

Aoki, M., & Shell, K. (1989). *Optimization of stochastic systems, second edition: Topics in discrete-time dynamics (economic theory, econometrics, and mathematical economics)*. Academic Press.

Arroyo, F., Gonzalo, A., & Moreno, L. (1991). Neural network design for mobile robot control following a contour. In A. Prieto (Ed.), *Lecture Notes in Computer Science, artificial neural networks* (Vol. 540, pp. 430–436). Springer Berlin / Heidelberg.

Arslanalp, R., & Tola, T. (2006, June). A new state space representation method for adaptive log domain systems. In *First NASA/ESA conference on adaptive hardware and systems, 2006. AHS 2006* (pp. 122–128).

Atkeson, C., Moore, A., & Schaal, S. (1997, February). Locally weighted learning. *Artif. Intell. Rev.*, *11*, 11–73.

Azmy, W., El Gayar, N., Atiya, A., & El-Shishiny, H. (2009). MLP, gaussian processes and negative correlation learning for time series prediction. In J. Benediktsson, J. Kittler, & F. Roli (Eds.), *Lecture Notes in Computer Science, multiple classifier systems* (Vol. 5519, p. 428-437). Springer Berlin / Heidelberg.

Barry, R., & Chorley, R. (2003). *Atmosphere, weather and climate 8th edition*. Routledge.

Baruch, I., & Stoyanov, I. (1995, March). Global model of neural networks, stability and learning. In *Eighteenth convention of electrical and electronics engineers in israel, 1995* (pp. 4.3.5/1–4.3.5/5).

Bay, J. (1998). *Fundamentals of linear state space systems*. McGraw-Hill Science.

Beelders, O. (2002). International stock market interdependence: A south african perspective [Working Paper Series]. *SSRN eLibrary*.

Ben-Dor, A., & Yakhini, Z. (1999). Clustering gene expression patterns. In

Proceedings of the third annual international conference on computational molecular biology, RECOMB '99 (pp. 33–42). New York, NY, USA: ACM.

Bezdek, J. (1981). *Pattern recognition with fuzzy objective function algorithms*. Norwell, MA, USA: Kluwer Academic Publishers.

Bi, C. (2009, July). A monte carlo EM algorithm for de novo motif discovery in biomolecular sequences. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, *6*, 370–386.

Bierens, H. (1994). The nadaraya-watson kernel regression function estimator. In *Topics in advanced econometrics estimation, testing, and specification of cross-section and time series models* (pp. 212–247). NATO ASI Series.

Borghers, E., & Wessa, P. (2011, March). *Statistics - econometrics - forecasting.* Office for Research Development and Education. Available from `http://www.xycoon.com/`

Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual ACM workshop on computational learning theory* (pp. 144–152). ACM Press.

Bosnic, Z., Kononenko, I., Robnik-Sikonja, M., & Kukar, M. (2003, September). Evaluation of prediction reliability in regression using the transduction principle. In *The IEEE region 8 EUROCON 2003, computer as a tool* (Vol. 2, pp. 99–103).

Box, G., & Jenkins, G. (1970). *Time series analysis: Forecasting and control*. San Francisco: Holden-Day.

Box, G., Jenkins, G., & Reinsel, G. (2008). *Time series analysis: Forecasting and control, 4th edition*. Wiley.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Brahmasrene, T., & Jiranyakul, K. (2007, September). Cointegration and

causality between stock index and macroeconomics variables in an emerging market. *Academy of Accounting and Financial Studies Journal*, *11*(113), 123–146.

Brown, R. (1983). *Introduction to random signal analysis and kalman filtering.* Wiley, New York.

Brummer, J., & Strydom, L. (1997). An euclidean distance measure between covariance matrices of speech cepstra for text-independent speaker recognition. In *Proceedings of the 1997 south african symposium on communications and signal processing, COMSIG '97* (pp. 167–172).

Calcagno, G., Staiano, A., Fortunato, G., Brescia-Morra, V., Salvatore, E., Liguori, R., et al. (2010, November). A multilayer perceptron neural network-based approach for the identification of responsiveness to interferon therapy in multiple sclerosis patients. *Inf. Sci.*, *180*, 4153–4163.

Camastra, F., & Filippone, M. (2007). SVM-based time series prediction with nonlinear dynamics methods. In B. Apolloni, R. Howlett, & L. Jain (Eds.), *Lecture Notes in Computer Science, knowledge-based intelligent information and engineering systems* (Vol. 4694, pp. 300–307). Springer Berlin / Heidelberg.

Cernansky, M., & Makula, M. (2005, July). Feed-forward echo state networks. In *Proceedings of IEEE international joint conference on neural networks, IJCNN '05* (Vol. 3, pp. 1479–1482).

Cevikalp, H., & Polikar, R. (2008, October). Local classifier weighting by quadratic programming. *IEEE Transactions on Neural Networks*, *19*, 1832–1838.

Chafekar, D., Shi, L., Rasheed, K., & Xuan, J. (2005, May). Multiobjective ga optimization using reduced models. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *35*(2), 261–265.

Chan, Z., Kasabov, N., & Collins, L. (2006, January). A two-stage method-

ology for gene regulatory network extraction from time-course gene expression data. *Expert Syst. Appl.*, *30*, 59–63.

Charalampidis, D., & Muldrey, B. (2009). Clustering using multilayer perceptrons. *Nonlinear Analysis: Theory, Methods & Applications*, *71*(12), e2807–e2813.

