Hybrid HMM and Soft Computing modeling with applications to time series analysis

Md. Rafiul Hassan

Submitted in total fulfilment of
the requirements for the degree of

Doctor of Philosophy

Department of Computer Science and Software Engineering
The University of Melbourne
Australia

August, 2007

The University of Melbourne
Australia

Abstract

Hybrid HMM and Soft Computing modeling with applications to time series analysis

Md. Rafiul Hassan

The Hidden Markov Model is a statistical model which assumes that the system to be modelled is a Markov process with unknown parameters. It is a challenge to compute the values of the hidden parameters, given that some sequence of output is observable.

HMMs have been successfully used in many pattern recognition domains including speech recognition, bioinformatics and robotics movement navigation. To build an HMM, initial parameter values for observation emission probabilities must be inferred from some distribution. The model is then trained to update the state transition probabilities, initial state probability, and also observation emission probabilities. In recent years, soft computing techniques, e.g. Artificial Neural Network (ANN), Fuzzy Logic (FL) and Evolutionary Approaches (EA) e.g., Genetic Algorithm (GA), have been used to select optimal initial parameter values. The purpose of this hybridization process is to overcome limitations of the individual techniques.

In this thesis, we introduce a novel method of hybridizing HMM to solve a real-world problem of time series forecasting. In addition to using soft computing to improve the initialization of parameter values, we use the HMM's output to identify similar data patterns, which are then used for clustering and forecasting mechanisms.

Initially, we use a single HMM to cluster a dataset. After training, the HMM is used in identifying similar data patterns in an unsupervised manner. Similar data items are then allocated to clusters. It is important to note that the number of clusters is not known a priori. We evaluate the performance of the model using a number of benchmark datasets with labels.

Next, we develop an HMM-based financial forecasting method. This consists of two components - training an HMM, and preparing a forecast using this HMM. The HMM is hybridized with an ANN and a GA to optimize the initial parameter values and to

find similar data patterns from the amplitude of the time series. This contrasts with the traditional approach of HMM, which is to detect data patterns or trends over time. The forecasting mechanism consists of two alternative approaches: a weighted average technique, and an interpolation technique. We interpolate the neighbouring values within the groups of identified similar data patterns to that of the current one to calculate the forecast value. In the second approach, the value differences between the matched days and the respective next days are calculated and the weighted average of the difference values is added to the current value to obtain the corresponding forecast. Performance of the developed model using the forecast error information is compared with that of ANNs and other traditional statistical approaches. We examine the performance of the forecast models using financial time series data collected from Australian Securities Exchange (ASX).

Finally, we introduce a hybrid HMM-Fuzzy Inference forecasting method to further improve on our forecasts. To achieve this we made use of a fuzzy inference system. Our clustering approach is used to generate the minimum possible number of fuzzy rules from the available dataset. This model facilitates the generation of an automatic data-driven fuzzy model without requiring any prior information about the structure of the dataset. A further augmentation of the HMM-Fuzzy approach is to incorporate a multi-objective evolutionary algorithm (HMM-Fuzzy with EA) to obtain a fuzzy model with the minimum number of fuzzy rules to improve on forecast accuracy. We compare the performance of the HMM-Fuzzy hybrid model with other data-driven fuzzy models using the classic time series, e.g. Mackey-Glass time series, Box-Jenkins gas-furnace time series and six ASX stock prices. Empirical results of the HMM-Fuzzy and HMM-Fuzzy with EA hybrids clearly demonstrate the efficiency of the reduced fuzzy rule methods.

The work reported in this thesis draws inspiration from statistical and econometrics approaches to forecasting. Our goal is to combine several streams of study to build a consistent, robust and useful hybrid model. Throughout our approach, we have looked at interrelationships connecting subsequent data features in the data, with HMM sorting the data based on similarities and/or relationships between predictors in the data item, and taken care to generate the minimum number of fuzzy rules. Our primary aim is to increase the forecast accuracy of the hybrid model. In doing this we made contributions to the application of the single HMM to a new approach, and the hybridization of the HMM with other soft computing paradigms.

This is to certify that

(i) the thesis comprises only my original work,

(ii) due acknowledgement has been made in the text to all other material used,

(iii) the thesis is less than 100,000 words in length, exclusive of table, maps, bibliographies, appendices and footnotes.

Signature_____

Date_____

Publications

During the course of this project, a number of public presentations have been made which are based on the work presented in this thesis. They are listed here for reference.

Peer reviewed publications related to this thesis

1. **Md. Rafiul Hassan**, Baikunth Nath and Michael Kirley, "Fuzzy modeling using HMM and EA for time series prediction", in Journal: *IEEE Transactions on Fuzzy Systems*. (Submitted)

2. **Md. Rafiul Hassan**, Yos Morsi and Rezaul Begg, "Breast cancer identification model using HMM-Fuzzy approach", in Journal: *Artificial Intelligence in Medicine*. (Submitted)

3. **Md. Rafiul Hassan**, Baikunth Nath and Michael Kirley, "A fusion model of HMM, ANN and GA for Stock Market Forecasting", in Journal: *Expert Systems with Applications*. Vol. 33, 2007, pages. 171-180.

4. **Md. Rafiul Hassan**, Rezaul Begg, Yos Morsi and Kate Lynch, "HMM-Fuzzy model for breast cancer diagnosis", *Proceedings of 15th International Conference on Mechanics in Medicine and Biology*, December 2006.

5. **Md. Rafiul Hassan**, Baikunth Nath and Michael Kirley, "A data clustering algorithm based on single hidden Markov model", *Proceedings of the International Multiconference on Computer Science and Information Technology*, 2006, pages. 57-66.

6. **Md. Rafiul Hassan**, Rezaul Karim Begg, Simon Taylor and Dinesh K Kumar, "HMM-Fuzzy model for recognition of gait changes due to trip related falls", *Proceedings of the 28th IEEE International Conference on EMBS*, 2006, pages. 1216-1219.

7. **Md. Rafiul Hassan**, Baikunth Nath and Michael Kirley, "A HMM based fuzzy rule extraction for time series prediction", *Proceedings of the World Congress on Computational Intelligence (FUZZ-IEEE)*, 2006, pages. 9966-9974. ***Received "Best session paper presentation award"***

8. **Md. Rafiul Hassan** and Baikunth Nath, "Stock market forecasting using hidden Markov model: A new approach", *Proceedings of the International Conference on Intelligent System Design and Application*, 2005,pages. 192-196.

9. **Md. Rafiul Hassan** , Baikunth Nath and Rezaul Begg, "Using HMM for forecasting stock market index", *Proceedings of the International Conference on Computer and Information Technology*, 2005, pages. 569-573.

10. **Md. Rafiul Hassan** , Rezaul Begg and Simon Taylor, "Fuzzy logic based classifier for recognizing gait changes due to trip related falls", *Proceedings of the 27th IEEE International Conference on EMBS*, 2005, pages. 4970-4973.

11. Rezaul Begg and **Md. Rafiul Hassan**, "Artificial Neural Networks in smart homes", Chapter-9 in *Designing Smart Homes: Role of Artificial Intelligence*, Edited by Juan Augusta and N. Nugent, Springer-Verlag, 2006, pages. 146-165.

12. Rezaul Begg , **Md. Rafiul Hassan** , Simon Taylor and Marimuthu Palaniswami, "Artificial Neural Network models in the diagnosis of balance impairments", *Proceedings of the International Conference on Intelligent Sensors and Information Processing*, 2005, pages. 518-522.

Peer reviewed other publications

1. **Md. Rafiul Hassan**, M Maruf Hossain, Rezaul Karim Begg and Ramamohanarao Kotagiri, "A fusion analysis of gene expression data to prognose early breast cancer", in Journal: *BMC Bioinformatics*. (Submitted)

2. **Md. Rafiul Hassan**, Rezaul Karim Begg, Ahsan Khandaker and Robert Stokes, "Automated recognition of human movement states using body acceleration signals", *Proceedings of the 2nd Workshop on Biosignal Processing and Classification*, 2006, pages. 135-143.

3. **Md. Rafiul Hassan** and Baikunth Nath , "Data compression using Huffman coding - a novel approach", *Proceedings of the International Conference on Applied Computing- IADIS*,2005, pages. 143-148.

4. **Md. Rafiul Hassan** , Baikunth Nath and Mohammed Alauddin Bhuiyan , "Bengali phoneme recognition: a new approach", *Proceedings of the International Conference on Computer and Information Technology*, 2003, pages. 365-369.

Acknowledgments

I wish to thank my supervisor Assoc. Prof. Baikunth Nath for his guidance and unconditional support during my PhD studies at the University of Melbourne. I remain grateful to Prof. Nath for teaching me Statistics and about stock markets. Dr. Michael Kirley, my associate supervisor, helped me a lot by providing critical feedback on the thesis, which took many iterations. It would have not been possible to complete my thesis without their undivided guidance and support. I also must record my gratitude to Prof. Kotagiri Ramamohanarao, head of department and member of my PhD progress committee, for his insightful suggestions.

I am gratefully indebted to the University of Melbourne; Department of Education, Science and Technlogy of the Australian Government for supporting me through their scholarship programs for international students. I privileged to study in this wonderful University. I shall long cherish my experiences in this beautiful country, Australia. I also wish to acknowledge, Shah Jalal University of Science and Technology, Bangladesh, for granting me study leave to complete this degree.

Both scientific and general discussions with other fellow researchers in the department have been very rewarding. I also thank the department administrative staff Pinoo Bharucha, Julien Reid and Binh Phan for their unconditional help; our technical staff, especially Thomas Weichert and David Staples for their excellent support.

I owe my profoundest gratitude to my undergraduate teacher Dr. Muhammad Zafar Iqbal. My life changed under his tutelage.

Finally, I thank my wonderful parents for their never-ending love and support in every aspect of my life; and specially my wife Anny for her constant encouragement and unconditional support at daily life at Melbourne.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The research presented in this thesis analyzes and develops a hybrid computational method to model time series data. The research in this thesis is based on the Hidden Markov Model (HMM) and many other ideas for time series forecasting adopted from statistical, econometrics and computational intelligence (CI) disciplines. All these ideas are put together in combining the HMM with CI paradigms to hybrid models for non-linear time series analysis. An HMM is a statistical process used to model sequential processes in which a future event depends on the current event. Examples that represent sequential process are the sequence of letters in alphabet and time series data where the collection of observations on variables created sequentially in time [1].

Being ubiquitous for modeling sequential data, HMM has proved a consistent success in modern speech recognition systems [2] and computational molecular biology [3]. Typically, in time series problem, the HMM has been used to recognize the data patterns (i.e., speech recognition) and probabilistically forecast the changes in the trend. What distinguish this research from the previous work is rather than building models to forecast the basic movement/shift in the time series data, we apply the HMM hybridized with other soft computing (SC) paradigms to forecast the exact value for the peak and plateau of each and every movement in a time series. SC is a CI method where the decision boundary is presumed soft as opposed to crisp. The hybridization of the HMM with SC aims to overcome limitations that they pose individually and matching them to attain improved performance with application for forecasting time series data.

This chapter describes the basic concepts of HMM in brief and its typical application in time series data, the difficulties of predicting the exact value of a time series, the limitations of existing techniques for time series forecasting, motivation and objective of

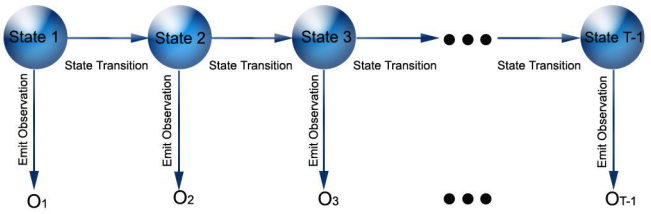


Figure 1.1: Hidden Markov Model

our hybrid HMM model. The organization of the thesis is also outlined in the later part of this chapter.

1.1 Hidden Markov Model

An HMM represents the probability distributions over sequences of observations. Let us denote the observation at time t by the variable O_t. This can be a symbol from a discrete alphabet, a real-valued variable, an integer, or any other object; it does not matter as long as we can define a probability distribution over it. We assume that the observations are sampled at discrete, equally-spaced time intervals, so t can be an integer-valued time index the range of which is from 1 to T (T = the total duration the observation sequence).

The HMM comprises a finite number of states to represent the observable sequences. Figure 1.1 shows a typical picture of the HMM. The HMM gets its name from two characteristics assumptions. First, it assumes that the observation at time t was generated by some process whose state *State t* is hidden from the observer. Secondly, it assumes that the state of this hidden process satisfies the *Markov property*, that is, given the values of *State t-1*, the current state *State t* is independent of all the states prior to $t-1$. In other words, the state at some time encapsulates, all we need to know about the history of the process in order to predict the future of the process.

1.2 Time Series

A time series is a sequence of values that a randomly varying attribute accumulates over time. A time series does not use any mechanism to adapt its values, and this makes it very different from other series. Common time series examples are stock markets, weekly weather reports, annual precipitation or weekly pizza sales. Real world time series data

tend to be continuous, and are usually a sequence of observations or values separated by equal time intervals. The corresponding probability model of a time series is called a stochastic process that represents a family of random variables with real values O_t, where $t \in \aleph$. Observations in a time series are not purely deterministic since it is a stochastic process [4].

A known characteristic of stochastic time series is that they are non-linear and non-stationary; in other words the distribution of the set $O_{t_1}, O_{t_2}, \ldots, O_{t_n}$ does not match with the distribution of the set $O_{t_1} + h, O_{t_2} + h, \ldots, O_{t_n} + h$ for every $n; t_1, t_2, \ldots, t_n$ and h. The table in Fig. 1.2 is a time series data that depicts the daily stock price of the Commonwealth Bank, Australia, from 16th February 2006 to 9th March 2006. Figure 1.2 plots the time series data presented in the table. Time series are often presumed to consist of components that enable us to predict future patterns. These components are [1]:

- Trend,
- Cycle,
- Seasonal variations, and
- Irregular fluctuations.

A trend is a variation in the movement of the time series that happens over a period of time. This component shows us patterns of long term growth or otherwise in a time series. For instance, a statement like *"The Connex authority has increasingly declined the quality of services over the last three years"* refers to a trend that occurred over the last five years.

A cycle is a recurrent oscillatory pattern associated with trend levels. Oscillations can be quantified as per the duration of similar movements that frequently occur over the length of the trend.

Seasonal variations are periodic patterns found in a time series. Tourist inflow into Australia, for instance, shows seasonal variations.

Random movements other than components defined above are called *random/irregular fluctuations* in a time series.

1.2.1 Time series prediction

Time series prediction/forecasting is the process of studying known past events and extrapolating the results to predict future events, or in other words, the process of predicting

Date	Closing Price
16-Feb-2006	44.13
17-Feb-2006	43.99
20-Feb-2006	43.54
21-Feb-2006	43.83
22-Feb-2006	43.82
23-Feb-2006	43.89
24-Feb-2006	44.16
27-Feb-2006	44.50
28-Feb-2006	44.85
01-Mar-2006	43.71
02-Mar-2006	44.27
03-Mar-2006	44.00
06-Mar-2006	43.70
08-Mar-2006	43.70
09-Mar-2006	43.65

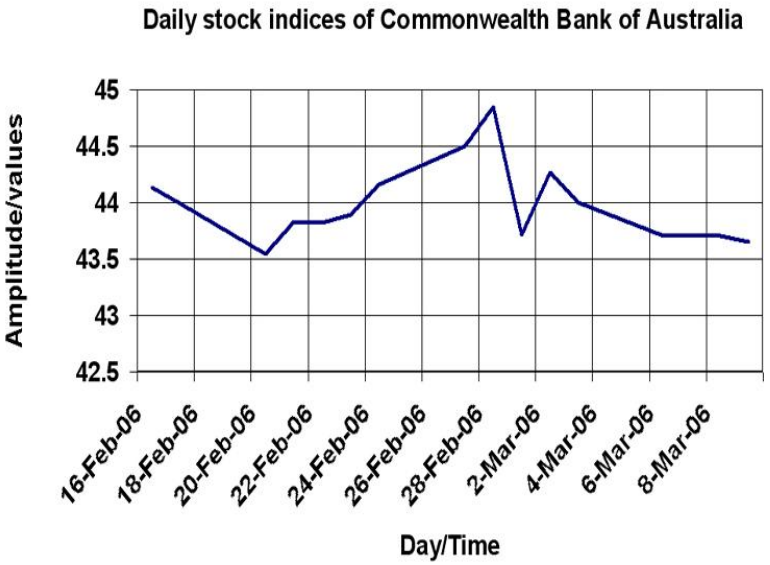


Figure 1.2: Time series data: Daily stock price of Commonwealth Bank of Australia.

future data points before they actually exist to confirm the measurements. Prediction of future values is complex and often difficult owing to the inherently volatile and non-linear nature of time series. Usually, prediction methods predict time series one or more steps ahead by evaluating historic data values alongside related data that may have influenced the series itself. In this thesis, we use the term prediction and forecast interchangeably to mean the forecast the future value.