Chen, S., & Poon, S. (2007, October). Modelling international stock market contagion using copula and risk appetite [Working Paper].

Chiang, T., & Doong, S. (2001, November). Empirical analysis of stock returns and volatility: Evidence from seven asian stock markets based on TAR-GARCH model. *Review of Quantitative Finance and Accounting*, *17*(3), 301–18.

Choi, D., & Roy, B. (2006, April). A generalized kalman filter for fixed point approximation and efficient temporal-difference learning. *Discrete Event Dynamic Systems*, *16*, 207–239.

Chowdhury, A. (1994). Stock market interdependencies: Evidence from the asian NIEs. *Journal of Macroeconomics*, *16*(4), 629–651.

Christou, C., & Papageorgiou, E. (2007). A framework of mathematics inductive reasoning. *Learning and Instruction*, *17*(1), 55–66.

Clout, V., & Willett, R. (2009). Investigating the relationship between market values and accounting numbers for long-lived companies [Working Paper Series]. *SSRN eLibrary*.

Collins, D., & Biekpe, N. (2003, March). Contagion and interdependence in african stock markets. *South African Journal of Economics*, *71*(1), 181–194.

Costa, M., Filippi, E., & Pasero, E. (2005). Multi-layer perceptron ensembles for pattern recognition: some experiments. In *IEEE international conference on IEEE world congress on computational intelligence, neural networks 1994* (Vol. 7, pp. 4232–4236). IEEE Press.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector*

machines: and other kernel-based learning methods. New York, NY, USA: Cambridge University Press.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, *2*, 303–314.

Davidson, E. (2006). *The regulatory genome: gene regulatory networks in development and evolution*. Academic Press.

Davidson, E., & Erwin, D. (2006). Gene regulatory networks and the evolution of animal body plans. *Science*, *311*(5762), 796-800.

DeLong, J. (2002). The phillips curve and expectations. In *Macroeconomics*. McGraw-Hill.

Drew, M., & Chong, L. (2002, February). *Stock market interdependence: Evidence from australia* (School of Economics and Finance Discussion Papers and Working Papers Series No. 106). School of Economics and Finance, Queensland University of Technology.

Dudani, S. (1976, April). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, *SMC-6*(4), 325–327.

El Dajani, R., Miquel, M., Maison-Blanche, P., & Rubel, P. (2003, April). Time series prediction using parametric models and multilayer perceptrons: case study on heart signals, ICASSP '03. In (Vol. 2, p. II-773-6).

Federal Reserve Statistical Release. (2011, March). *H.15 daily interest rates for selected u.s. treasury and private money market and capital market instruments*. Available from `http://www.federalreserve.gov/releases/h15/data.htm`

Forbes, K., & Rigobon, R. (1999, July). *No contagion, only interdependence: Measuring stock market co-movements* (Working Paper No. 7267). National Bureau of Economic Research.

Freund, R., William, J., & Sa, P. (2006). *Regression analysis: statistical modeling of a response variable*. Academic Press.

Frieβ, T.-T., & Harrison, R. (1999). A kernel-based adaline for function approximation. *Intelligent Data Analysis*, *3*(4), 307–313.

Fung, G., & Mangasarian, O. (2001). Proximal support vector machine classifiers. In *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, KDD'01* (pp. 77–86). New York, NY, USA: ACM.

Georgiev, A. (1988). Asymptotic properties of the multivariate nadaraya-watson regression function estimate: The fixed design case. *Statistics & Probability Letters*, *7*(1), 35–40.

Gerstner, W., & Kistler, W. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press.

Glezakos, M., Merika, A., & Kaligosfyris, H. (2007). Interdependence of majorworld stock exchanges: How is the athens stock exchange affected? *International Research Journal of Finance and Economics*.

Gorr, W., Nagin, D., & Szczypula, J. (1994, June). Comparative study of artificial neural network and statistical models for predicting student grade point averages. *International Journal of Forecasting*, *10*(1), 17–34.

Govindarajan, M., & Chandrasekaran, R. (2010). Evaluation of k-nearest neighbor classifier performance for direct marketing. *Expert Systems with Applications*, *37*(1), 253–258.

Hagan, M., Demuth, H., & Beale, M. (1995). *Neural network design*. PWS Pub. Co.

Hamilton, J. (1994). *Time series analysis*. Princeton University Press.

Han, M., Fan, M., & Xi, J. (2005). Study of nonlinear multivariate time series prediction based on neural networks. In J. Wang, X. Liao, & Z. Yi (Eds.), *Lecture Notes in Computer Science, advances in neural networks, ISNN 2005* (Vol. 3497, p. 815-815). Springer Berlin / Heidelberg.

Hartikainen, J., & Sarkka, S. (2010, September). Kalman filtering and

smoothing solutions to temporal gaussian process regression models. In *IEEE international workshop on machine learning for signal processing (MLSP), 2010* (pp. 379–384).

Hastie, T., Tibshirani, R., & Friedman, J. (2003). *The elements of statistical learning: Data mining, inference, and prediction.* Springer.

Helfenstein, U. (2005). Arma and arima models. In *Encyclopedia of biostatistics.* John Wiley & Sons, Ltd.

Hinojosa, V., & Hoese, A. (2010, February). Short-term load forecasting using fuzzy inductive reasoning and evolutionary algorithms. *IEEE Transactions on Power Systems*, *25*(1), 565–574.

Ho, S., Xie, M., & Goh, T. (2002). A comparative study of neural network and box-jenkins ARIMA modeling in time series prediction. *Computers & Industrial Engineering*, *42*(2-4), 371–375.

Ho, Y., & Lee, R. (1965). Identification of linear dynamic systems. *Information and Control*, *8*(1), 93–110.

Holland, J., Holyoak, K., Nisbett, R., & Thagard, P. (1989). *Induction–processes of inference, learning, and discovery.* Cambridge, MA, USA: Cambridge University Press.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Hothorn, T., & Lausen, B. (2005). Bundling classifiers by bagging trees. *Computational Statistics & Data Analysis*, *49*(4), 1068–1078.

Hui, S., & Zak, S. (1994, November). The widrow-hoff algorithm for mcculloch-pitts type neurons. *IEEE Transactions on Neural Networks*, *5*(6), 924–929.

Hwang, Y. (2009). *Local and personalized models for prediction, classification and knowledge discovery on real world data modelling problems.* Unpublished doctoral dissertation, Auckland University of Technology, New Zealand.

Isa, N., & Mamat, M. (2011). Clustered-hybrid multilayer perceptron network for pattern recognition application. *Applied Soft Computing*, *11*(1), 1457–1466.

Isakov, D., & Parignon, C. (2000, June). On the dynamic interdependence of international stock markets: A swiss perspective. *Swiss Journal of Economics and Statistics (SJES)*, *136*(II), 123–146.

Islam, M., Yao, X., Shahriar Nirjon, S., Islam, M., & Murase, K. (2008). Bagging and boosting negatively correlated neural networks. *IEEE transactions on systems man and cybernetics Part B Cybernetics*, *38*(3), 771–784.

Jang, J., Sun, C., & Mizutani, E. (1997). *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*. Prentice Hall.

Jang, J.-S. (1993, May). ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics*, *23*(3), 665–685.

Jang, J.-S., & Sun, C.-T. (1995, March). Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, *83*(3), 378–406.

Jeen-Shang, L., & Yigong, Z. (1994). Nonlinear structural identification using extended kalman filter. *Computers & Structures*, *52*(4), 757–764.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the sixteenth international conference on machine learning, ICML '99* (pp. 200–209). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *International conference on machine learning (ICML)* (pp. 290–297).

Jones, D., Plank, M., & Sleeman, B. (2009). *Differential equations and mathematical biology, second edition*. CRC Press.

Jordan, M., & Jacobs, R. (1994, March). Hierarchical mixtures of experts

and the EM algorithm. *Neural Comput.*, *6*, 181–214.

Kahla, M., Faraj, Z., Castanie, F., & Hoffmann, J. (1994). Multi-layer adaptive filters trained with back propagation: A statistical approach. *Signal Processing*, *40*(1), 65–85.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*(82 (Series D)), 35–45.

Kasabov, N. (1996). *Foundation of neural networks, fuzzy systems, and knowledge engineering.* MIT Press.

Kasabov, N. (2001). Evolving fuzzy neural networks for supervised/unsupervised on-line knowledge-based learning. *IEEE Transactions on Systems, Man and Cybernetics*, *31*, 902–918.

Kasabov, N. (2007a). *Evolving connectionist systems, second edition.* Springer.

Kasabov, N. (2007b). Global, local and personalised modeling and pattern discovery in bioinformatics: An integrated approach. *Pattern Recognition Letters*, *28*(6), 673–685.

Kasabov, N. (2009). Integrative probabilistic evolving spiking neural networks utilising quantum inspired evolutionary algorithm: a computational framework. In *Proceedings of the 15th international conference on advances in neuro-information processing - volume part I, ICONIP '09* (pp. 3–13). Berlin, Heidelberg: Springer-Verlag.

Kasabov, N. (2010, January). To spike or not to spike: A probabilistic spiking neuron model. *Neural Netw.*, *23*, 16–19.

Kasabov, N., Chan, Z., Jain, V., Sidorov, I., & Dimitrov, D. (2004). Gene regulatory network discovery from time-series gene expression data: a computational intelligence approach. In *Lecture Notes in Computer Science* (Vol. 3316, pp. 1333–1353). Springer Berlin / Heidelberg.

Kasabov, N., & Pang, S. (2003, December). Transductive support vector

machines and applications in bioinformatics for promoter recognition. In *Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, 2003, IEEE press* (Vol. 1, pp. 1–6).

Kasabov, N., & Song, Q. (2002, April). DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems*, *10*, 144–154.

Kim, H., Pang, S., Je, H., Kim, D., & Bang, S. (2002). Support vector machine ensemble with bagging. In S. Lee & A. Verri (Eds.), *Lecture Notes in Computer Science, pattern recognition with support vector machines* (Vol. 2388, pp. 131–141). Springer Berlin / Heidelberg.

Kim, K., Lee, J., & Park, C. (2009, January). Adaptive two-stage extended kalman filter for a fault-tolerant INS-GPS loosely coupled system. *IEEE Transactions on Aerospace and Electronic Systems*, *45*(1), 125–137.

Kim, R., Ji, H., & Wong, W. (2006). An improved distance measure between the expression profiles linking co-expression and co-regulation in mouse. *BMC Bioinformatics*, *7*, 1–8.

Kim, T., & Adali, T. (2003, July). Approximation by fully complex multilayer perceptrons. *Neural Comput.*, *15*, 1641–1666.