1.2.2 Time series prediction processes

Though, predicting time series data has been broadly explored in the research fields of prediction and time series analysis, still improvement in the prediction accuracy is a challenge due to the unknown, complex and non-dominated mechanism that drives the time series behaviour.

The field of forecasting/predicting started in the 1950 and now is reaching maturity [5]. A large number and variety of forecasting/prediction methods have been developed since then. They range from the most naïve (e.g. Random walk theory) to highly complex approaches (e.g. Artificial Neural Networks (ANNs)) [6]. As such, a wide range of forecasting/prediction methods have been proposed and developed which we categorize into two main types based on the type of approaches they adopt: statistical methods and computational intelligence (CI) method.

Statistical Processes

In statistics the methods ranging from Random Walk [7] to Generalized Autoregressive Conditional Heteroscedasticity (GARCH) [8] have been developed to predict time series data. The 'Random Walk' theory suggests that the most accurate prediction for a future value is the most recently made one. However, the application of this theory is hurdled by what is known as Brownian motion [9] and such a naïve approach sometimes does not fit into complex time series analysis and prediction. Compared to the Random Walk the other naïve theory moving average [10] considers the effect of subsequent past datasets and hence takes into account the turning points in the time series data which usually tend to lag behind turning points in the time series. Regression [11] is the other forecast method which typically considers the linear relationship among the predictor variables and the predicted variables. Autoregressive Moving Average (ARMA) [12] was the improved time series analysis and prediction model, that combined both the regression and moving average to handle both the underlying trends and relationship among the variables within the same model.

However, all these methods at early stage of time series prediction were hindered with the underlying non-linearities and the Autoregressive Integrated Moving Average (ARIMA) [13] was developed to handle the non-linearities in the time series data along with the other properties (e.g., trend) of the time series. ARIMA, whcih has been proven not robust enough to handle level shifts, is recommended for use with stable time series data. Another known problem is that it requires seasonal adjustments to compensate for shifts. Autoregressive Conditional Heteroscedasticity (ARCH) [14] and the generalized version of ARCH: GARCH, on the other hand, are methods that evaluate past variances to explain future variances. The crux of the matter is that GARCH is a time series techniques that enables users to model the serial dependence of volatility. However, this model does not offer a complete solution. In spite of being honed to model time varying conditional variances, GARCH models are oblivious to irregular phenomena, for instance extreme market fluctuations like crashes and rebounds, and random unexpected events that can require in depth changes to the structure.

Soft Computing Processes

Traditionally, time series theory in statistics dealt with the randomness of data [15], which in turn requires domain experts to handle the tools effectively. However, soft computing (SC) paradigms can analyze and predict time series as they can determine an underlying complex relationship that governs apparently random fluctuations in time series data and consequently plot forecasts that are more accurate that those produced using other approaches. SC is a CI method where the decision boundary is presumed soft as opposed to crisp. Paradigms that are considered to comprise the genre of SC methods include:

1. ANN [16, 17],
2. Fuzzy Logic (FL) [18], and
3. Evolutionary Algorithm (EA) [19].

Of the existing SC methods, ANNs are effective in realizing an input-output mapping even when the exact relationship between the input and output is not known or hard to detect mathematically. This characteristic makes it particularly useful as a time series prediction technique. One representative example of application of ANN in time series forecasting is the ANN by Balkin and Ord [20]. They found that an ANN, given an intelligent choice of inputs, competes well with other linear models while forecasting the 36 time series data from M3 competition dataset [21]. Their investigation concluded that an ANN could provide superior forecasts when the process is non-linear. The performance of ANN is constrained in that it needs a long time series to be able to forecast more efficiently and accurately. However, the results of the automated ANN (i.e. data dependent ANN) compare well with other methods, but might produce poor results if used under certain conditions. Furthermore, 'Black Box' models like ANNs do not offer insight into the nature of the dataset when applied in time series forecasting and poses difficulties in fine tuning a model. Also, choosing the best ANN architecture for a specific problem is itself considered an NP-problem [22].

Compared to the ANN, FL offers a clear understanding insight into the model. FL is especially popular to deal with non-linear time series data and has been proved its robustness by solving a number of non-linear problems e.g., flood forecasting [23], stock index forecasting [24] and temperature prediction [25]. In a recent study, Cheng *et al.* [26] used FL to forecast the trends of the stock index. Nevertheless, the performance of the

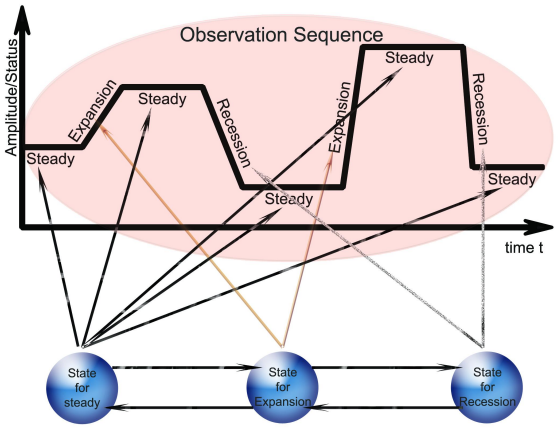


Figure 1.3: A Hidden Markov Model for forecasting trend of a time series

method depends on the fuzzification of the time series data. The generation of appropriate fuzzy rules for a time series data is often challenging. For instance, a dataset with n dimension will produce a maximum of n^n rules.

The Hybrid CI approach aims to overcome the limitations of two paradigms by mixing and matching them to obtain improved performance while applying to the time series data. One combination that complies well is the Adaptive Neuro-Fuzzy Inference System (ANFIS) [27]. However, the complexity of this hybrid system exaggerates drastically with the increase of fuzzy rules. Such inherent complexity may restrict the user from actually solving a non-linear time series problem.

Hidden Markov Models for forecasting

Time series prediction and forecasting processes include HMM applications more often than not, and in many applications, states are presumed to be optimal for predicting a certain situation, e.g., sudden movement in change i.e., up or down movement within the time series. For example, a time series formed by economic status of a business is comprised of three states: steady, recession and expansion. In a traditional approach, the HMM for this time series would be as shown in Fig. 1.3. It should be noted that the observation sequence for the HMM is the time series itself and each of the states of the HMM represents to be expert for forecasting a specific status of the time series. As an example, Andersson *et al.* [28] trained an HMM that is able to call an alarm before the turn while monitoring business cycles. However, such an application of HMM in time

series is unable to predict the exact future value of the time series data.

In another related study, Weigen and Shi [29] predicted the daily probability distributions of S&P 500 stock index returns using the HMM. These authors were not deterministic in choosing the appropriate number of states and considered several discrete states as prediction experts. This means that exact future values may not be accurate. In a typical application of HMM for time series prediction, it does not consider the effects of other events in the change of the considered time series. For instance, in Fig. 1.3 the HMM has been modeled for a time series in business cycle, which does not consider the effects of other events on the change of the trend in the sequence. It is a daunting task to combine the effects of predictors other than the past value of the predictee itself while using the HMM for forecasting a time series data.

The ongoing challenge is to choose the optimal initial values for the HMM parameters. The HMM is characterized by three parameters:

- **Observation emission probability:** the probability distribution of a specific observation/status/event emitting from the states in the HMM,
- **State transition probability:** the probability to switch the states in the HMM, and
- **Prior probability:** the probability of choosing a specific state at starting of the system to be modeled.

For a given phenomenon, classical HMMs make strong presumptions on statistical properties. For example, first order Markov chains are used to model stochastic processes involved; and this effectively restricted by the parametric form of the probability density functions that represent the emission probabilities associated with the states [30]. When combined, ANNs and HMMs could eliminate the individual limitations of each technique (for example, see [31]). Typically, such a hybrid system presumes that the ANN's output is sent straight to an HMM for pattern recognition (as in speech recognition) [32, 33]. Architectures proposed by Bourlard *et al.* [31] tend to depend on a probabilistic interpretation of ANN outputs. The output of the ANN is actually trained to be used as a non-parametric estimate of the *posterior* probability of an HMM state for the given observation sequences and the information of the previous state. This represents one of the hybrid models that have influenced the following approaches. Such a hybrid, however, depends on the availability of an analytically motivated global optimization scheme defined

at the whole system level, and also is constrained by the requirements of huge ANNs to be trained on the problem solving techniques [30].

1.3 Aim of this research

The aim of this research is to develop a coherent and powerful hybrid HMM and soft computing method for modeling time series data. The motivation is to overcome the limitations of HMM, by fusing other forecasting approaches from soft computing, statistics and econometrics disciplines. Here, the HMM is applied in a novel manner to handle multiple variables/predictors in a multivariate time-series data to obtain a better forecast.

This work extends how HMM can be applied to identifying similar data patterns, formed by the predictors, in sequential time series data, and then combined with other approaches to obtain accurate forecasts. Ergo, it may be said that the goal of this research is:

To develop a simple and easy to understand HMM-based hybrid soft-computing method which is intended to be a powerful model for prediction of time series with improved accuracy and be computationally inexpensive.

1.4 Contribution to the thesis

In order to achieve primary research goals, the following approaches have been developed:

Firstly, HMM is applied to determine the degree of similarities amongst data patterns within the dataset. This determination is based on the produced HMM's scoring for each of the data patterns. This approach aims:

- to reduce the computation by using single HMM,
- to group or cluster similar data patterns in an unsupervised manner, and
- to identify the possible number of clusters from the available dataset.

Secondly, HMM-based similar data pattern identification and sorting is applied to mark similar financial behaviour in historical datasets as is being currently observed, and then the difference between the matches and the next-to-the-matched data pattern in the

sequential time series data is used to generate a one day-ahead forecast for the financial time series data. This approach aims:

- to forecast future value of the time series data given historical dataset, which solves the problem of prediction of the exact value for the time series,
- to use only a single HMM for forecasting time series data which is termed HiMMI, and
- to improve the developed HMM-based forecast model (HiMMI) by fusing ANN and GA, which replies the challenge of choosing the optimal initial values for the HMM parameters.

The third part of this approach introduces an improvement to the HiMMI by incorporating FL as a means of forecasting. To accommodate this change at this stage, the HMM-based sorting and partitioning approach is modified, so that an automated fuzzy rule generator that is based on the HMM-data partitioning approach is produced. This enables the generation of an automatic data driven fuzzy model that does not require any prior information about the dataset's structure. The approach is fine-tuned by incorporating a Multi-Objective EA to obtain a fuzzy model with the minimum required fuzzy rules and for improving performance on time series data prediction.

This work draws inspiration from different branches of SC. We have aimed to bring together ideas from these various branches and combine them into a high performance SC method. Certainly, we do have specific objectives that lead to the creation of such a method, but on a broad basis one might say better interpretation and improved prediction accuracy describes the end to which these objectives are but a means. In our quest, we considered interrelationships between subsequent features in data patterns using HMM, we sorted data patterns based on similarities and/or relationships amongst features within data patterns, and minimized the number of possible fuzzy rules to hone prediction capabilities. Before this, no one has applied a single HMM to such new approach as we prescribed and hybridized HMM-based data sorting/partitioning method with existing computational techniques. The focus has been on utilizing HMM with a different perspective and combining it with other methods. Contributions that stand out are the methods used to combine outputs from the HMM and the application of Bayesian estimation to parameters in the expectation maximization method while the HMM is being trained. In

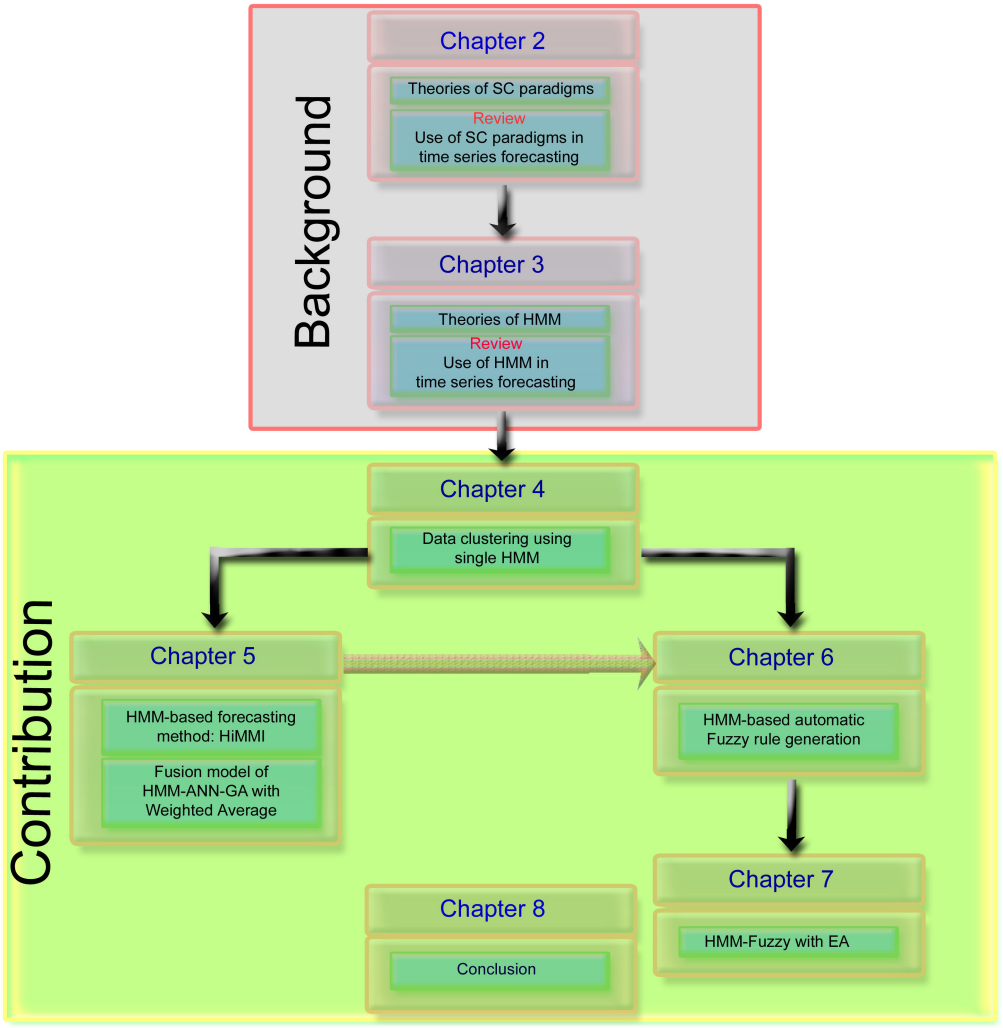


Figure 1.4: Organization of the thesis

their own ways, honing the HMM by embedding an EA in it, the fuzzy rule generation approach and the new forecasting approach are unique contributions.

1.5 Organization of the thesis

Figure 1.4 shows the organization of the thesis, which presents that the development of the hybrid model occurs in three aspects. Each of these three aspects is presented as a chapter in its own right. The background has been provided in Chapter 2 and Chapter 3. The contributions to the thesis are presented from Chapter 4 to Chapter 7. The concept of using the HMM for clustering, described in Chapter 4, has been extended to develop the

two forecasting models in Chapter 5. Furthermore, the clustering concept has also been used in generating fuzzy rules to forecast time series data which are presented in Chapter 6 and 7. Details about each of the chapters are as follows:

Prior to detailing major contributions to the field, details of existing SC techniques that predict time series data are put forth in Chapter 2. Owing to our developmental objective to develop a SC model, background information along with details of techniques and their application have been provided. HMM application is a highlight of our ongoing research. Ergo, HMM has been the focus elsewhere, and hence Chapter 2 includes only a review of application the SC model to time series prediction. Chapter 3 introduces the HMM in details. Insight into HMM and the difference between discrete HMM and continuous HMM is primary to understanding to the contribution of this thesis. Thus, details have been provided in that chapter as a prelude to detailing the contributions. That chapter also presents a review of application of HMM to time series forecasting.

Chapter 4 illustrates the extension of an HMM application to identifying similar data patterns and group them into a number of clusters. Details about the new approach of clustering data patterns in an unsupervised manner have been described.

Chapter 5 extends the HMM-based clustering method to generate forecasts on financial time series data. An HMM is used to mark similar data patterns from historical datasets and current datasets, and two alternate techniques are developed that construct a 'one day-ahead' forecast based on the HMM-identified patterns. Our first approach interpolates neighbouring values of identified data patterns to compute forecast values. The second one locates similar data patterns from historical data and uses a weighted average of the differences between neighbouring data items. This value is then added to the current value of the variable of interest to produce an appropriate forecast value.

Fuzzy Logic is introduced in Chapter 6 as a means of generating forecasts for time series data. Two types of Fuzzy Models, the Mamdani model, and the Takagi-Sugeno model, are introduced in that chapter. The challenge of extracting an optimum number of fuzzy rules from numerical data is described and details of traditional techniques are explored. The latter part of the chapter focuses on the new approach of HMM-based fuzzy rule generation (HMM-Fuzzy).