Kirchgässner, G., & Wolters, J. (2007). *Introduction to modern time series analysis*. Springer.

Komogortsev, O., & Khan, J. (2008). Eye movement prediction by kalman filter with integrated linear horizontal oculomotor plant mechanical model. In *Proceedings of the 2008 symposium on eye tracking research & applications, ETRA '08* (pp. 229–236). New York, NY, USA: ACM.

Koskela, T., Lehtokangas, M., Saarinen, J., & Kaski, K. (1996). Time series prediction with multilayer perceptron, FIR and elman neural networks. In *In proceedings of the world congress on neural networks* (pp. 491–496). Press.

Kukar, M. (2003). Transductive reliability estimation for medical diagnosis.

Artificial Intelligence in Medicine, *29*(1-2), 81–106. (Artificial Intelligence in Medicine Europe AIME '01)

Labonte, G. (2002). Adaline neural network for online self-learning and adaptive control of a vehicle with thrusters in a fluid. In *Proceedings of IEEE international symposium on intelligent control, 2002* (pp. 258–265).

Lawrence, C., & Reilly, A. (1990). An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *PubMed*, *7*(1), 41–51.

Lee, H., & Tsoi, A. (1995). Application of multi-layer perceptron in estimating speech/noise characteristics for speech recognition in noisy environment. *Speech Communication*, *17*(1-2), 59–76.

Lee, S., Lim, J., Baek, S., & Sung, K. (1999). Time-varying signal frequency estimation by VFF kalman filtering. *Signal Processing*, *77*(3), 343–347.

Lee, Y., Lieberman, M., Lichtenberg, A., Bose, F., Baltes, H., & Patrick, R. (1997, January). Global model for high pressure electronegative radio-frequency discharges. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, *15*, 113–126.

Lei, Z., Yang, Y., & Wu, Z. (2006, May). Ensemble of support vector machine for text-independent speaker recognition. *International Journal of Computer Science and Network Security*, *6*(5), 163–167.

Levine, M., & Davidson, E. (2005). Gene regulatory networks for development. In *Proceedings of the national academy of sciences of the united states of america* (Vol. 102, pp. 4936–4942). National Academy of Sciences.

Li, C., & Yuen, P. (2001). Transductive learning: Learning iris data with two labeled data. In G. Dorffner, H. Bischof, & K. Hornik (Eds.), *Lecture Notes in Computer Science, artificial neural networks, ICANN 2001* (Vol. 2130, pp. 231–236). Springer Berlin / Heidelberg.

Li, F., & Wechsler, H. (2004). Watch list face surveillance using transductive inference. In D. Zhang & A. Jain (Eds.), *Lecture Notes in Computer*

Science, biometric authentication (Vol. 3072, pp. 1–15). Springer Berlin / Heidelberg.

Li, J., & Chua, C. (2003, September). Transductive inference for color-based particle filter tracking. In *Proceedings of international conference on image processing, ICIP 2003* (Vol. 3, p. III-949-52).

Li, R., Bhanu, B., & Dong, A. (2008, May). Feature synthesized EM algorithm for image retrieval. *ACM Trans. Multimedia Comput. Commun. Appl.*, *4*, 10:1–10:24.

Li, X., Chen, H., Li, J., & Zhang, Z. (2010, January). Gene function prediction with gene interaction networks: A context graph kernel approach. *IEEE Transactions on Information Technology in Biomedicine*, *14*(1), 119–128.

Lin, I., & Liou, C. (2007). Least-mean-square training of cluster-weighted modeling. In *Proceedings of the 17th international conference on artificial neural networks, ICANN '07* (pp. 301–310). Berlin, Heidelberg: Springer-Verlag.

Lin, T., Yeh, C., & Liu, M. (2010). Application of svm-based filter using lms learning algorithm for image denoising. In *Proceedings of the 17th international conference on neural information processing: models and applications - volume part II, ICONIP '10* (Vol. 6444, pp. 82–90). Berlin, Heidelberg: Springer-Verlag.

Lin, X., Yang, Z., & Song, Y. (2009). Short-term stock price prediction based on echo state networks. *Expert Systems with Applications*, *36*(3, Part 2), 7313–7317.

Liu, B., & Liu, J. (2002, February). Multivariate time series prediction via temporal classification. In *Proceedings of 18th international conference on data engineering, 2002* (p. 268).

Liu, Y., & Sun, L. (2008). Analysis of cointegration between macroeconomic variables and stock index. In *Proceedings of the 2008 fourth international*

conference on natural computation (Vol. 5, pp. 318–322). Washington, DC, USA: IEEE Computer Society.

Lofting, C., Dougherty, M., & Southwood, D. (1997). The coriolis effect in a rapidly rotating magnetosphere. *Advances in Space Research*, *20*(2), 239–242.

Lucey, B., & Muckley, C. (2010). Global stock market interdependencies and long-term portfolio diversification [Working Paper Series]. *SSRN eLibrary*.

Lucks, M., & Oki, N. (1999). A radial basis function network (RBFN) for function approximation. In *Proceedings of the 42nd midwest symposium on circuits and systems* (Vol. 2, pp. 1099–1101).

Lukmanto, L., Widiputra, H., & Lukas. (2009). Dynamic interaction network to model the interactive patterns of international stock markets. *World Academy of Science, Engineering and Technology*, *59*, 257–261.

Maass, W., & Bishop, C. (2001). *Pulsed neural networks*. MIT Press.