Chapter 7 introduces an extension to the HMM-Fuzzy approach that can further improve performance. A multi-objective EA is used to find a range of trade-off solutions between the number of fuzzy rules and accuracy. Before EA is used, a logical and mathe-

matical presentation of the HMM-Fuzzy approach is made.

Chapter 8 concludes the thesis and includes a quick review of the developed methods. Extensions to the HMM approach and hybridization with SC techniques are evaluated against benchmark non-linear time series data and financial time series data. Prediction results are compared to those of existing methods on the same datasets.

Part I

Background and Literature Review

Chapter 2

Soft Computing with applications to time series forecasting

This chapter provides, through selective reference to some of the literature, a clearer understanding of different SC, models and statistical techniques that are widely used in time series forecasting. Experimental observations of various time series forecasting phenomena are reviewed, and some of the conflicting conclusions about the existing techniques reviewed in the respective literatures are discussed.

This thesis involves the design and analysis of a hybrid computational method which combines HMM with the SC paradigms namely ANN, FL and EA to model time series data. Therefore, a brief theory on these SC paradigms is presented at beginning of this chapter. Since, HMM is fundamental of this thesis, the detail theory along with it's application in time series data is described in Chapter 3. In this thesis, we compare our results with that of a popular statistical technique: ARIMA model and hence a brief theory of this statistical technique is provided in this chapter following the theories of SC paradigms.

Time series is the application area in this thesis and therefore the extensive review of the literature is performed focusing on many different individual approaches of the SC paradigms applied in time series analysis. For each of the techniques, we review some of the key papers that have been highly influential to the application area and report the findings. Most of the papers attempted to solve some of the reported limitations of the respective techniques while applying to time series analysis. Therefore, we report the limitations of each of the techniques after describing the findings of the papers reviewed rather than sticking them immediately after the presentation of the findings.

The remainder of the chapter is organized as follows. Section 2.1 briefly reviews the theories of SC paradigms. Section 2.2 outlines the ARIMA model. In Section 2.3, we review the findings of studies that used SC techniques for forecasting time series data. The analysis of the usage of SC in time series forecasting is provided in different subsection after the review of each of the SC paradigms. Finally, a summary is presented in Section 2.4

2.1 Preliminaries of Soft Computing Paradigms

2.1.1 Artificial Neural Network

Artificial Neural Networks (ANN) or alternatively known as Neural Networks (NN), are computational techniques that are based on biological brain activity as illustrated in Fig. 2.1. Processing networks are made of interconnected neurons [34] that resemble biological neurons, originally described by Cajal [35]. ANNs 'learn' relationships between inputs and outputs by 'studying' examples. The ANN does not have prior knowledge of the underlying relationships within datasets. It is this adaptive attribute that makes ANNs feasible for time series analysis and prediction tasks [36]. Some basic features of ANNs are briefly described below:

Neuron

Neurons are the building blocks of an ANN. Each neuron accepts many inputs and produces an output that is the result of some processing. Neurons are interconnected by links that are weighted, where the weight represents the strength of the connection. Figure 2.1 gives analogy of a computational neuron to a biological neuron. An artificial neuron's inputs are a man-made version of a biological neuron's dendrites. Synapse 'activities' are represented by weights on the edges, soma are represented as mathematical equation shown in the circle, and the axon is the calculated output of each neuron.

Network Architecture

Neurons form interconnected networks, and thus, the way they are connected is best visualized using network architecture. Depending on the position of neurons in an ANN, there exist three layers: the input layer, the hidden layer, and the output layer. The number of hidden layers may range from zero to as many as the complexity of the problem requires. Depending on the problem being solved, researchers have presented various

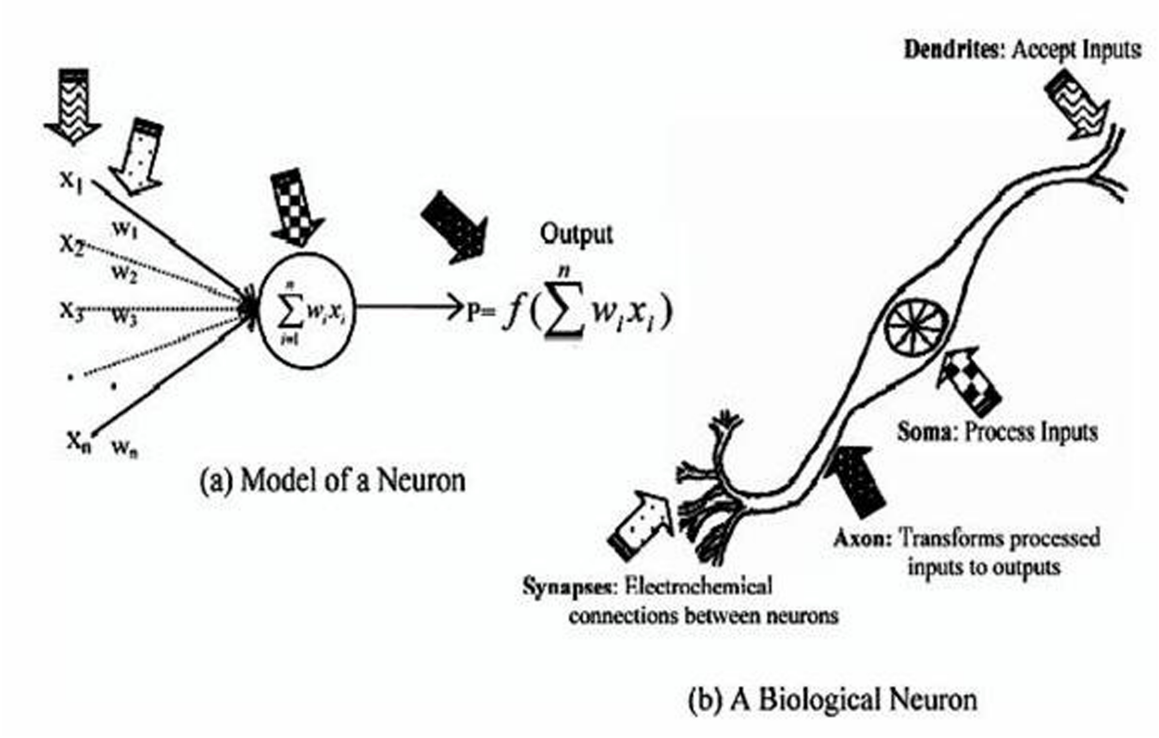


Figure 2.1: The model of a neuron

network architectures. Architectures that are most commonly used are: feed-forward fully connected NN, recurrent NN, and time-delay NN.

Learning Process

ANNs are unique amongst mathematical processing methods in that they can 'learn' from data characteristics, and adapt network parameters according to underlying structures in the training dataset. The process is called '*learning*'. A good definition of the process was put together by Haykin [34], "*Learning is a process by which the free parameters of a NN are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which parameter changes take place*".

The learning environment can dictate which of the two primary learning paradigms is applied: supervised learning [17], and unsupervised/self-organized learning [34]. In a supervised learning paradigm, the ANN is given some target outputs that it aims to compute correctly during the training process, or in other words, it is supervised. Unsupervised learning, as the name implies, is a training process where the ANN learns on its own from

the data distribution and structural similarities within the dataset.

Learning Algorithms

The commonly used back-propagation algorithm fits most ANN architectures. This algorithm modifies weights by calculating the error via a suitable error function. In case the error function is a differentiable function, then performing gradient descent on such a function would produce a learning rule. Several researchers have put forward this idea and argued for the above technique [e.g. [37]] in the past. Practical techniques for this powerful engineering tool were developed by Rumelhart *et al.* [38]. Back-propagation techniques follow the delta rule [17] when modifying network weights.
Applications of ANN to time series forecasting are discussed in Section 2.3.1.

2.1.2 Fuzzy Logic

Fuzzy logic (FL) typically processes non linear datasets by mapping input data (feature) vectors into scalar output, i.e., it maps numbers into numbers. The non linear mapping is established with the help of *fuzzy set Theory and fuzzy logic* [39]. FL is a superset of conventional (read Boolean) logic, which allows a gradual and continuous transition as opposed to a sudden change in (usually binary) values of the features [39].

According to Lotfi A. Zadeh, the father of the Fuzzy Set Theory and Fuzzy Logic [18], "*fuzzy logic is nearly synonymous with fuzzy set theory. Fuzzy set theory, as its name suggests, is a theory of classes with un-sharp (or fuzzy) boundaries. It is much broader than fuzzy logic in its narrow sense and includes the latter as one of its branches. The importance with the fuzzy system is that any crisp theory can be fuzzified by generalizing the concept of a set within that theory to the concept of a fuzzy set. The impetus for the transition from a crisp theory to a fuzzy one originates from the fact that both the generality of a theory and its applicability to real-world problems are substantially enhanced by replacing the concept of a set with that of a fuzzy set.*"

To understand the primary concept of fuzzy systems, referred by Munakata and Jani [40], readers are directed to literatures: [41–45]. In classical (non-fuzzy) set theory, an element U either belongs to or does not belong to the universal set S. This mapping of the elements of S to the elements of the set {0,1} is as follows [46]:

$$U : S \rightarrow \{0, 1\} \tag{2.1}$$

In the equation 2.1, considering ordered sequences of elements in S and U respectively makes the mapping a set of ordered pairs as is considered in ordinary classical set theory. In the ordered pair, the first element belongs to S while the second element is an element of {0,1}. '0' representing the **non-membership/false/no status** while the value '1' represents the **membership/true/yes**. Thus, this mapping used in classical set theory may be represented linguistically as follows:

'x' is in U , where $x \in S$,

Like the classical set, in a fuzzy set, a fuzzy subset F of the set S is defined as a set of ordered pairs where each of the ordered elements from S is mapped to the corresponding ordered element from the interval [0, 1]. A fuzzy set is a generalized representation of the classical set which introduces a degree of membership for each element U to indicate the involvement of the element in the set S. The membership degree is the real number in the range of between 0 and 1 [40]. The value '0' is used to represent absolute non-membership, the value '1' is used to represent complete membership and values in between are used to represent intermediate degrees of membership [46].

To illustrate the idea of a fuzzy set, let us consider a human attribute – *Age*. In this illustration, the set S (the universal set of discourse) would represent the set of all *people* with varying *ages*. For the sake of exemplification, let us define one fuzzy subset as '**YOUNG**'. Following Zadeh, '**YOUNG**' is a linguistic variable, which would represent a cognitive category of '*Age*'. Every element *(person)* in the set S, one must assign a degree of membership to the **YOUNG** fuzzy subset. One accepted way to do this is to use a membership function based on the *person's age* (or the element's attribute). While in classical set theory, the element may be either '**YOUNG**' or '**Not YOUNG**', in a fuzzy system, the membership function will represent the *degree of membership*, or *how 'Young'* the element is. This degree would be in the range of 0 – 1. If the degree of membership function is 0.25, then this would mean that the person belongs to the set S: the universe of discourse, with a degree/energy of 0.25. Equation 2.2 represents one such membership function to impose degree of membership function for a *person* – 'x', when the *age* is given [46].

$$\begin{aligned} young(x) &= 0 && \text{if age}(x) > 60yr, \\ &= (60yr - age(x))/2yr, && \text{if } 35yr \leq age(x) \leq 60yr, \\ &= 1, && \text{if age}(x) < 35yr \end{aligned} \tag{2.2}$$

Using the above membership function, one may interpret the meaning of '**YOUNG**' in FL, as opposed to its meaning in linguistic fields. When more than one attribute is interpreted in order to identify a unique *person/element*, the statement would look more like: The person is **YOUNG** and **TALL**.

It may be noted that the statement uses two fuzzy-linguistic terms: **YOUNG** and **TALL**. In the real world, such terms occur randomly. The fuzzy system takes into account such linguistic representations of an object's attributes and generalizes them using the membership function.

Generating the Fuzzy Model

The following steps are used to generate a fuzzy model :

1. Fuzzification,
2. Generate fuzzy rules,
3. Generate the fuzzy inference,
4. Defuzzify the fuzzy output.

Fuzzificaton

Input and output variables usually hold numeric values or linguistic (categorical) values. Variables that are numeric are translated, depending on their range, into fuzzy linguistic terms. For example, the age of a person 45 years could be considered either '**YOUNG**' or '**OLD**'. As illustrated in Fig. 2.2, a membership function is applied to translate these values into fuzzy terms.

Generating fuzzy rules

FL handles non-linearity well because of the fuzzy rules it uses to map the non-linear relationship between inputs and outputs. The following fuzzy rule relates the variables *temperature* and *humidity* with controlling a Heating, Ventilation and Air-Conditioning (HVAC) system.

If temperature is **hot** and humidity is **high** then turn the knob **fully clockwise**.

Fuzzy rule generation is usually based on past knowledge and experience. There do exist some problems for which it is hard to find such knowledge or experience; however,

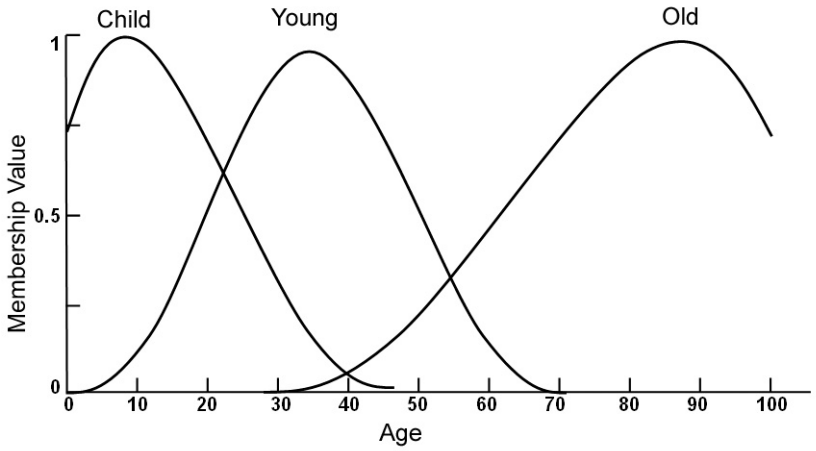


Figure 2.2: The fuzzy membership function

data pertaining to solutions might be available. Data driven fuzzy models generate fuzzy rules from available data, though success rates usually vary.

This chapter provides only the introductory theory of the techniques. The challenges to the data driven fuzzy models, along with the traditional approaches, are described at the beginning of Chapter 6 in Part III of this thesis.

Fuzzy Inference

The functional mapping of inputs and outputs for the fuzzy rules depends on how the rules are fired. When designing a fuzzy model we must ponder the situation, "*given the status of each input feature, what would be the output desired?*" Fuzzy inference is the process that produces outputs for each of the fuzzy rules. Figure 2.3 shows two fuzzy rules and produces the respective outputs for each of the fuzzy rules.

Defuzzification

The objective of FL is to produce a quantifiable result, and the process that does this is called *defuzzification*. Usually, fuzzy systems have rules that transform numeric variables into fuzzy values, and then, the result is expressed as degree of membership in fuzzy sets. A proper defuzzification method would aggregate the results of the rules in an appropriate manner. The majority of membership functions have the graph of a triangle. Were this triangle cut in a straight horizontal line between the top and the bottom, and were the top to be removed, the remainder would form a trapezoid. Defuzzification, in its first

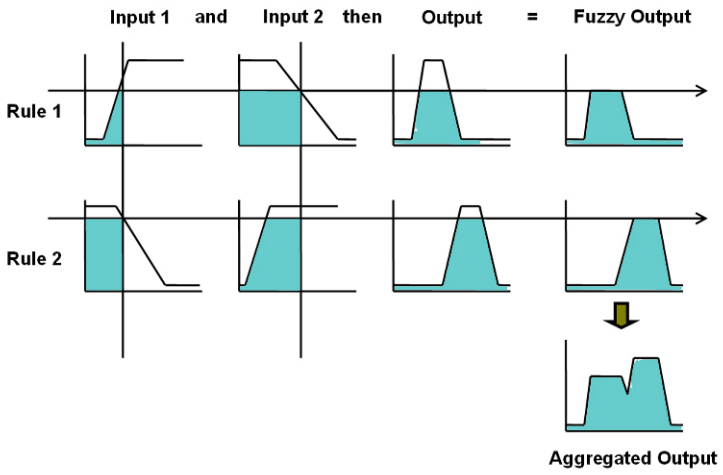


Figure 2.3: The fuzzy Inference

step, crops parts of the graphs to create trapezoids (or other shapes depending on the initial shape). The most common way of summing up all the results is to superimpose these trapezoids one on the other to form a coherent geometric shape. The centroid of this shape, known as the fuzzy centroid, is calculated, and the x-coordinate of the centroid is taken to be the defuzzified value. Figure 2.3 shows the process of aggregating outputs using the individual fuzzy outputs produced by the respective fuzzy rules.
Applications of FL to time series forecasting are discussed in Section 2.3.2.

2.1.3 Evolutionary Algorithm

Genetics and natural selection principles have paved the way for an optimization and search technique known as the Evolutionary Algorithm (EA). Given a certain population, an EA can enable the populace to evolve based on predetermined selection criteria so that members exist at maximum '*fitness*', or in other words, their state of existence minimizes the cost function. [47]

Using an EA has some known advantages, in that it:

- Optimizes with continuous or discrete variables,
- Does not require derivative information,
- Simultaneously searches from a wide sampling of the cost surface,
- Deals with a large number of variables,

- Is well suited for parallel computers,
- Optimizes variables with extremely complex cost surfaces (they can jump out of a local minimum),
- Works with numerically generated data, experimental data, or analytical functions.

A well known instance of an EA is sometimes a Genetic Algorithm (GA). If the problem is represented using a binary chromosome, i.e., 1 or 0 then the process is named as GA. In this thesis, we have used both the EA and GA to refer to the standard evolutionary search approach regardless of the types of population unless it is explicitly remarked.

The '*survival of the fittest*' adaptation EA applies differentiates it from other search techniques. A set of strings, each with their own fitness values, makes up a population. The fitness value is used to sort potential parent strings from the current population. EA uses certain operators: selection, crossover and mutation, and these operators require an initial population to work operate upon. The crossover operator produces offspring from two selected strings by crossing over at a certain point. In time, the mutation causes a variation in the population as bit values in the string change. As with natural evolution, the process is iterative, and generations evolve until a given criterion has been met.

The advantages of EA apart, in situations where traditional optimization performs appallingly EA perform very well. Like all optimization algorithms, EA defines optimization variables, the cost function and the cost. It concludes with convergence testing, just like any other optimization algorithm. It is in between that this algorithm differs.
Applications of EA to time series forecasting are discussed in Section 2.3.3.

2.2 Statistical Techniques

Forecasts on time series data have been reported using several techniques. Common techniques may be put in two domains: the statistics domain, and the SC/Machine Learning domain. The well known SC techniques have already been described in Section 2.1. We describe the statistical methods for time series forecasting in the following paragraphs.

The majority of literature on forecasting is statistical. Clemen [48] found that forecasting as we know it now began when Laplace thought of combining regression coefficient estimates. Stigler [49] argues that Laplace's estimates derive and compare the properties of two estimators:

1. least squares, and

2. a kind of order statistic.

Currently, many methodologies exist that forecast time series, amongst which the well known are Moving Average (MA) [10], Autoregression (AR) [11], Least Square Linear Regression [11], ARMA [12], ARIMA [13], ARCH [14] and GARCH [8]. The focus of this thesis is HMM (detailed theory of HMM is presented at Chapter 3) and SC techniques. However, a brief review of ARIMA is presented so that the thesis is self-contained. In the statistics domain ARIMA is popular and widely used method for time series analysis and forecasting [6].

ARIMA

The ARIMA method is generated from the ARMA method with addition of differencing the 'd' times for an ARIMA(p,d,q). An ARMA is a time series prediction method which combines the two base prediction methods AR(p) and MA(q), p = *the time lag in the time series* and q = *the moving window size in the time series over which the average is calculated*. As described in [50] the basic relation of the ARIMA is as follows:

$$A_{(q)}y_{(t)} + \sum_{i=1}^{n} B_{i(q)}U_{i(t)} = e_{(t)} \tag{2.3}$$

where,

$y(t)$=output;

$U_{i(t)}$=terms are inputs of ARIMA;

$e(t)$=noise term or error term;

n=number of inputs;

$A_{(q)}$= Autoregressive operator;

$B_{i(q)}$= Moving average operator;

The values of $A_{(q)}$ and $B_{i(q)}$ are found using the following equations:

$$A_{(q)} = 1 + a_1 q^{-1} + a_2 q^{-2} + ... + a_k q^{-k} \tag{2.4}$$

$$B_{i(q)} = b_{i0} + b_{i1} q^{-1} + b_{i2} q^{-2} + ... + b_{im} q^{-m} \tag{2.5}$$

$$(i = 1, 2, ..., n)$$

In these relations, k is order of the delay polynomial $A_{(q)}$ and m is order of the delay polynomial $B_{i(q)}$. $a_1, a_2, ..., a_k$ are coefficients of $A_{(q)}$ and $b_{i0}, b_{i1}, ..., b_{im}$ are coefficients of $B_{i(q)}$. q indicates delay operator or backshift operator. The delay operator of order j is defined as:

$$q^{-j} S_{(t)} = S_{(t-j)} \tag{2.6}$$

where, $S(t)$ is a time domain function. Thus:

$$A_{(q)} y_{(t)} = y_{(t)} + a_1 y_{(t-1)} + a_2 y_{(t-2)} + ... + a_k y_{(t-k)} \tag{2.7}$$

$$B_{i(q)} U_{i(t)} = b_{i0} U_{i(t)} + b_{i1} U_{i(t-1)} + b_{i2} U_{i(t-2)} + ... + b_{im} U_{i(t-m)} \tag{2.8}$$

To forecast the value of $y(t)$ using this ARIMA model, it is assumed that past values of this variable and all values of inputs up to time t are known.

2.3 Application of SC techniques in time series forecasting

Soft Computing (SC) techniques are often applied to forecast time series data. A brief review of SC paradigms in time series forecasting is presented below.

2.3.1 Use of Artificial Neural Networks

Neural network is probably the most frequently used non-linear technique for time series forecasting over the last two decades. What follows is a brief review at the application of ANNs to time series forecasting.

Referenced by Zhang *et al.* [51], Hu [52] applied Widrow's Adaptive Linear Network, a type of ANN, to weather forecasting, which was one of the first areas ANNs were applied to for forecasts. With time, ANNs were put to other applications where it was necessary to analyze and forecast time series data, cf. [53, 54]. In a related study, Hill *et al.* [55] found the NNs effective for discontinuous time series and as an alternative to the traditional statistical forecasting methods. In this study, the time series consisting of systematic sample of 111 series in the first M-competition [21] was used. Hill divided each series into a portion for model estimation and a holdout portion for model evaluation. These time series contained annual data, quarterly data, and monthly data. The structure of the NN was identified using the test data of two annual, quarterly, and monthly series. For the annual data, there were three input nodes, two hidden layer nodes, and one output node.

For the quarterly data, there were four input nodes, two hidden layer nodes, and one output node. For the monthly data, there were nine input nodes, four hidden layer nodes, and one output node. All NN models were estimated using the back propagation algorithm of Rumelhart and McClelland [38]. Nelson *et al.* [56] investigated the necessity of statistical deseasonalization of data to obtain an improved forecast accuracy prior to using NN to the 68 monthly time series data from the M-competition [21]. In another study of NN for time series forecasting, Farway and Chatfield [57]compared the performance of several NNs. They forecast monthly airline passenger trends by varying the input variables, the architecture, the activation functions and the criterion for the best model. Their results suggest that NN models with many parameters are better fits, but are not good for forecasting. On the other hand, when compared to the Box-Jenkins [58] and Holt-Winters methods, NN brought forth better forecasts when the best ones were combined.

ANNs for forecasting non-linear benchmark time series

The application of ANNs to the prediction of non-linear dynamic time series was under focus in several studies. Lapedes and Farber [59,60] were the first to successfully apply an ANN to forecast chaotic time series data. They used a feed forward ANN for aping and predicting dynamic nonlinear time series data. Their predictions for two chaotic time series generated by a logistics map [61] and the Mackey-Glass equation [62] illustrated that ANNs could forecast such time series with incredible accuracy. Geva [63] applied several back-propagation ANNs to predict three sets of chaotic time series. Prior to the application, he preprocessed the input dataset with wavelet decomposition at different scales. He found that in terms of prediction error (measured in normalized mean square error), single scale architectures were outperformed by corresponding multi-scale NN architectures.

Oliveira *et al.* [64] labeled NNs an important technique in their time evolution predictions for non-linear systems. This team forecast well-known time series, The Lorenz Series, the Hanon Series [65] and logistics map [61] with a two layer feed forward NN. The experiment was repeated with varying network architectures and conditions to both train the net and to make forecasts. They found that the best architecture to forecast states of chaotic temporal series was $m : 2m : m : 1$. This architecture outperformed many contemporary results reported.

In their attempt to forecast total solar radiation time series for several years, Mihalakakou *et. al.* [66] used the strengths and limitations of the NN approach to model to

forecast the time series. With regard to 'one-lag' and 'multi-lag' output radiation, the NN's performance was well documented. To predict hourly total solar radiation values, they used a multi-layer back-propagation NN. Other papers that have contributed to our understanding of ANN applications in the analysis and prediction of chaotic time series were provided by Zhang *et al.* [51] are Jones *et al.* [67], Chan and Prager [68], Rosen [54], Ginzburg and Horn [69, 70], Poli and Jones [71].

ANNs in solving real world time series problems

Other than the chaotic time series, ANNs have also been applied to real world time series e.g. electricity demand forecasting, rainfall forecasts and business trend forecasts. Electricity load forecasts were made with an ANN by Taylor and Buizza [72]. Their input dataset and common variables consisted of weather ensemble predictions that affected short term and long term demand. Forecasting from one to ten days ahead produced more accurate electricity load forecasts than those given by many traditional procedures. Baratti *et al.* [73] modeled the rainfall-runoff process for different time step with the help of NN. In that study NN is used to forecast the 1-day ahead runoff in the Sardinia basis. The performance of the multilayer perceptron network was better than the forecast produced by ARMAX model. Huang *et al.* [74] have forecasted river flow in Apalachicola river using an ANN. Daily flows and rainfall data were fed into the three-layer feed forward ANN as input. The ANN was trained using scaled conjugant gradient algorithm. For the period of 1939–2000, the ANNs predictions of rainfall were quite encouraging and better than that of ARIMA forecasting model. Zhang *et al.* [51] in their review study, referred other papers applied ANN in the area of short term forecasting. These include Bakirtzis *et al.* [75], Dash *et al.* [76] and Kiartzis *et al.* [77].

Tkacz [78] found the NNs to produce statistically lower short-term forecast errors for the year-over-year growth rate of real GDP relative to linear and univariate models. He used NNs to forecast the financial and monetary growth of Canadian output. The structure of the NN was a three layer NN with back propagation algorithm. The input layer consists of 127 neurons, where the 127 variables are obtained from the set of seven leading indicators of future output growth. The number of hidden units was chosen in between one and four. Two NNs were used for the one and four quarter-forecast horizon while another NN was used to minimize the mean squared error (MSE) for the test dataset.

ANNs in foreign exchange prediction

Artificial Neural Networks (ANNs) have been successfully used for forecasting the foreign exchange rate of different foreign currencies. Yu *et al.* [79] proposed a novel nonlinear ensembles forecasting model integrating generalized linear auto-regression (GLAR) with ANNs. The model uses ANN to handle the nonlinearities and principal component analysis (PCA) to choose the number of individual forecasting models, to be combined in the hybrid model for forecasting, and then integrates GLAR with ANN models as a unique hybrid model to obtain forecast for the time series of foreign exchange rate. The forecasting performance of the integrated nonlinear ensemble model was better than that of the individual GLAR and ANN, as well as better than other hybrid model of linear combination models. The forecast result for exchange rates between the US Dollar and three other major currencies—German Marks, British Pounds and Japanese Yen have been studied considering most of the evaluation criteria, e.g. MSE, normalized MSE (NMSE) , Mean Absolute Percentage Error (MAPE), Directional change statistics (D_{stat}) , and Annual Return Rate (R). In another study, Yao and Tan [80] recorded empirical evidence of the applicability of NNs to the prediction of foreign exchange rates. They found that NNs, simply given the time series data as input, produce results comparatively poor compared to that of with time series data and technical indicators to capture the underlying "rules" of fluctuations in currency exchange rates. The rescaled range analysis has been used to detect the long-memory effect in the time series of foreign exchange rate over a time period. This measurement is used to indicate whether the market is efficient or not. Authors in that study concluded that, NN can accurately forecast foreign exchange rate if the market is not efficient. Abraham [81] analyzed several techniques individually and hybridizing them to predict monthly average forex rates of Australian Dollar with respect to US Dollar, Singapore Dollar, New Zealand Dollar, Japanese Yen and United Kingdom Pounds. The root MSE (RMSE) values of the test results show that the performances of the hybrid CART-MARS model and the hybrid Neuro-Fuzzy model were better than that of the individual MARS model or the individual CART model.

ANNs for forecasting financial time series

Artificial Neural Networks (ANNs) have been used to forecast financial time series because they are able to find the complex relationships between financial predictor variables and

the variable to be predicted. ANNs are frequently used in predicting stock performance and selecting stocks for trading on stock markets [82]. Successful stock market forecasts with back-propagation ANNs were first reported by White [83]. White used the IBM daily common stock returns and found that the results tended to be overly optimistic. Chan *et al.* [84] found NN useful and simpler to apply while training conjugate gradient learning algorithms for financial time series forecasting. The experimental result for the daily trading data of eleven listing companies during 1994–1996 collected from Shanghai Stock Exchange, empirical evidenced the adaptability of the NN to model stock price based on historical trading data.

Leigh *et al.* [85] used both the template matching technical analysis through the use of "bull flag" stock price, volume pattern heuristics and NNs to support decisions about investment in specific stock. For a further improvement in the correlation of actual and predicted values, a GA is used to find the optimal set of input features out of 22 initial input features. These 22 input features consist of 10 for stock price, 10 for volume and 2 for window heights; and they include the difference between the lowest price and the highest price in the 60-day price window template, and the corresponding for the volume. Here, the window is composed of price/volume of 60 trading days at a time which is fitted into a 10 X 10 grid "bull flag" template matching as per the description in Duda and Hart [86]. The 10 input prices, or volumes, are collected from this template fitting process. Initially, these inputs are fed into a NN of 22 input nodes, 1 hidden layer with 6 nodes and 1 output node. The output is the prediction of the future price at a 20-trading day forecasting horizon. A GA is used to determine the subset of the 22 input features to obtain a higher correlation between the predicted values and the actual values. Kolarik *et al.* [87] applied NN for forecasting time series and found its performance better comparing with ARIMA technique. They found that increasing the input window (number of inputs) from the univariate data does not result in better forecast due to the loss of generalization power of the network. For both the IBM daily stock price dataset and airline passenger time series dataset, ANNs forecasting performance beat that of the ARIMA.

Kuo [88] proposed and implemented an intelligent stock market forecasting system able to deal with both qualitative and quantitative input features. The system consisted of ANN, fuzzy Delphi and then again another ANN to produce the final output. In there the Fuzzy-Delphi, a method where the final forecast is generated through the convergence of the intermediate forecast by adopting a feedback mechanism, was used to capture the

stock experts' knowledge which are typically non-quantitative data. Experimental result on Taiwan Stock market using the integrated ANN was found much better compared with the Single ANN. Atiya *et al.* [89] perceived fundamental company information as a predictor for forecasting stock market, while NN was used as a forecasting tool. The target was defined using a set of discretized values to help the investor decide which stock should be bought, which should be sold, which should be kept etc. Thus, total annual profit gain was compared to that of by following the forecast of the NN tool. The test result indicated that the NN could guide the investor in investing money or selling their stock for the Standards & Poor (S&P) 500 stocks for the period of 1993– 1994. Kimoto *et al.* [90] have reported on the effectiveness of alternative learning algorithms and prediction methods using ANN, when developing a Tokyo Stock exchange prices index prediction system. In other work, Chiang *et al.* [91] have used ANN to forecast the end-of-year net asset value of mutual funds. Trafalis [92] used feed-forward ANN to forecast the change in the S&P(500) index. In that model, the input values were the univariate data consisting of weekly changes in 14 indicators. Similarly, Choi *et al.* [93] forecasted the daily direction of change in the S&P 500 index futures using ANN. Wu and Lu [94] developed a forecasting model based on ANN. They implemented this model aiming at forecasting stock-market, specifically the S&P500 indices. They found that both ANN and ARIMA model fail to predict the market if there exist irregular fluctuation. However, in this case ANNs performed slightly better. For some cases, the ANN forecast random wild values, and therefore recorded poor forecast performance.

Limitations

Although ANNs have been used a lot for predicting the time series data, researchers have put much effort in finding the best ANN for the respective problem. The main hurdle dealing with the ANN is to find the best architecture for a problem. Several approaches have been adopted to identify the suitable ANN architecture when predicting the time series data. Oliveira *et al.* [64] adopted a trial and error method to find a better ANN architecture. The study, however, did not preprocess/filter the input data space, which could have improved the performance of the model. Farway and Chatfield [57] considered to choose the input variables along with the architecture and activation function of the ANN to select the 'best' model. The study revealed that even the best ANN model failed to give better forecasts when faced with an increased number of parameters.