Mailier, P. (2010). Can we trust long-range weather forecasts? In A. Troccoli (Ed.), *Management of weather and climate risk in the energy industry* (pp. 227–239). Springer Netherlands.

Mankiw, N. (2006). *Macroeconomics, 6th edition*. Worth Publishers.

Marin, F., Garcia-Lagos, F., Joya, G., & Sandoval, F. (2002, March). Global model for short-term load forecasting using artificial neural networks. *IEEE Proceedings on Generation, Transmission and Distribution*, *149*(2), 121–125.

Marinaro, M., & Scarpetta, S. (2000). On-line learning in RBF neural networks: a stochastic approach. *Neural Networks*, *13*(7), 719–729.

Masih, A., & Masih, R. (2001). Dynamic modeling of stock market interdependencies: An empirical investigation of australia and the asian NICs. *Review of Pacific Basin Financial Markets and Policies*, *4*(2), 235–264.

Maybeck, P. (1972, June). The kalman filter - an introduction for potential

users. *Air Force Flight Dynamics Laboratory*, *TM-72-3*.

Maybeck, P. (1979). *Stochastic models, estimation, and control, volume 1.* Academic Press.

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133.

Medeiros, A., Amaral, W., & Campello, R. (2006, October). Ga optimization of obf ts fuzzy models with linear and non linear local models. In *Ninth brazilian symposium on neural networks, sbrn '06* (pp. 66–71).

Minsky, M., & Papert, S. (1969). *Perceptrons: an introduction to computational geometry.* MIT Press.

Mitchell, T. (1997). *Machine learning.* McGraw-Hill Science.

Moiseenko, V., & Saenko, O. (1994). The use of the kalman smoothing algorithm for the analysis and processing of oceanographic data. statement and methods of solution. *Physical Oceanography*, *5*, 35–42.

Mukherjee, S., Chen, Z., & Gangopadhyay, A. (2006, November). A privacy-preserving technique for euclidean distance-based mining algorithms using fourier-related transforms. *The VLDB Journal*, *15*, 293–315.

Müller, K., Smola, A., Rätsch, G., Schökopf, B., Kohlmorgen, J., & Vapnik, V. (1999). Using support vector machines for time series prediction. In (pp. 243–253). Cambridge, MA, USA: MIT Press.

Nadaraya, E. (1964). On estimating regression. *Theory of Prob. and Appl.*, *9*, 141–142.

Nanni, L., & Lumini, A. (2006). Fuzzybagging: A novel ensemble of classifiers. *Pattern Recognition*, *39*(3), 488–490.

National Insitute of Weather and Atmosphere. (2010). *The National Climate Database.* Available from `http://cliflo.niwa.co.nz/`

Nguyen, M., Abbass, H., & McKay, R. (2008, January). Analysis of CCME: Coevolutionary dynamics, automatic problem decomposition, and regularization. *IEEE Transactions on Systems, Man, and Cybernetics, Part*

C: Applications and Reviews, *38*, 100–109.

Okunev, J., Wilson, P., & Zurbruegg, R. (2000). The causal relationship between real estate and stock markets. *The Journal of Real Estate Finance and Economics*, *21*, 251–261.

Pang, S. (2004, July). SVM classification tree algorithm with application to face membership authentication. In *Proceedings of IEEE international joint conference on neural networks 2004, IJCNN '04* (Vol. 1, p. 436).

Pankratz, A. (1983). *Forecasting with univariate box - jenkins models: Concepts and cases*. Wiley.

Phylaktis, K., & Ravazzolo, F. (2005, April). Stock market linkages in emerging markets: implications for international portfolio diversification. *Journal of International Financial Markets, Institutions and Money*, *15*(2), 91–106.

Platelt, M., Schliebs, S., & Kasabov, N. (2007, September). A versatile quantum-inspired evolutionary algorithm. In *Ieee congress on evolutionary computation, cec 2007* (pp. 423–430).

Poggio, F. (1994). Regularization theory, radial basis functions and networks. In *From statistics to neural networks: Theory and pattern recognition applications* (pp. 83–104). NATO ASI Series.

Popescu, M., Balas, V., Perescu-Popescu, L., & Mastorakis, N. (2009, July). Multilayer perceptron and neural networks. *WSEAS Trans. Cir. and Sys.*, *8*, 579–588.

Proedrou, K., Nouretdinov, I., Vovk, V., & Gammerman, A. (2002). Transductive confidence machines for pattern recognition. In T. Elomaa, H. Mannila, & H. Toivonen (Eds.), *Lecture Notes in Computer Science, machine learning: ECML 2002* (Vol. 2430, pp. 221–231). Springer Berlin / Heidelberg.

Psillaki, M., & Margaritis, D. (2008). Long-run interdependence and dynamic linkages in international stock markets: Evidence from france germany

and the U.S. *Journal of Money, Investment and Banking*.

Qian, G., Sural, S., Gu, Y., & Pramanik, S. (2004). Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on applied computing, SAC '04* (pp. 1232–1237). New York, NY, USA: ACM.

Reinsel, G. (1997). Vector arma time series models and forecasting. In *Elements of multivariate time series analysis.* Springer.

Remy, E., & Thiel, E. (2005, February). Exact medial axis with euclidean distance. *Image Vision Comput.*, *23*, 167–175.

Roche, A., Malandain, G., Pennec, X., & Ayache, N. (1998). The correlation ratio as a new similarity measure for multimodal image registration. In W. Wells, A. Colchester, & S. Delp (Eds.), *Lecture Notes in Computer Science, medical image computing and computer-assisted interventation, MICCAI'98* (Vol. 1496, pp. 1115–1124). Springer Berlin / Heidelberg.

Rodrigues, P., Gama, J., & Pedroso, J. (2008, May). Hierarchical clustering of time-series data streams. *IEEE Trans. on Knowl. and Data Eng.*, *20*, 615–627.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–408.

Rossi, F., & Conan-Guez, B. (2005, January). Functional multi-layer perceptron: a non-linear tool for functional data analysis. *Neural Netw.*, *18*, 45–60.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In (pp. 318–362). Cambridge, MA, USA: MIT Press.

Russell, S., & Norvig, P. (1995). *Artificial intelligence a modern approach, second edition.* Prentice Hall.

Saad, E., Prokhorov, D., & Wunsch, D. (1998, November). Comparative study

of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks*, *9*(6), 1456–1470.

Sapankevych, N., & Sankar, R. (2009, May). Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, *4*(2), 24–38.

Sarkka, S., Vehtari, A., & Lampinen, J. (2007). CATS benchmark time series prediction by kalman smoother with cross-validated noise density. *Neurocomputing*, *70*(13-15), 2331–2341.

Sartori, M., Passino, K., & Antsaklis, P. (1992, June). A multilayer perceptron solution to the match phase problem in rule-based artificial intelligence systems. *IEEE Trans. on Knowl. and Data Eng.*, *4*, 290–297.

Schliebs, S., Nuntalid, N., & Kasabov, N. (2010). Towards spatio-temporal pattern recognition using evolving spiking neural networks. In K. Wong, B. Mendis, & A. Bouzerdoum (Eds.), *Lecture Notes in Computer Science, neural information processing. theory and algorithms* (Vol. 6443, pp. 163–170). Springer Berlin / Heidelberg.

Schliebs, S., Platel, M., Worner, S., & Kasabov, N. (2009, June). Quantum-inspired feature and parameter optimisation of evolving spiking neural networks with a case study from ecological modeling. In *International joint conference on neural networks, ijcnn 2009* (pp. 2833–2840).

Scholz, M., & Klinkenberg, R. (2007, January). Boosting classifiers for drifting concepts. *Intell. Data Anal.*, *11*, 3–28.

Serguieva, A., Kalganova, T., & Khan, T. (2003). An intelligent system for risk classification of stock investment projects. *Journal of Applied Systems Studies*, *4*(2), 236–261.

Serguieva, A., & Wu, H. (2008). Computational intelligent in financial contagion analysis. *International Journal on Complex Systems*, *2229*, 1–12.

Shabri, M., Kameel, A., & Azmi, M. (2008). Interdependence of ASEAN-

5 stock markets from the US and Japan. *Global Economic Review: Perspectives on East Asian Economies and Industries*, *37*(2), 201–225.

Shang, F. (2008). An estimating traffic scheme based on adaline. In F. Sun, J. Zhang, Y. Tan, J. Cao, & W. Yu (Eds.), *Lecture Notes in Computer Science, advances in neural networks - ISNN 2008* (Vol. 5264, pp. 632–641). Springer Berlin / Heidelberg.

Shapiai, M., Ibrahim, Z., Khalid, M., Jau, L., & Pavlovich, V. (2010). A non-linear function approximation from small samples based on nadaraya-watson kernel regression. *International Conference on Computational Intelligence, Communication Systems and Networks*, *0*, 28–32.

Shiblee, M., Kalra, P., & Chandra, B. (2009). Time series prediction with multilayer perceptron (MLP): A new generalized error based approach. In M. Koppen, N. Kasabov, & G. Coghill (Eds.), *Lecture Notes in Computer Science, advances in neuro-information processing* (Vol. 5507, pp. 37–44). Springer Berlin / Heidelberg.

Shoenauer, M., & Xanthakis, S. (1993). Constrained ga optimization. In *Proceedings of the 5th international conference on genetic algorithms* (pp. 573–580). Morgan Kaufmann Publishers Inc.

Sichtig, H., Schaffer, J., & Riva, A. (2010, July). Evolving spiking neural networks for predicting transcription factor binding sites. In *The 2010 international joint conference on neural networks (ijcnn)* (pp. 1–8).

Simaan, M., Ferreira, A., Chen, S., Antaki, J., & Galati, D. (2009, January). A dynamical state space representation and performance analysis of a feedback-controlled rotary left ventricular assist device. *IEEE Transactions on Control Systems Technology*, *17*(1), 15–28.

Soltic, S., Wysoski, S., & Kasabov, N. (2008, June). Evolving spiking neural networks for taste recognition. In *Ieee international joint conference on neural networks, ijcnn 2008. (ieee world congress on computational intelligence)* (pp. 2091–2097).

Song, F., & Smith, S. (2000). Using least mean square learning error to improve takagi-sugeno type fuzzy logic controller optimization. In *19th international conference of the north american fuzzy information processing society, NAFIPS '00* (pp. 475–479).

Song, Q., & Kasabov, N. (2001). ECM - a novel on-line, evolving clustering method and its applications. In M. Posner (Ed.), *Foundations of cognitive science* (pp. 631–682). The MIT Press.

Song, Q., & Kasabov, N. (2004). TWRBF–transductive RBF neural network with weighted data normalization. In N. Pal, N. Kasabov, R. Mudi, S. Pal, & S. Parui (Eds.), *Lecture Notes in Computer Science, neural information processing* (Vol. 3316, pp. 633–640). Springer Berlin / Heidelberg.