Apart from finding the best architecture, some research focused on the preprocessing the input dataset as was done by Geva [63]. After preprocessing the input data, improved prediction accuracy is achieved. However, what preprocessing should be performed and the scale of data preprocessing is still an open question. In another study, Nelson *et al.* [56] achieved an improved performance by deseasonalizing the data. Still, the problems of choice of data preprocessing and finding the best ANN architecture remain. Kimoto *et al.* [90] investigated the effect of alternative learning algorithms on the prediction accuracy and Abraham [81] addressed the choice of alternative learning algorithms and also tried to find the best architecture and activation function. Being a 'black box' model the challenge of understanding the whole process is yet to be met.

Most of the studies on ANN reviewed in this chapter report that the performance of an ANN was superior to the traditional statistical model, except the study by Balkin and Ord [20]. Their research indicates that the application of ANN is effective only when the process is non-linear. In contrast Wu and Lu [94] identified that the ANN cannot predict a given time series if there exist irregular fluctuations.

To overcome some of the reported limitations of ANNs, hybridization of ANN with GA has been reported by a number of researchers [95–97]. When using the evolutionary approach caution should be exercised about designing the cost function. For instance, Hansen and Nelson [96] used the GA to optimize the ANN architecture only while the cost function was to obtain improved prediction accuracy. This study did not consider the effect of other ANN parameters e.g. initial weights, activation function and learning algorithms on the ultimate performance of the ANN. In another study, Mayer and Schwaiger [98] used the GA not only to optimize the ANN architecture but also to determine the size and composition of the training dataset. However, the resultant ANN may not be the best, because there are other factors, e.g., the activation function, that govern the choice of the type of the ANN. The EPnet system [99] is another evolving weighted ANN structure. This study did not call attention to choosing the optimal activation function and the selection of the composition of training dataset as well. Referred by Mayer and Schaiger [98], DeFalco *et al.* [100] trained the fixed multi-layer perceptron architecture by evolving the weights of the connections.

GA is one of the most popular searching algorithms that has been applied to optimize ANN though, the performance of the overall hybrid system depends on several factors, i.e., what to optimize, how to use GA in the optimization. Thus, it is difficult to obtain the

best ANN model with all the factors optimized. Furthermore, the evolutionary approach does not always guarantee to give the best optimal (global) solution for a given problem because of the heuristic nature inherent to this search technique.

2.3.2 Use of Fuzzy Logic

Fuzzy Logic (FL) is a SC paradigm that has been successfully applied to forecast time series with accurate results as compared to classical methods. Including linguistic concepts in modeling a relationship between predictors and desired values for a time series data allows us to handle the imperfection or incompleteness of information comprising the data collection. Applying FL to predict or forecast time series data results in fuzzy numbers rather as opposed to precise numeric values. After defuzzifying the resultant fuzzy numbers it was noted that an accurate numeric prediction can be obtained [15]. In the sequel we briefly describe some studies where FL was used to analyze, predict and forecast time series data.

FL for time series analysis in health sector

FL has been applied in the analysis and diagnosis of patients with time series sequential data pertaining to a specific patient under consideration [101–104]. A technique based on the concept of a 'fuzzy trend template' that identifies characteristic patterns in multiple time series was developed by Lowe *et al.* [105]. This technique was applied to diagnosing specific anesthesia related problems and has a documented sensitivity and specificity of 95% and 65% respectively. Referenced by Lowe *et al.* [105], FL has also been used in several studies that implemented an intelligent patient monitoring system [101–104]. The fuzzy rules that these systems use were generated based on expert knowledge documented in interviews.

In a recent year, Tvasruskó *et al.* [106] used a FL based dynamic object tracking system to analyze and visualize dynamic processes in living cells. They proposed and developed a new technique for time-resolved image analysis and continuous time-space visualization for quantitative and visual interpretation. Based on this new concept of tracking the object in a dynamic environment, a FL based model was developed.

FL applied to solving other real world time series problems

Time series comprising temperature, weather and the environment have also been analyzed and forecast with FL applications. Chen and Hwang [25] derived a FL based forecasting model they called the 'Two Factors Time-Variant Fuzzy Time Series Model'. Their model overcame a hurdle faced by most traditional forecasting methods, viz. the problem of dealing with historical data represented by linguistic values. Their aim was to develop a model for temperature prediction that showed high accuracy in its forecasts.

Nayak *et al.* [23] experimented with applying FL to flood forecasting. The team analyzed the problem and developed a FL computing strategy that worked on the rainfall-runoff model to give real time flood forecasts. The famous Narmada basin in India was the testing ground where the approach was evaluated. This study illustrated the potential of fuzzy models to simulate unknown relationships between a set of relevant hydrological data like rainfall and river flow. The fuzzy model was obtained by a subtractive clustering [107] based fuzzy model generation approach. During the defuzzification process, Takagi-Sugeno's Fuzzy Inference was applied (details about the Takagi-Sugeno fuzzy inference are presented at the beginning in Chapter 6).

The Fuzzy approach has also been successfully applied in predicting the time series of university enrolments. Chen [108, 109], used the concept of fuzzy time series first introduced by Song and Chessom to forecast university enrolments. Chen's model was evaluated and validated with the forecast results published by Song and Chissom [110]. This model was found to be robust even when the historic data was inaccurate or 'noisy'. Hwang *et al.* [111], in an independent study, developed a new method based on the time series concept. They fuzzified the historic enrolment data with a heuristic rule that differed from the actual study [110, 112]. In another study, Huarng [113] studied this very dataset, but instead he simplified Chen's time series model integrating domain-specific heuristic knowledge. It was found that by integrating domain-specific knowledge with Chen's model resulted better performance compared to the basic model. Recently, Cheng *et al.* [26] used the trend-weighted fuzzy time series model to forecast university enrolment data and the Taiwan Stock Exchange Index (TAIEX). This model beat all others during a reported forecast of enrolment trends at the University of Alabama in the USA.

Fuzzy Logic Advisory Tool (FLAT) was a FL based software tool that could forecast and control complex and extremely non-linear materials purchasing system for a well known telecommunication brand: Nokia. The FLAT combined the traditional fuzzy set

theory with the linguistic equation approach to create an Adaptive Expert System. This expert system could be used to forecast market demand for signal transmission devices within a certain market area, including both products and computers.

In the last few years, the fishery industry has taken to FL applications to analyze and predict the future value of the considered event in a big way. Two years ago, Cheung *et al.* [114] estimated the intrinsic vulnerability to fishing with a fuzzy expert system. The system integrated life histories and ecological characteristics of marine fauna to estimate vulnerability levels. Input variables included ecological characteristics and features selected from the life history that were fuzzified with membership functions. Outputs were expressed in four linguistic categories that each referred to a certain level of vulnerability to extinction: very high, high, moderate, and low. The Jack-Knife Scheme for Data Features from FishBase [115] and cross validation were used to evaluate the system's performance.

CLUPEX, a FL based expert system, was developed by Mackinson [116] to bridge existing gaps and at the same time documenting the biological mechanisms comprising the behavioral responses of herring, and the overall effect these mechanisms work within the spatial dynamics of a shoal of herring. The system developed used FL to record and combine scientific and local fisherman-knowledge in order to illustrate the herring shoal structure, and understanding its dynamics and distribution. Input data included information about both biotic and abiotic environmental factors. CLUPEX used heuristics to predict structure, dynamics and the mesoscale distribution of migratory herring shoals at different stages of their life cycle. An earlier study by Mackinson *et al.* [117] analyzed the stock recruitment relationships and predicted recruitment with techniques that applied principles of the fuzzy set theory. Their system took into account the effects of nonstationarity with its incorporation of fuzzy rules. Chen [118] evaluated the impact of environmental conditions on fish stock recruitment relationships with a FL based model he developed. This approach differs from the traditional approach in FL's usage of a continuous membership function. This approach was applied and tested on herring stock from the west coast of Vancouver Island using the Pacific Decadal Oscillation as the environmental variable. The researchers discovered that the FL model they developed consistently beat traditional approaches based on several evaluation criteria.

The generalized Takagi-Sugeno (GTS) fuzzy model (details about the Takagi-Sugeno model has been presented at beginning in Chapter 6) was proposed by Fiordaliso [119] that introduce a convex combination of input patterns rather than using affine functions

to obtain crisp output. This allowed for a better interpretation and analysis of the fuzzy rules. This modified model has been used to combine forecasts of individual models and has proven itself a fully transparent model. The developed GTS system used a decremental (i.e. pruning) algorithm (DEC) for generating fuzzy rules. The GTS DEC model was tested with the Mackey-Glass time series data [62] and the laser time series from [120]. The test result obtained using the dataset considered was quite encouraging.

FL in financial forecasting

Aside from the above mentioned time series data, FL has also been applied to the stock market and several studies have been made on such applications. For example, FL was applied to the decision making process in the stock trading field by Simutis [24]. Simutis discovered that the FL based expert system could reproduce information almost as well as an experienced stock trader. This FL based stock trading system, christened STRASS, was tested with historic data from NASDAQ, NYSE, and AMAX stock records, and they yielded about 30% annual paper profit. While designing the fuzzy model, eighty one fuzzy rules were generated with knowledge gleaned from a skilled stock trader. Lee and Kim [121] designed and implemented a fuzzy knowledge based tool that analyzed investment stock. The model used multiple fuzzy cognitive maps: a fuzzified version of a cognitive map, usually a fuzzy signed directed graph, in order to build a causal knowledge base for stock investment analyses.

A sales forecasting system that integrated ANNs and FNNs (Fuzzy Neural Networks) with fuzzy weight elimination was developed a few years ago by Kuo *et al.* [122]. The system comprised three broad parts: (1) The Data Collection Module, (2) A Special Pattern Model (FNN), and (3) A Decision Integration Module. The system determines the qualitative factors affecting the sales first and then a feed forward neural network that works on an error back-propagation learning algorithm is used to take into consideration the expected/observed effect. The If-Then rules used to collect data are generated by the Fuzzy Delphi method (Delphi method is a process of accumulating information regarding the solution of a problem, from experts and other people without necessarily meeting with the respective person face to face [123]). FNN architecture is developed based on these If-Then rules. To fine tune the FNN, weights that tend to zero or are below a pre-determined threshold are assumed to be the source of bias and removed during training. Finally a neural network integrates the promotional effect gotten from the FNN with the time series

to bring out a prediction of the final promotional effect on sales.

Huarng and Yu [124] preferred to use a Type 2 Fuzzy Model – which is an expanded version of the primary fuzzy model for stock index forecasting. Applying the Type 2 Fuzzy Model provided more information that enriched the fuzzy relationships begotten using the primary fuzzy model (Type 1). This approach extends fuzzy relationships determined using the primary Type 1 Fuzzy Model. New fuzzy operators are defined to handle relationships generated from both Type 1 and Type 2 observations. Forecasts that are made using the dataset from TAIEX (Taiwan Stock Exchange) show that Type 2 Fuzzy Models generally bring out better results compared to Type 1 Models.

Applications of the Takagi-Sugeno Fuzzy model to Financial Market Modeling was studied in detail by Setnes and Drempt [125]. Their approach was to generate different fuzzy model structures and evaluate them against the Dutch-AEX price index. Models were considered based on how they performed to forecast the market behavior where the prediction models were cosidered based on historical data. Monthly price indices from the Dutch-AEX from 1990 through 1996 were used in developing the model. Input variables were chosen with a variable selection method and in order to manage time variances in the time series data, a moving window (Size 35) was used for training to forecast the monthly observations at a time index 36.

The application of fuzzy sets in incorporating uncertainty into decision making problems was broadly explored by Tarrazo and Gutierrez [126], and they most specifically dealt with financial planning model related problems. The approach they came up with was capable of making inferences related to financial planning under conditions of uncertainty. This study used fuzzy sets to handle the inaccuracy and uncertainty of financial planning, but did not attempt to generate fuzzy rules to build a fuzzy model. Alternately, Fuzzy Calculus (the calculus that uses fuzzy numbers and fuzzy sets instead of crisp numbers and classical set) was used to infer future financial planning directions, in other words, the inaccuracies and uncertainties were fuzzified.

Fuzzy numbers have also been used to select portfolios (security). Vercher *et al.* [127] wanted to minimize the trade-off between maximum returns and minimum risk for investments in a certain security. Applying a FL methodology allowed them to incorporate uncertainties in datasets, and to include influential subjective matters into models. The methodology they developed essentially generalized the classical mean-variance model with fuzzy numbers. The membership functions that fuzzified real world data were trapezoidal

and two objective functions were presented to de-fuzzify the fuzzy decision into a crisp value.

A decision support system for demand forecasting with fuzzy If-Then rules was developed by Petrovic *et al.* [128]. Called DSS_DF, the system used a modified Mamdani-Fuzzy model (Details about the Mamdani-Fuzzy model is presented at the beginning in Chapter 6) to generate a forecast. It may be noted that the four inputs for this model were forecast values determined using four different pre-existing time series forecasting methods:

1. Forecasts made by the Customer,

2. Forecasts made by the Market Expert,

3. Forecasts that used time series analyses based on decomposition (TSAD), and

4. Forecasts that used an auto regressive integrated moving average (ARMA) model. Fuzzy rules were generated with heuristics and the authors' expert knowledge on the matter.

Trend-Weighting was incorporated into fuzzy time series models to gain a high forecast accuracy by Cheng *et al.* [26]. Their model partitioned the universe of discourse into seven linguistic values based on the theories by Miller [129]. In such an environment, the relationships amongst the fuzzy linguistic terms are classified as three trends, and the counts of these trends are converted into incremental weights that are attached to their respective fuzzy relationships. Users' opinion regarding confidence levels of decisions are also incorporated into the system. Test results of a forecast for Taiwan Stock Exchange (TAIEX) and of another one for university enrolment suggest that this model outperforms others designed for the same purpose.

A Parallel-Structure Fuzzy System (PSFS) was developed and used to predict sunspot cycles in a certain railway's communication and power system by Kim [130]. The system used a smoothed sunspot number time series. The PSFS comprised of several fuzzy systems connected in parallel. Each fuzzy system made independent predictions for the same future data, and the PSFS decided the final predicted value after evaluating all predictions by each independent fuzzy system. Subtractive clustering based methods are used to generate fuzzy rules in the PSFS. Test results for the sunspot time series data suggest that PSFS is capable of successfully predicting the smoothed sunspot numbered area.

Limitations

As evident by the studies reviewed in this section FL has been used successfully on non-linear time series. The benefit of using FL is that it deals with the vague information regarding the solution of a problem and the solution can be represented using human interpretable language. If an ANN is perceived to be "black box" then FL may be considered "white box".

Existing limitations regarding to FL are reported here. One of the limitations of FL is generating appropriate fuzzy rules. Usually, fuzzy rules are generated using two alternative ways: (1) Using expert knowledge [101, 102] and (2) Analyzing the available data-*data-driven fuzzy model* [131, 132].

It is easy to obtain expert knowledge or heuristics based knowledge for controlling a non-linear physical object (e.g., PID, Washing Machine, Vehicles etc.). However, in terms of time series data the representation of available expert knowledge into FL may effect on the overall performance of the system. The appropriate fuzzification of data represented by linguistic values is also a challenge [25]. The new concept of fuzzy time series [109] has also been used in forecasting time series data, but the proper fuzzification of the time series data restricts the overall performance. In continuation of improving this approach of dealing with fuzzy time series, the weighted fuzzy time series [26] was found superior than the other improved versions [111, 113] of the approaches. However, imposing the weights is subjective and demands more research.

For a non-linear time series dataset, without expert knowledge, it is a difficult task to identify the exact knowledge that controls the overall process. Consequently, generating rules using the*data-driven approach* is still a challenge. Fuzzy rules generated using ad-hoc process is one way to build the fuzzy model. One such approach adopted in the STRASS [24] yielded a good profit margin with expense of generating 81 fuzzy rules. The large number of fuzzy rules not only brings down interpretability but also increases the system's complexity. The integration of ANN with Fuzzy approach [122] is quite appealing. However, the deletion of weights is somehow ad-hoc that may be responsible for the degradation of the overall performance as well. The type 2 fuzzy model [124] has met with success since its introduction but it is quite probable that the increased computational complexity may restrict its application in real world situations.

Regarding the automated generation of fuzzy rules, a number of approaches have been developed in the last decade. The most popular approach is to divide the data space using

a clustering algorithm. In this case the performance of the fuzzy model depends on the appropriate choice of the number of clusters [131], because each cluster of data is used to generate a fuzzy rule. Popular clustering techniques used for generating fuzzy rules are the fuzzy c-means algorithm [133, 134] and the subtractive clustering algorithm [107].

A second technique commonly used to generate fuzzy rules is the grid partitioning technique [135]. In this method, each input feature is used to generate independent membership function. Consequently, this process suffers from exponential rule explosion with an increasing the number of input features. Moreover, this method may generate rules where no data instances appear. Referenced by [132] an example of rule extraction method using this approach is the learning from example [135] and its modification [136].