Song, Q., & Kasabov, N. (2005, December). NFI: a neuro-fuzzy inference method for transductive reasoning. *IEEE Transactions on Fuzzy Systems*, *13*(6), 799–808.

Song, Q., & Kasabov, N. (2006). TWNFI – a transductive neuro-fuzzy inference system with weighted data normalization for personalized modeling. *Neural Networks*, *19*(10), 1591–1596.

Song, Q., Ma, T., & Kasabov, N. (2006). TTLSC–transductive total least square model for classification and its application in medicine. In X. Li, O. Zaiane, & Z. Li (Eds.), *Lecture Notes in Computer Science, advanced data mining and applications* (Vol. 4093, pp. 197–204). Springer Berlin / Heidelberg.

Soucy, P., & Mineau, G. (2001). A simple KNN algorithm for text categorization. In *Proceedings IEEE international conference on data mining, ICDM 2001* (pp. 647–648).

Sun, X., Tian, Y., & Wang, W. (2008, September). The application of kalman filtering to corporate bankruptcy prediction. In *15th annual conference proceedings of international conference on management science and en-*

gineering, ICMSE 2008 (pp. 670–676).

Takagi, T., & Sugeno, M. (1985, February). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, *15*(1), 116–132.

Tan, S. (2005). Neighbor-weighted K-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, *28*(4), 667–671.

Teixeira, R., Braga, A., & Menezes, B. (2000, August). Control of a robotic manipulator using artificial neural networks with on-line adaptation. *Neural Process. Lett.*, *12*, 19–31.

Teo, K., Wang, L., & Lin, Z. (2001). Wavelet packet multi-layer perceptron for chaotic time series prediction: Effects of weight initialization. In V. Alexandrov, J. Dongarra, B. Juliano, R. Renner, & C. Tan (Eds.), *Lecture Notes in Computer Science, ICCS 2001* (Vol. 2074, pp. 310–317). Springer Berlin / Heidelberg.

The Federal Reserve Archival System for Economic Research. (2011, March). *Economic report of the president: Downloadable reports/tables.* GPO Access. Available from `http://www.gpoaccess.gov/eop/download.html`

Thissen, U., Brakel, R. van, Weijer, A. de, Melssen, W., & Buydens, L. (2003). Using support vector machines for time series prediction. *Chemometrics and Intelligent Laboratory Systems*, *69*(1-2), 35–49.

Tomic, W. (1995). Training in inductive reasoning and problem solving. *Contemporary Educational Psychology*, *20*(4), 483–490.

Tong, S., & Chang, E. (2001). Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on multimedia, MULTIMEDIA '01* (pp. 107–118). New York, NY, USA: ACM.

Trefil, J. (2003). *The nature of science: An A-Z guide to the laws and principles governing our universe.* Houghton Mifflin Harcourt.

Tsai, C., & Kurz, L. (1983). An adaptive robustizing approach to kalman filtering. *Automatica*, *19*(3), 279–288.

Vapnik, V. (1998). *Statistical learning theory*. Wiley-Interscience.

Verbeek, M. (2008). *A guide to modern econometrics*. John Wiley & Sons.

Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, *20*(3), 391–403.

Vitousek, P. (1992). Global environmental change: An introduction. *Annual Review of Ecology and Systematics*, *23*, 1–14.

Vitousek, P. (1994). Beyond global warming: Ecology and global change. *Ecology*, *75*(7), 1861–1876.

Vreeken, J. (2003). *Spiking neural networks, an introduction* (Tech. Rep.).

Vu, K. (2007). *Optimal discrete control theory: The rational function structure model*. AuLac Technologies Inc.

Wang, L., Liu, B., & Wang, R. (2010, December). Comparative investigation of bp and rbf neural network on seismic damage prediction of multi-story brick buildings. In *2nd international conference on information engineering and computer science, ICIECS '10* (pp. 1–4).

Wang, L., Zhang, Y., & Feng, J. (2005, August). On the euclidean distance of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 334–1339.

Wang, Z., Yang, F., Ho, D., Swift, S., Tucker, A., & Liu, X. (2008, March). Stochastic dynamic modeling of short gene expression time-series data. *IEEE Transactions on NanoBioscience*, *7*(1), 44–55.

Watson, G. (1969). Smooth regression analysis. *Sankhya, Series*, *A*(26), 359–372.

Wei, W. (2005). *Time series analysis : Univariate and multivariate methods (2nd edition)*. Addison Wesley.

Weisberg, S. (2005). *Applied linear regression, 3rd edition*. John Wiley and

Sons.

Welch, G., & Bishop, G. (1995). *An introduction to the kalman filter* (Tech. Rep.). Chapel Hill, NC, USA.

Welling, M. (2001). *The kalman filter* (Tech. Rep.). Pasadena, CA, USA.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Unpublished doctoral dissertation, Harvard University.

West, M., Blanchette, C., Dressman, H., Huang, E., Ishida, S., Spang, R., et al. (2001). Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, *98*(20), 11462-11467.

Weston, J., Perez-Cruz, F., Bousquet, O., Chapelle, O., Elisseeff, A., & Scholkopf, B. (2003). Feature selection and transduction for prediction of molecular bioactivity for drug design. *Bioinformatics*, *19*(6), 764-771.