2.3.3 Use of Evolutionary Algorithm

Being efficient in obtaining the optimized solution to a problem, EA has been widely used in a hybridized form with other SC paradigms to analyze time series data. However, few studies have attempted to apply the EA independently to forecast time series data. Some representative studies are described here.

Szpiro [137] proposed the use of GAs to find equations that describe the behavior of a times series. The method was used to perform global forecasts of the time series. The advantage of the method was that very little data was needed to train the method and, besides, it could indicate the dynamics within the time series in a functional form. In the algorithm, the equations that mimic the dynamics within the time series are searched for using the GA. The strings/individuals in the population of the GA represent equations. To overcome the over-fitting problem, the string equation that scores the top value is further tested on an unknown test set and the predictive power of the respective equation is rechecked with the new data. The algorithm was validated using both clean and noisy chaotic data, and also with the sunspot series.

Cortez *et al.* [138] used both the Genetic and Evolutionary algorithms (GEA) to forecast several time series. In the model, the feature analysis based on statistical measures is used for dimensionality reduction. The fitness (cost function) of the GEA is the quality measure; i.e. the measure of each solution for an individual in the population. A reasonable alternative to this is the use of simple statistics, but it brings with it a penalty that

is a function of model complexity, such as the Bayesian Information Criterion (BIC).

$$BIC = N.ln(\frac{SSE}{N} + p.ln(N) \tag{2.9}$$

where, N denotes the number of training examples, SSE is the sum of the squared error and p is the number of parameters in the forecast tool. To evaluate the GEA time series forecast tool, the two models - linear combination based GEA and ARMA based GEA - have been considered.

A new GA based rule generation to predict the future performance of stock was proposed by Mahfoud and Mani [139]. Their GA operates upon all of the attributes in the dataset and finds the favorable or unfavorable circumstances reflected by the relationships between the respective attributes and the output variable to be predicted. This differs from the traditional GAs, which performs optimization. The GA system is compared to an established NN system in the domain of financial forecasting, using the results from over 1600 stocks and roughly 5000 experiments.

Jeong *et al.* [140] proposed the guided GA (GGA) for predicting supply chain management measurements. The guided GA was used to determine the optimal co-efficient of a linear causal forecasting model. In the GGA, penalty operators in the fitness function and the population diversity index (PDI) were used to overcome the premature convergence of the algorithm. The empirical results illustrate that the GGA outperforms the canonical GA (CGA).

2.4 Summary

In this chapter, we have reviewed the applications of SC paradigms in the analysis of time series data. We identified that none of the paradigms can forecast within the acceptable level due to their own limitations. Among the SC paradigms, ANN has the adaptive process to find the complex relationship between the set of input variables and the output. However, the successful usage of ANN is constrained by several parameters that need to be adjusted to best suit the problem. FL, on the other hand, is interpretable and more transparent than the ANN. Nevertheless, the generation of appropriate fuzzy rules from the available (training) time series data remains a challenging task. Few studies have used GA independently to analyze the time series data, because it is difficult to define the cost/objective function for the GA without using any forecasting/prediction model.

Our objective in this thesis is to develop an HMM-based forecasting method using the rich statistical and mathematical foundation behind the HMM. We have used the SC paradigms to improve the HMM's performance for forecasting time series data. Following a description of the background theories of HMM in the next chapter, we shall proceed towards describing our HMM-based methodologies in the subsequent chapters.

Chapter 3

Hidden Markov Models with application to time series forecasting

In this chapter, we describe the Markov process, Hidden Markov Models (HMMs) and review some studies that have used HMMs for forecasting time series data. As the Markov process is the basis of HMM, we introduce the Markov process at the beginning and then provide a thorough introduction to HMMs. The challenges in handling the continuous dataset using the HMM are also described in this chapter. Finally, we review the application of HMMs in forecasting time series data and point out the limitations.

3.1 Markov Process

A *Markov process* is a stochastic process where the future event depends on the immediate preceding event. That is, a Markov process [2] assumes that the probability of the occurrence of an event in the next will solely depend on the occurrence of the current event. Any process that follows the Markov property is called a *Markov process*. Thus any typical time series sequential data where the most recent j events have the highest influence on the conditional probability of the current event, given all the past and present events, is called a *Markov process*.

To consider a simple example, suppose that there is a sequence of letters as given in Fig. 3.1. In Fig. 3.1, the total number of changes, as the next step value in the sequence, from A to B is 2; from A to C is 2; from B to A is 2; from B to C is 1; from C to A is 1 and from C to B is 1. These values can be used to find the conditional probabilities for the sequence shown in 3.1. The conditional probabilities are:

AACBACCABBAABC

Figure 3.1: A sequence of letters: A, B and C

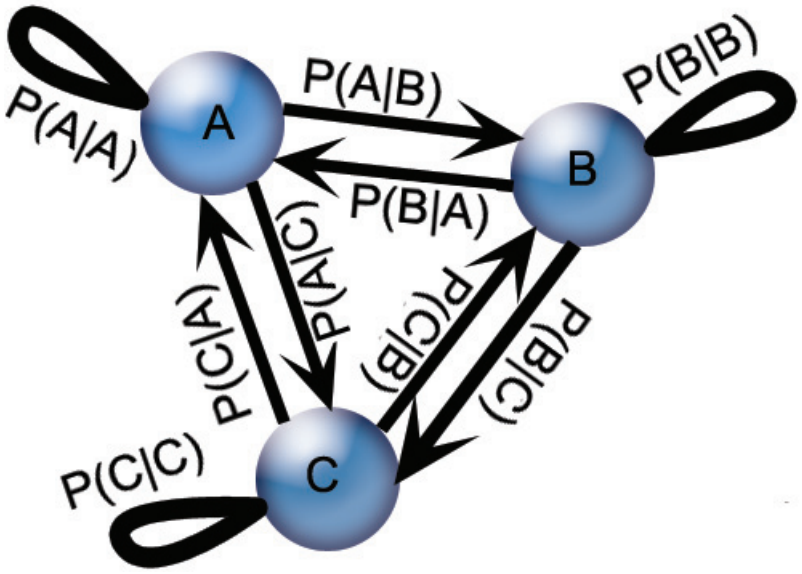


Figure 3.2: The Markov process illustrated by directed graph

$$\Pr(A|A) = 2/14; \Pr(A|B) = 2/14; \Pr(A|C) = 1/14$$
$$\Pr(B|A) = 2/14; \Pr(B|B) = 1/14; \Pr(B|C) = 1/14$$
$$\Pr(C|A) = 2/14; \Pr(C|B) = 1/14; \Pr(C|C) = 1/14$$

The sequence in Fig. 3.1, representing a Markov process can also be illustrated using a directed graph as shown in Fig. 3.2.

In Fig. 3.2, each node represents a state from which a specific symbol is emitted. For instance, the node labeled as A emits the symbol A. The lines connecting two nodes represent the probability of changing states; i.e., the state transition probabilities are represented by each of the edges connecting two states. The state transition probabilities can also be represented in a matrix form, as is shown in Fig. 3.3. In a Markov process, the sum of the row elements in the state transition matrix must be equal to one, since the state transition event is expressed in probabilities.

$$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} \Pr(A|A) & \Pr(A|B) & \Pr(A|C) \\ \Pr(B|A) & \Pr(B|B) & \Pr(B|C) \\ \Pr(C|A) & \Pr(C|B) & \Pr(C|C) \end{bmatrix} \end{array}$$

Figure 3.3: State transition matrix

3.2 Hidden Markov Model

Markov models are useful where the dataset can be used as a sequence of some fixed discrete symbols [2]. In a Markov process, everything is visible. This process is too simple to model many real world time series problems where there may be a number of factors which influence the observable sequence [2]. To handle more complex problems (e.g., time series data), a double stochastic process called HMM seems to be more suitable [28, 141], as in HMM the states are hidden. HMM can be used to generate the observation sequence, while considering the unobservable events. Here the unobservable events are represented by the hidden state sequence. Each of the hidden states has their own observation emission probabilities. The hidden process itself is a Markov process; i.e., the next state is dependent on the current state and the transition probabilities [142]. For a better understanding of how this works a brief description about HMM is presented below.

An HMM is a finite automata with double probabilities. The double probabilistic approach of combining the observations/data of a process into states [143] make it an efficient method for handling noisy data. The variations among the observations are represented by the transitions between states [144]. In HMM, the states are hidden and only the observations are visible.

There are a number of features of HMM that can be adopted to solve real world problems. Depending on the types of data they handle, HMMs can be either discrete HMMs or continuous HMMs. Discrete HMM is the basis of continuous HMM and is easier to understand. In the following we describe the details of the discrete HMM and then proceed to the continuous HMM.

3.2.1 Discrete Hidden Markov Model

In discrete HMM, as the name suggests, the observations are assumed to be discrete. For a better understanding about the HMM, let us consider an example. Assume there are N persons in a closed room (see Fig. 3.4). Each person is holding a number of colored sticks

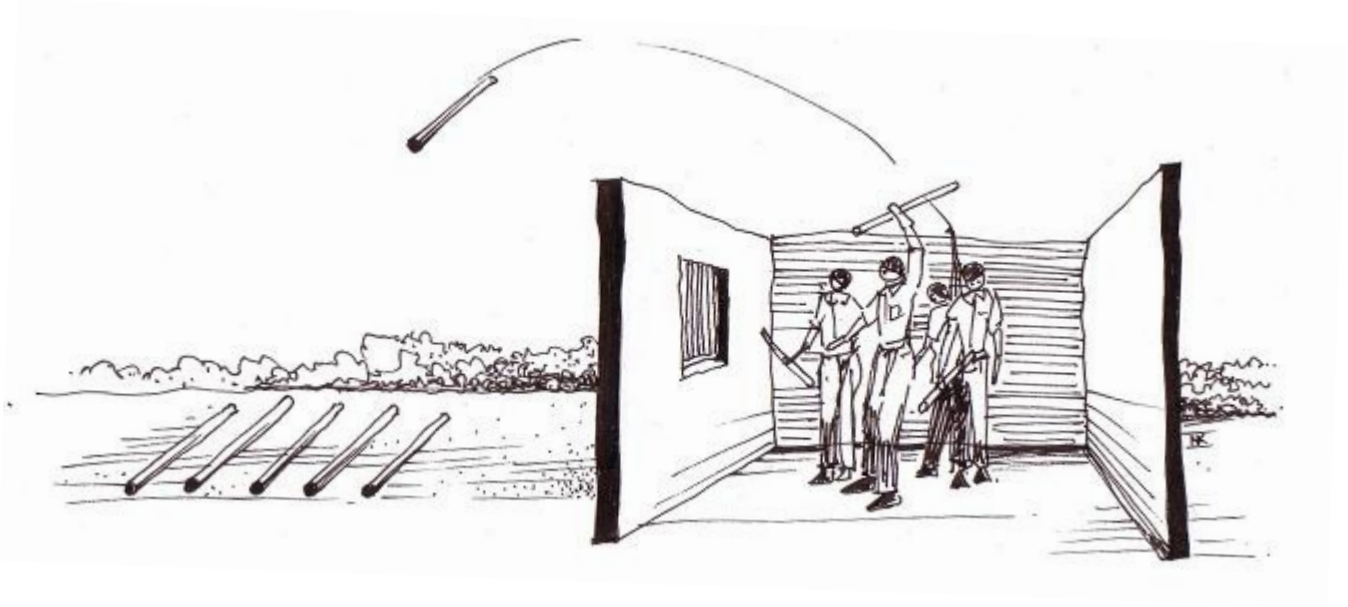


Figure 3.4: The person and stick model

(of M distinct color). Now, each person throws the colored sticks, one after another, out of the room. At this stage, the only visible outcome is the sequence (order) of colored sticks received. We do not know who (out of N persons) has thrown which colored sticks.

In this example, both the selection of a person and color are completely random. The individuals are hidden from the view, making it a hidden process. The overall process generates a sequence of colors/colored sticks, which forms the output observation sequence [145]. For this process, we do not know the sequence in which a person is throwing the colored sticks, nor the sequence of persons throwing the sticks [146]. The output sequence is very much dependent on the transition probabilities between the various persons/states, and the choice of initial person/state at the beginning. This example can be associated with an HMM, where the states are hidden (like the persons in the example) and the output is the sequence of observations [147].

Formally, according to Rabiner [147], an HMM is characterized by the following attributes:

1. N, the total number of states (the number of persons in the above example). In most applications these states are hidden, though states may represent some significant physical meaning attached to the problem.

2. M, the number of distinct observation symbols per state (the number of distinct

colored sticks) that correspond to the discrete observations in the real world problem.

3. A, the state transition probability (the probability of switching persons) matrix, where $a_{ij} \in A, a_{ij}$ = probability of changing state i to j at time t, and $1 \leq i, j \leq N$

4. B, the observation emission probability (the probability of a specific person throwing a specific color of stick) matrix, where $b_j(k) \in B$ and $b_j(k)$ = probability of selecting observation symbol x_k at time t, from the state j. The values of j and k vary as: $1 \leq j \leq N$ and $1 \leq k \leq M$.

5. π, the prior probability matrix, which represent the initial state transition matrix, where, $\pi_i \in \pi$ and π_i = probability of selecting state i at time 1, $1 \leq i \leq N$.

An HMM is generally referred to as a set, $(\lambda = A, B, \pi)$, where the three parameters describe the HMM for a specific problem. The structure of an HMM for a problem is dependent on the choice of the number of states and the number of distinct observable components. For instance, in the above person and stick model, we can build an HMM containing $N = 3$ states, if there are 3 persons. We assume that there are 7 distinct colors of sticks held by the persons. Hence, the number of distinct observations here would be $M = 7$. The observation emission probabilities for each of the colored sticks, which would correspond with the observation symbol x_k, from each of the states would correspond with $b_j(k)$, where, $b_j(k)$ is the probability of emitting the specific color symbol (stick) that has the index k (index the color set red, yellow, blue, green, black, white and orange into [1 2 3 4 5 6 7] respectively) from a specific state (person) with the index j (index the three persons as [1 2 3] respectively).

Initially, one individual is selected following the prior probability distribution π. Let us assume the first stick is always thrown from the person with index 2. Thus the prior probability matrix would be as shown in Fig. 3.5. We assume the state transition probability matrix and the emission probability matrix are as shown in Fig. 3.6 and 3.7. The graphical representation of the HMM for the person and stick model is shown in Fig. 3.8.

Thus the parameters that guide us to build the HMM for a given observed sequence are the prior probabilities, state transition probabilities, and the observation emission probabilities. These parameter values adapt with the available training dataset during the training of the model which we discuss below.

$$\pi = \begin{bmatrix} \Pr(Person1) \\ \Pr(Person2) \\ \Pr(Person3) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Figure 3.5: Prior probability matrix

$$\begin{array}{c} \\ Person1 \\ Person2 \\ Person3 \end{array} \begin{array}{c} \begin{array}{ccc} Person1 & Person2 & Person3 \end{array} \\ \begin{bmatrix} 0.2 & 0.4 & 0.3 \\ 0.1 & 0.5 & 0.4 \\ 0.4 & 1.5 & 0.1 \end{bmatrix} \end{array}$$

Figure 3.6: State transition probability matrix

$$\begin{array}{c} \\ red \\ yellow \\ blue \\ green \\ black \\ white \\ orange \end{array} \begin{array}{c} \begin{array}{ccc} Person1 & Person2 & Person3 \end{array} \\ \begin{bmatrix} 0.09 & 0.6 & 0.01 \\ 0.1 & 0.05 & 0.4 \\ 0.03 & 0.03 & 0.3 \\ 0.07 & 0.02 & 0.2 \\ 0.6 & 0.17 & 0.01 \\ 0.01 & 0.08 & 0.02 \\ 0.1 & 0.05 & 0.06 \end{bmatrix} \end{array}$$

Figure 3.7: Observation emission probability matrix

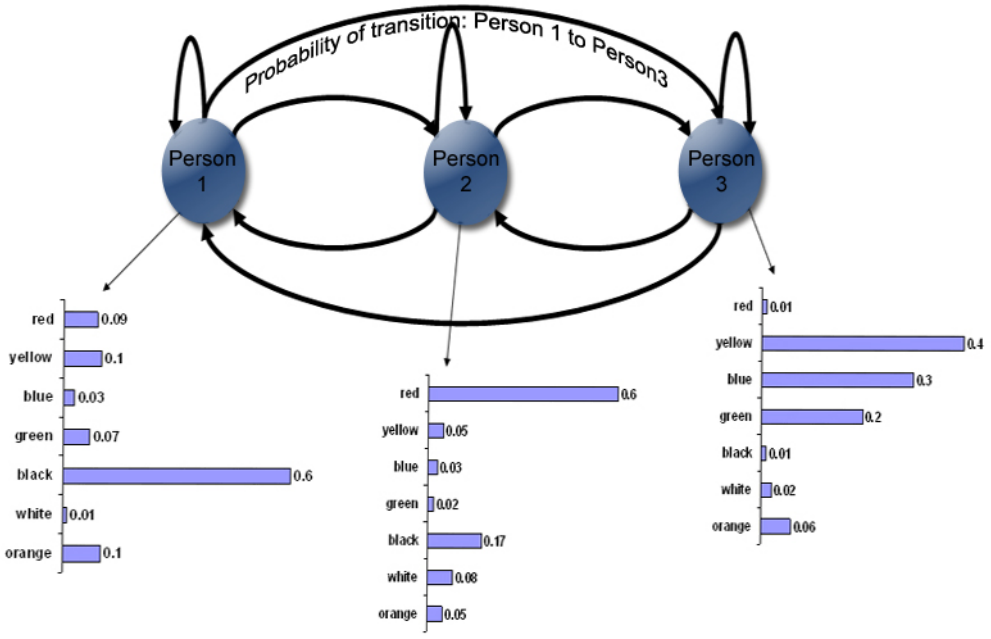


Figure 3.8: The HMM representation for the person and stick model given the parameter values of A, B and π

HMM training

When training the HMM for a given problem/dataset, the joint probability over hidden states (S_i) and observed symbols (O_i) are taken into consideration and the properties of the Markov chain embedded in the sequential data is used to simplify it. The obtained joint probability is:

$$\Pr(S, O) = \Pr(S_1) \prod_{t=2}^{T} \Pr(S_t|S_{t-1}) \prod_{t=1}^{T} \Pr(O_t|S_t) \tag{3.1}$$

where,

S = sequence of states in the time period 1 to T : $S_1, S_2, ..., S_T$,

O = sequence of observation in the time interval 1 to T: $O_1, O_2, ..., O_T$.