Whittle, J., & Schumann, J. (2004, December). Automating the implementation of kalman filter algorithms. *ACM Trans. Math. Softw.*, *30*, 434–453.

Widiputra, H., Kho, H., Lukas, Pears, R., & Kasabov, N. (2009). A novel evolving clustering algorithm with polynomial regression for chaotic time-series prediction. In C. Leung, M. Lee, & J. Chan (Eds.), *Lecture Notes in Computer Science, neural information processing* (Vol. 5864, pp. 114–121). Springer Berlin / Heidelberg.

Widiputra, H., Pears, R., & Kasabov, N. (2009). Personalised modelling for multiple time-series data prediction: a preliminary investigation in asia pacific stock market indexes movement. In *Proceedings of the 15th international conference on advances in neuro-information processing - volume part I, ICONIP '08* (Vol. 5506, pp. 1237–1244). Berlin, Heidelberg: Springer-Verlag.

Widiputra, H., Pears, R., & Kasabov, N. (2011a). Dynamic interaction net-

works versus local trend models for multiple time-series prediction. *Cybernetics and Systems*, *42*, 1–24.

Widiputra, H., Pears, R., & Kasabov, N. (2011b). Kalman filter to estimate dynamic and important patterns of interaction between multiple variables. In J. Gomez (Ed.), *Kalman filtering.* Nova Science New York.

Widiputra, H., Pears, R., & Kasabov, N. (2011c). Multiple time-series prediction through multiple time-series relationships profiling and clustered recurring trends. In *Proceedings of the pacific asia cconference on knowledge discovery and data mining 2011, PAKDD '11.*

Widiputra, H., Pears, R., Serguieva, A., & Kasabov, N. (2009). Dynamic interaction networks in modelling and predicting the behaviour of multiple interactive stock markets. *Intelligent Systems in Accounting, Finance and Management*, *16*, 189–205.

Widrow, B., & Hoff, M. (1960). Adaptive switching circuits. In *IRE WESCON convention record, part 4* (pp. 96–104). New York: IRE.

Widrow, B., & Lehr, M. (1993). Adaptive neural networks and their applications. *International Journal of Intelligent Systems*, *8*, 453–507.

Wooldridge, J. (2006). *Introductory econometrics: a modern approach, 3rd edition.* Thomson South-Western.

Wu, D., Bennett, K., Cristianini, N., & Shawe-Taylor, J. (1999). Large margin trees for induction and transduction. In *Proceedings of the sixteenth international conference on machine learning, ICML '99* (pp. 474–483). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Wysoski, S., Benuskova, L., & Kasabov, N. (2010, September). Evolving spiking neural networks for audiovisual information processing. *Neural Netw.*, *23*, 819–835.

Yahoo! Finance. (2010). *Major World Indices-Asia/Pacific.* Available from `http://finance.yahoo.com/intlindices?e=asia`

Yamada, T., Yamashita, K., Ishii, N., & Iwata, K. (2006, June). Text clas-

sification by combining different distance functions with weights. In *Seventh ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing, SNPD 2006* (pp. 85–90).

Yang, H., Chan, L., & King, I. (2002). Support vector machine regression for volatile stock market prediction. In *Third international conference on intelligent data engineering and automated learning, IDEAL '02* (pp. 391–396). Springer.

Yang, S., Ho, C., & Lee, C. (2006, March). HBP: improvement in BP algorithm for an adaptive MLP decision feedback equalizer. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *53*(3), 240–244.

Yao, X., & Liu, Y. (1996, May). Ensemble structure of evolutionary artificial neural networks. In *Proceedings of IEEE international conference on evolutionary computation 1996* (pp. 659–664).

Yao, X., & Liu, Y. (1998, June). Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *28*, 417–425.

Zadeh, L. (1965, June). Fuzzy sets. *Information and Control*, *28*(3), 338–353.

Zadeh, L. (1973, January). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, *SMC-3*(1), 28–44.

Zhang, C., & Zhang, J. (2008). A local boosting algorithm for solving classification problems. *Computational Statistics & Data Analysis*, *52*(4), 1928–1941.

Zhang, D., & Ionescu, D. (2009, February). A new method for measuring packet loss probability using a kalman filter. *IEEE Transactions on Instrumentation and Measurement*, *58*(2), 488–499.

Zhang, X., Hang, C., Tan, S., & Wang, P. (1994, June-July). The delta rule and learning for min-max neural networks. In *IEEE international*

conference on neural networks, 1994 - IEEE world congress on computational intelligence, 1994 (Vol. 1, pp. 38–43).

Zhanggui, Z., Yau, H., & Fu, A. (1999, July). A new stock price prediction method based on pattern classification. In *International joint conference on neural networks, 1999. IJCNN '99* (Vol. 6, pp. 3866–3870).

Zhou, Z., & Jiang, Y. (2003, March). Medical diagnosis with C4.5 rule preceded by artificial neural network ensemble. *IEEE Transactions on Information Technology in Biomedicine*, *7*, 37–42.

Zivot, E., & Wang, J. (2006). Vector autoregressive models for multivariate time series. In *Modeling financial time series with s-plus, 2nd edition.* Springer.

Zou, B., & Umugwaneza, M. (2008). Shape-based trademark retrieval using cosine distance method. In *Proceedings of the 2008 eighth international conference on intelligent systems design and applications, ISDA '08* (Vol. 2, pp. 498–504). Washington, DC, USA: IEEE Computer Society.