The simplified representation of Eq. (3.1) can be obtained from one of the solutions of the three well known problems with HMM. The three key problems are:

1. Given the HMM: $\lambda = (A, B, \pi)$ the computation of the probability of the observation sequence O, i.e., $\Pr(O|\lambda)$, where, O = Observation sequence in the time duration from 1 to T: $O_1, O_2, ..., O_T$.

2. Assuming that we have the observation sequence O and the model λ, selection of the best state sequence S in the time duration 1 to T that will most likely generate the observation sequence O.

3. Assuming we have the observation sequence O and the initial model λ, with the parameter values of A, B, π; the re-estimation of the parameter values so that the model λ best explains the observation sequence with the modified parameter values.

Mathematical solutions to the problems 1 and 3 are presented in the following sections. Problem 2 is of little interest for the proposed models in this thesis and hence is omitted.

Probability of observation sequence computation

To calculate the probability of an observation sequence, $O = O_1, O_2, ..., O_T$, we assume that the complete model, λ, is known to us; i.e., the parameter values of A, B, π are known. The calculation can be done by considering each and every possible state sequence that can produce the observation sequence O given λ. The sum of the obtained probabilities

is the probability of generating the observation sequence O by the model, λ. A formal description is presented below.

Assume that one of the state sequences that might generate the observation sequence O is as follows:

$$S = S_1, S_2, ..., S_T \tag{3.2}$$

For the state sequence in the Eq. (3.2), the probability of generating the observation sequence O is:

$$\Pr(O|S,\lambda) = \prod_{i=1}^{T} \Pr(O_i|S_i,\lambda) \tag{3.3}$$

$$= b_{S1}(O_1)b_{S2}(O_2).....b_{ST}(O_T) \tag{3.4}$$

The probability of the state sequence S for the model λ is calculated using the following equation:

$$\Pr(S|\lambda) = \pi_{S1}a_{S_1S_2}a_{S_2S_3}....a_{S_{T-1}S_T} \tag{3.5}$$

Note that Eq. (3.5) is already embedded in Eq. (3.1) as we get the following equations:

$$\begin{aligned}\Pr(S,O) &= \Pr(S_1)\prod_{t=2}^{T}\Pr(S_t|S_{t-1})\prod_{t=2}^{T}\Pr(O_t|S_t) \\ &= \pi_{S1}a_{S_1S_2}a_{S_2S_3}....a_{S_{T-1}S_T}b_{S1}(O_1)b_{S2}(O_2).....b_{ST}(O_T) \\ &= \Pr(S|\lambda)\Pr(O|S,\lambda)\end{aligned} \tag{3.6}$$

Again, we get the following joint probability, which calculates the probability of the observation sequence O and the state sequence S simultaneously, as in the Eq. (3.7):

$$\Pr(O,S|\lambda) = \Pr(O|S,\lambda)\Pr(S|\lambda) \tag{3.7}$$

The probability of the observation sequence O from the model λ is obtained by taking into consideration every possible state sequence that can produce the sequence O. From this we get the Eq. (3.8) and Eq. (3.9):

$$\Pr(O|\lambda) = \sum_{\text{all S}} \Pr(O|S,\lambda)\Pr(S|\lambda) \tag{3.8}$$

$$= \sum_{S_1,S_2,....,S_T} \pi_{S1} a_{S_1S_2} a_{S_2S_3} ....a_{S_{T-1}S_T} b_{S1}(O_1) b_{S2}(O_2) .....b_{ST}(O_T) \tag{3.9}$$

For a better illustration of the computation of Eq. (3.8) and Eq. (3.9), let us further consider the person and stick model described at the beginning of this section. Considering the parameter values given in Fig. 3.5, 3.6 and 3.7, let us compute the probability of the observation sequence $\{red, green\}$ using the above equations.

Here, the Observation sequence $O = \{red, green\}$. The probability of finding the sequence O, for the state sequence of all S i.e., sequences formed by the states $person1$, $person2$ and $person3$ is:

$$\begin{aligned}
&\Pr(O|\{person1, person2\}) = 0 \\
&\Pr(O|\{person1, person1\}) = 0 \\
&\Pr(O|\{person1, person3\}) = 0 \\
&\Pr(O|\{person2, person1\}) = 0.6 * 0.1 * 0.07 \\
&\Pr(O|\{person2, person3\}) = 0.6 * 0.4 * 0.2 \\
&\Pr(O|\{person2, person2\}) = 0.6 * 0.2 * 0.02 \\
&\Pr(O|\{person3, person1\}) = 0 \\
&\Pr(O|\{person3, person2\}) = 0
\end{aligned} \tag{3.10}$$

Now, the total probability becomes:

$$\begin{aligned}
\Pr(O|\lambda) &= 0.6 * 0.1 * 0.07 + 0.6 * 0.4 * 0.2 + 0.6 * 0.2 * 0.02 \\
&= 0.0546
\end{aligned} \tag{3.11}$$

Note that in Eq. (3.10) the probability of the observation sequence O for all possible states has been calculated and then, in the Eq. (3.11) the sum of all probabilities is calculated to obtain the probability of the observation sequence O for the HMM, λ. The computational complexity in calculating the probability using Eq. (3.2) to Eq. (3.9) would be quite high. For instance, for an N state HMM, the computational complexity is in the order of $2\,T.N^T$ calculations [147]. Hence, an alternative approach for calculating the probability of an observation sequence for an HMM is required.

The forward-backward algorithm [148] is used as an alternative in calculating the

above mentioned probability with relatively less computations. This algorithm is described below.

The forward variable $\alpha_t(i)$ is defined as:

$$\alpha_t(i) = \Pr(O_1, O_2, ..., O_t, S_{t=i}|\lambda) \tag{3.12}$$

Here, $\alpha_t(i)$ indicates the probability of the partial observation sequence; i.e., the observation sequence up-to time t and the state i that is assumed to reach at the same time t, for a given model λ. This probability is calculated inductively as follows:

Forward-Backward algorithm

Step 1 : **(Initialization)** $\alpha_1(i) = \pi_i b_i(O_1)$ for all states i, where, $i \in \{1 \text{ to N}\}$

Step 2 : **(Induction)** Obtain the values of $\alpha_t(j)$ for each time unit t, i.e., for $t = 2, ..., T$, and all state j ($j \in \{1 \text{ to N}\}$), compute:

$$\alpha_t(j) = [\sum_i \alpha_{t-1}(i)a_{ij}]b_j(O_t) \qquad \text{where, } i \in [1..N] \tag{3.13}$$

Step 3: **(Termination)** Obtain the resultant probability of the observation sequence for the given model λ by the following equation:

$$\Pr(O|\lambda) = \sum_i \alpha_T(i) \tag{3.14}$$

In this algorithm, each of the states at time $t-1$ is completely computed before considering the states at time t. Therefore the last state in the computation constitutes the ultimate probability of generating/producing the observation sequence O, given the model λ. We notice that the computation in the forward algorithm is in the order of $O(N^2T)$.

The backward algorithm computes the probability of the observation sequence for the given λ, in a similar way to that of the forward algorithm. In this case, instead of a forward variable, a backward variable $\beta_t(i)$ is introduced, as follows:

$$\beta_t(i) = \Pr(O_{t+1}, O_{t+2}, ..., O_T|S_{t=i}, \lambda) \tag{3.15}$$

In Eq. (3.15), the backward variable $\beta_t(i)$ accommodates the probability of the partial

observation sequence at time $t+1$ to the time T, considering the state i at time t and the model λ. The calculation of the backward variable is done inductively, as the forward variable and the final probability is obtained by using the following equation:

$$\Pr(O|\lambda) = \sum_i \pi_i b_i(O_1)\beta_1(i) \qquad \text{where, } i \in [1..N] \tag{3.16}$$

As mentioned above, any of the forward or backward algorithms can be used to compute $\Pr(O|\lambda)$ for the evaluation problem. They can also be used together to formulate a solution to the problem of model parameter estimation, as is discussed in the following section.

Re-estimation of HMM parameter values

The most involved problem in HMM is to adjust/re-estimate the model parameters (A, B, π) so that the probability of generating the observation sequence is maximized [2]. No analytical solution for this problem exists for modeling the maximum likelihood criterion. However, researchers have optimized the parameters using an iterative algorithm or gradient technique. The well known iterative algorithm for optimizing the HMM parameters is the Baum-Welch algorithm [149]. An explanation of the Baum-Welch algorithm is given below.

Note that in the forward-backward algorithm we calculated the probability of generating the observation sequence for a given model, λ. There we assumed that the parameter values were known to us. Thus, if the parameter values are known to us, we can evaluate the probabilities produced by the model parameters for a given observation sequence. Once we obtain the probability for the observation sequence, based on this current probability, we can further estimate what the ideal parameter for the model would be [2].

The idea mentioned above can be implemented initially choosing some non-zero parameter values to form an initial model, λ. The probability for a given observation sequence is calculated assuming the initial model, λ which is best suited with the observation. The *a posterior* probability of transitions γ_{ij}, from state i to state j is calculated using the Eq. (3.19).

$$\gamma_t(i,j) = \Pr(S_t = i, S_{t+1} = j|O, \lambda) \tag{3.17}$$

$$= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\Pr(O|\lambda)} \tag{3.18}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_i \alpha_T(i)} \tag{3.19}$$

In Eq. (3.17), the value $\gamma_{i,j}$ refers to the probability of the transition from state i to state j at time t and $t+1$ respectively, for a given λ. In the Eq. (3.18) and (3.19), the variable $\alpha_t(i)$ is the forward variable representing the probability of all paths to state i at time t; $\beta_{t+1}(j)$ is the backward variable representing the probability of all paths at time $t+1$ to the time T.

The value of the variable $\gamma_t(i)$, representing the *a posterior* probability of state i at time t for the observation sequence and the model λ, is calculated using the following equation:

$$\begin{aligned} \gamma_t(i) &= \Pr(S_t = i|O, \lambda) \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_T(i)} \end{aligned} \tag{3.20}$$

The new calculated values of $\gamma_t(i)$, at time $t = 1$ are usually assumed to be the new estimates $\bar{\pi}_i$ of the initial state probabilities π_i. This can be understood by solving the following problem.

For the person and stick experiment described earlier in this section, calculate the probability of choosing the person 1 at time t, given the observation sequence O and the model λ.

The solution for the above problem is calculated using Eq. (3.20). Thus, the probability matrix of choosing all states/persons at time t ($t = 1$) should become the new initial state probability matrix $\bar{\pi}$ as shown in Eq. (3.21).

$$\bar{\pi}_i = \gamma_1(i) \tag{3.21}$$

Now, we calculate the probability of emitting the color *red* (*redstick*) from the person 1 at time t, for the given observation sequence O and model λ. This amount could be obtained by Eq. (3.22):

Emission probability of red stick

$$\begin{aligned} &= \frac{\text{Expected number of selecting person 1 and emitting red stick}}{\text{Expected number of selecting person 1}} \\ &= \frac{\sum_{t \in O_t = red} \gamma_t(1)}{\sum_{t=1}^{T} \gamma_t(1)} \end{aligned} \tag{3.22}$$

Thus the new estimate of the probability of observation symbol v_k from state j would be as in Eq. (3.23).

$$\bar{b}_j(k) = \frac{\sum_{t \in O_t = v_k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{3.23}$$

To obtain the re-estimated value of the probability of the transition from state i to state j, a_{ij}, we calculate the ratio of the expected number of state transitions from state i to state j and the expected number of total transitions from state i. Thus the new estimate of the value of $\bar{a}_{ij}$ is as follows:

$$\begin{aligned} \bar{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \sum_j \gamma_t(i,j)} \\ &= \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned} \tag{3.24}$$

During the re-estimation phase, at some point the re-estimated model $\bar{\lambda}$ might reproduce the same or appear close to the same model λ within a tolerance/acceptance level before the re-estimation took place. If such a thing happens for a number of subsequent iterations the re-estimation process is stopped, assuming that we have found the final HMM. The perfect re-estimation of parameter values happens when the condition $\Pr(O|\bar{\lambda}) \leq \Pr(O|\lambda)$ is satisfied.

After a number of iterations of the re-estimation phase, the final λ is obtained, which is suitable to generate the observation sequence O with a maximum probability. To obtain the re-estimated parameters at each iteration, Eq. (3.20) through (3.24) are used.

Note that the re-estimation algorithm known as the Baum-Welch algorithm has a lot of similarity with the Expectation-Maximization algorithm, as in the Baum-Welch algorithm the expectations of each parameter are maximized. To handle more than one observation sequence (say n number of observation sequences), the re-estimation Eq. (3.24) and (3.23)

are re-written in the following form:

$$\bar{a}_{ij} = \frac{\sum_n \sum_{t=1}^{T_n-1} \gamma_t^n(i,j)}{\sum_n \sum_{t=1}^{T_n-1} \sum_j \gamma_t^n(i,j)} \tag{3.25}$$

$$\bar{b}_k = \frac{\sum_n \sum_{t \in O_t^n = v_k} \gamma_t^n(j)}{\sum_n \sum_{t=1}^{T_n} \gamma_t^n(j)} \tag{3.26}$$

3.2.2 Continuous Hidden Markov Model

In the previous subsection, we have described an HMM dealing with a sequence of discrete symbols. Most of the real world problems are, however, continuous (e.g., speech signal recognition, human movement recognition and stock indices prediction) and hence an HMM that is able to deal with a continuous dataset is required. This can be achieved in a number of ways. Firstly, the continuous dataset can be converted into a number of discrete sets by adopting a quantization technique. In fact, a number of studies that deal specifically with continuous speech data [150] first translate the continuous features into a set of discrete symbols. Another approach is to map the discrete output distribution $b_j(k)$ to the continuous output probability density function. The advantage of doing this, over the quantization technique is that the inherent quantization error can be eliminated [2]. Hence the HMM with a continuous output probability density function is more error prone than the discrete HMM with quantization of the continuous data. In this chapter, the continuous output probability density function HMM is referred to as CDHMM (Continuous Density HMM). The following subsection gives a brief introduction of CDHMM and how it differs from discrete HMM (especially during the re-estimation of parameter values and in calculating the log-likelihood value of a sequence).

Parameter Re-estimation

The CDHMM employs the continuous probability density function. Baum and his coworkers [149, 151] described the generalization of the Baum-Welch algorithm to deal with such a continuous density function. The condition of doing so is that the probability density functions must be strictly log concave, which constrain the choice of continuous probability density function to the Gaussian, Poison or Gamma distributions [2]. A brief summary of re-estimation of the CDHMM parameters is given below [2].

For CDHMM, the continuous observation sequences are represented as: $D \in R^l$ where,

R^l is $l-dimensional$ space of continuous numbers. For a CDHMM, λ_{cdhmm}, we assume that, the initial values of all parameters are known *a priori*. Using equation (3.9), the probability of the observation sequence D (here D represents the continuous observation sequence), given the CDHMM, λ_{cdhmm}, would be:

$$\begin{aligned} f(D|\lambda_{cdhmm}) &= \sum_{\text{all S}} f(D,S|\lambda_{cdhmm}) \\ &= \sum_{\text{all S}} \prod_{t=1}^{T} a_{S_{t-1}S_t} b_{S_t}(D_t) \qquad (D_t \in D, a_{S_0,S_1} = \pi_{S_1}) \end{aligned} \tag{3.27}$$

As mentioned above, the output density functions of the CDHMM are continuous, and in this case, for simplicity, the density functions $b_i(D_t)$ are assumed to have ellipsoidal symmetry as follows:

$$b_i(D_t) = |\sum_i|^{-1/2} f_i(q_i(D_t)) \tag{3.28}$$

Here, $q_i(D_t)$, represents a positive definite quadratic form as in equation (3.29), having that the $l-by-l$ scaling matrices $\sum_i$ (for all states S_i) are positive definite and symmetric. The values of μ_i are chosen randomly from the Euclidean $l-space$.

$$q_i(D_t) = (D_t - \mu_i)^t \sum_i^{-1} (D_t - \mu_i) \qquad (D_t \in D) \quad i \in [1 \cdots N] \tag{3.29}$$

As referenced by Huang [2], and following the results of Liporace [152], to satisfy the condition of elliptically symmetric densities another assumption (Eq. (3.30)) is required, as long as the density function $b(D_t)$ satisfies the consistency conditions of Kolmogorov [153].

$$b(D_t) = \int_0^\infty N(D_t, \mu, u^2 \sum) dG(u) \qquad (D_t \in D) \tag{3.30}$$

where, G = some probability distribution in the range of $[0, \infty]$,

$N()$ = multivariate Gaussian density with mean vector μ and covariance matrix $u^2 \sum$.

The joint probability density function of the state sequence S and observation sequence

D for the model, λ_{cdhmm}, is obtained as:

$$f(D,S|\lambda_{cdhmm}) = f_{\ddot{U}} \prod_{t=1}^{T} a_{S_{t-1}S_t} N(D_t, \mu_{S_t} u_t^2 \sum_{S_t}) dG(u_1)....dG(u_T) \tag{3.31}$$

$$= E_U f(D,S,U|\lambda_{cdhmm}) \tag{3.32}$$

where, $\ddot{U} = [0,\infty]^T$, $U = (u_1, u_2, ..., u_T)^t$, E_U denotes the average with respect to T-fold distribution $G(u_1), ...G(u_T)$ and

$$f(D,S,U|\lambda_{cdhmm}) = \prod_{t=1}^{T} a_{S_{t-1}S_t} N(D_t, \mu_{S_t}, \sum_{S_t}, u_t^2 \sum_{S_t}) \tag{3.33}$$

From Eq. (3.31), (3.33) and (3.27) the density function of D is found as:

$$f(D|\lambda_{cdhmm}) = \sum_{S} \prod_{t=1}^{T} a_{S_{t-1}S_t} N(D_t, \mu_{S_t}, \sum_{S_t}, u_t^2 \sum_{S_t}) \tag{3.34}$$

In a similar manner to the discrete HMM, the re-estimation process of the CDHMM is stopped; i.e., when the re-estimated CDHMM and the CDHMM before re-estimation took place remain the same within an acceptance level for a subsequent number of iterations. In the case of CDHMM, the equation for re-estimating the value of mean $\bar{\mu}_j$ (for the state j at time t), for the continuous observation sequence D (where $Dt \in D$), is obtained as follows:

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)D_t}{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)} \tag{3.35}$$

Here, $\rho_t(j)$ = Gaussian densities at time t for the state j.

Similarly, we obtain the following equation to re-estimate the covariance matrix $\bar{\sum}_j$:

$$\bar{\sum}_j = \frac{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)(D_t - \bar{\mu}_j)(D_t - \bar{\mu}_j)^t}{\sum_{t=1} T\rho_t(j)\beta_t(j)} \tag{3.36}$$

The re-estimation formulas deal with a single (univariate) observation sequence. For the multivariate Gaussian densities, $\rho_t(j) = \alpha_t(j)$, these formulas can be used with a slight modification. The modified re-estimation equations are as follows:

$$\bar{\mu}_j = \frac{\sum_{t=1} T\gamma_t(j)D_t}{\sum_{t=1} T\gamma_t(j)} \tag{3.37}$$

and

$$\bar{\sum}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)(D_t - \bar{\mu}_j)(D_t - \bar{\mu}_j)^t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{3.38}$$

where $\gamma_t(j)$ is the *a posteriori* probability which has the same meaning as used for the discrete HMM.

In the Eq. (3.37) and (3.38), the symbols $\bar{\mu}_j$ and $\bar{\sum}_j$ refer to the mean and corresponding sample covariance matrix of those samples for the state j respectively. The following constraint should follow, such that all of the samples may have at least some role in the re-estimation.

$$0 \leq \gamma_t(j) \leq 1$$

Note that during the calculation of the re-estimated value of the covariance matrix (in Eq. (3.38)) the re-estimated mean $\bar{\mu}$ is used, which might be inconvenient. To avoid this, Eq. (3.38) can be expressed as:

$$\bar{\sum}_j = \frac{\sum_{t=1}^{T} \gamma_t(j) D_t D_t^t}{\sum_{(} t = 1)^T \gamma_t(j)} - \bar{\mu}_j \bar{\mu}_j^t \tag{3.39}$$

An alternative re-estimation equation can also be written as follows:

$$\bar{\sum}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)(D_t - \mu_j)(D_t - \mu_j)^t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{3.40}$$

This equation is obtained based on the heuristics, assuming that the μ are approximately equal to $\bar{\mu}$ in contiguous iterations.

3.3 Application of HMM in time series forecasting

HMM is a ubiquitous method for modeling time series data. HMMs have been used successfully in almost all current speech recognition systems, in numerous applications in computational molecular biology, and in data compression etc. Apart from the time series data formed by the molecular constituents and speech signals, HMM has few applications in other real world time series (e.g. financial data and chaotic non-linear benchmark data) prediction or forecasting the future states of a time series given its current state. Some of the applications of HMM in time series forecasting are as follows.

Andersson *et al.* [28] compared three likelihood-based methods including the HMM for continuously monitoring business cycles, especially for detecting the turning point times for

example, in a change from a recession phase to an expansion phase. They analyzed these using several features, such as the knowledge of the shape and parameters of the curve, the types and probabilities of transitions and smoothing. Each of the three methods was used for the duration of a fixed time period in Swedish industrial production, and would set off an alarm before the turn, largely due to the plateau just before the transition.

Kohlmorgen *et al.* [141] have presented a framework for the unsupervised online segmentation and identification of switching dynamics. In this study, each state of an HMM is represented as a prediction expert. These experts are trained by Expectation Maximization (EM) and by using a deterministic annealing schedule for the state transition probabilities. To predict which mode/status is present at time t, the latest available information is taken into account for each of the experts in the HMM. It has been demonstrated for the Mackey-Glass time series data [62] that mode changes can be detected much earlier by taking into consideration the latest available input data.

Papageorgiou [154] presented a Markov model based method to analyze coupled time series. The emphasis in the study is on reducing the burden of estimating the large number of parameters in the Markov models. While modeling the coupled time series, the state space is defined as the Cartesian product of the smaller state space (as is done in factorial Markov models). To reduce the large parameter values of the transition matrix (due to the Cartesian product the size of the transition matrix becomes larger), each of the underlying components of the elementary transition matrices are combined as a convex combination. The resultant combinations are used to represent the final transition matrix. A similar approach to representing a transition matrix is found in [155], which is known as the mixed Markov model. The proposed model was tested on the currency exchange rates of the British Pound, Canadian Dollar, Deutsch Mark, Japanese Yen, and Swiss Franc against the US Dollar from 06/01/73 to 05/21/87.

Viovy and Saint [156] analyzed the dynamics of vegetation represented by remotely sensed data using HMM. Here, the observable quantities are the canopy and its temporal evolution. As the plant changes continuously from state to state during its whole life, it is assumed that each state is dependent on past states and also that there are some external influences, e.g. drought, harvest etc. Thus the states are hidden; that is, they are not directly observable. The HMM has been applied in the forecasting of some phenological stages for the savanna regions in West Africa. The data used here is from a six year sequence from the weekly global vegetation index from January 1983 to December 1988.

Shi and Weigend [157] used the continuous HMM for forecasting the daily S&P500 stock data and currency exchange rate. In the study, each of the states of the HMM are supposed to represent a specific region in the time series data. Within the individual regions, the time series is assumed to satisfy the requirement of stationarity. The trained HMM is used to predict the entire probability of return for the considered time series data. To represent the emission probability density of each of the states, traditional Gaussian distribution is chosen. However, in an extended version of this study Weigen and Shi [29] predicted the daily probability distributions of S&P500 returns using a modified continuous HMM called "Hidden Markov Expert". They addressed the problem of a small signal to noise ratio in the financial time series and focused particularly on non-Gaussian density forecasts. The density prediction is calculated using the following summation of M distributions:

$$\begin{aligned} &\Pr(Y^{t+1}|\text{information set at time } t\text{, model parameters}) \\ &\quad = \sum_{j=1}^{M}(.)\Pr(y^{t+1}|x^{t+1},\ldots,\text{model parameters}) \end{aligned} \tag{3.41}$$

where, $\Pr(y^{t+1})$= The predicted density at time $t+1$ and γ_j^{t+1}= weight given to Gaussian j for the prediction for time $t+1$ with the constraint $\sum_{j=1}^{M}\gamma_j = 1$. Several discrete states have been considered, each of which corresponds to the functional input-output mapping expressed by feed forward NN. These sub-models are called as experts. It is assumed that at each time step a single expert is responsible for generating the corresponding observation. The sequence of the hidden states is described by a first-order Markov process, as is assumed in HMM. The model was tested to predict the density of the daily S&P500 and achieved better performance compared with that of GARCH(1,1).

In a related work Zhang [158] has used the Gaussian mixtures instead of the NN to be the prediction expert for the specific region attached to the state in the continuous HMM. A prediction pool has been introduced in the prediction model. In this pool, the last available data is added to take account of the change in the time series and the first data in the pool is removed. Weights are imposed on each of the data in the pool to put more emphasize on the current data than the data those are distant from current one. The prediction model is used to predict the movement in the financial time series of the S&P500 and Dow Jones. The prediction results using the model clearly outperform the model proposed by Shi and Weigend [157].

Limitations

HMMs have been used in a few studies to forecast time series data. However, in most of these studies the states of the HMM are used as an *expert* to predict a specific status in the time series, e.g. high or low, expansion or recession, etc. [28]. In some of the studies [29], the time series is assumed to be composed of a number of regions: each of the regions having different mean and standard deviation. Each of the states in the HMM would represent an individual region and using the HMM the density of the time series is predicted [29]. A probability density function for a non-linear and non-stationary time series data is useful on identifying the characteristics of the time series, though it cannot help in predicting the exact future value of the time series. Zhang [158], in his Masters thesis, used continuous HMM with a mixture of Gaussian distribution as the emission probability function for each of the states considered. He predicted the exact value by calculating the means of the mixture of Gaussian distribution functions for the state that was predicted as the most probable next state. This approach to prediction is somewhat ad-hoc and its performance would depend on a suitable choice of the number of mixture distributions. Furthermore, the performance would depend on the optimal HMM structure.

It is interesting to note that none of the studies that used the HMM in time series forecasting considered multiple time series data, which could have an influence on the time series data to be predicted.

3.4 Summary

In this chapter, we have introduced HMMs and the algorithms necessary to build an HMM for a given observation sequence. We described both the discrete HMM and continuous HMM in detail with the re-estimation processes. We reviewed some of the studies that have used HMM for forecasting time series data. In the typical application of HMM, the best matching state sequence is determined assuming that the current state is known. Then the next probable state of emitting the next observation is predicted using the emission probability matrix, state transition probability matrix, and the prior probability matrix. What differs in our research from this typical approach is that we use HMM for forecasting exact values instead of trends (up, down, steady etc.) that are represented by the states. We use multivariate dataset to forecast the time series data. Initially we use the HMM

for identifying similar patterns and then adopting some means to forecast the exact future value in the time series data. In the next chapter we explore the use of a single HMM for clustering similar data patterns.

Part II

HMM-based forecasting methodologies

Chapter 4

Clustering multivariate data using a single HMM

In this chapter, we propose a new approach to identifying similar data patterns and grouping them into clusters using a single CDHMM. The identification of similar data patterns in the multivariate time series data is used to forecast time series throughout the thesis. We cluster similar objects into a group to expedite our search. Therefore this chapter describes how CDHMM can be used in identifying similar data patterns; and evaluates its efficiency by comparing the results with other existing techniques. From this point onwards we shall refer to the CDHMM as HMM, as we have made use of CDHMM throughout the thesis.

Clustering

Clustering is a process where, given a collection of unlabeled patterns (a dataset), the data items are divided into groups (clusters) based on some measure of similarity [159]. A variety of clustering techniques have been proposed in the machine learning, pattern recognition, data mining and statistics domains. Well known examples include k-means [160], fuzzy c-means [133, 134], Self-Organizing Map (SOM) [161, 162], ANNs [163] and Support Vector Machines [164]. These techniques belong to one of two groups – *supervised clustering* and *unsupervised clustering* – depending upon the underlying method used to cluster the data items.

In supervised clustering the algorithm is trained using a proportion of the dataset (the training set) and then this trained algorithm is used to classify an unknown dataset (test set). Typically, the number of clusters for supervised learning is pre-specified. Alterna-

tively, in unsupervised clustering a training dataset is not used. Here, assumptions have to be made about the number of clusters to be used and/or the spread of each cluster prior to clustering the data. The the algorithm partitions the data into clusters. For instance, to cluster the dataset using k-means, fuzzy c-means and hierarchical clustering [165], the number of clusters must be known *a priori*.

Typically, for pattern classification/identification, a number of HMMs are used in combination with supervised techniques. In this chapter we propose and develop a single HMM based unsupervised clustering algorithm. In our model a single HMM is used to identify the number of clusters in a given dataset by segmenting the log-likelihood values of each of the data patterns into "windows", which we term as "buckets" or "bins". The data items are then labeled and partitioned into the appropriate clusters. The findings of this chapter have been published in [166].

The remainder of the chapter is organized as follows. In Section 4.1 we describe the HMM-based data clustering algorithm in details. Section 4.2 presents the experimental results and, finally, the summary of the chapter is provided in Section 4.3.

4.1 Clustering a dataset using a single HMM

In our proposed model, a single HMM is used to identify the number of clusters in a given dataset. The data items are then labeled and partitioned into the appropriate clusters. Initially, the HMM is used to calculate log-likelihood values for each of the data items. Note that, here, the log-likelihood values on one hand represent how well the data fits the trained HMM, and on the other provide a similarity measure between data items. Based on these log-likelihood values the dataset is then partitioned into windows of fixed size. The plot of the log-likelihood values after segmenting into "windows" is used to identify the possible number of clusters in the dataset. If the difference in the sorted log-likelihood values exceeds some threshold, a new cluster is formed. Once the number of clusters has been determined, the sorted log-likelihood values are used to divide the data items into appropriate clusters. It should be noted that the key feature of the model is that, for a given dataset, we do not specify *a priori* the number of clusters. The model is deemed capable of identifying appropriate partitions in the dataset and divides the data items into appropriate clusters.

There are three steps in implementing the HMM based clustering model. The first

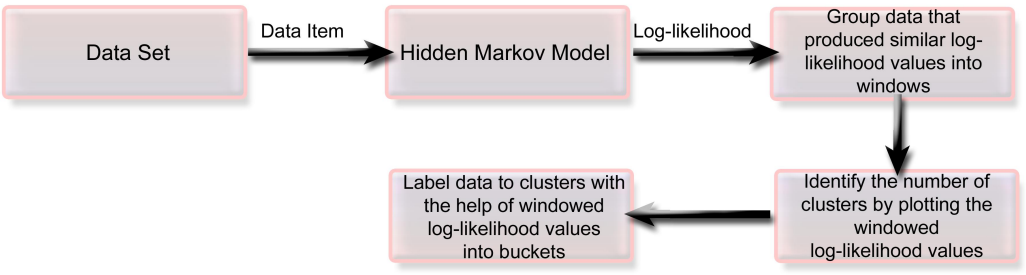


Figure 4.1: The HMM based model to cluster multidimensional dataset

step consists of training the HMM and producing log-likelihood values of the data items. The second step is sorting the log-likelihood values of the dataset according to the defined "windows" or "bins". This process helps in identifying the number of clusters. In the third and final step, labeling of data items to their respective clusters is carried out. Figure 4.1 illustrates the basic functionality of the model. The specific details of the model follow.

4.1.1 Generating likelihood values using HMM

Initially, a single HMM is built by choosing an arbitrary architecture for the dataset to be grouped into clusters. As a rule of thumb, the number of states is chosen to be equal to the number of variables in each of the data items. For instance, let us assume the input data space is X and each element of X is x_m; i.e., $x_m \in X$ where, $x_m = m^{\text{th}}$ vector and $x_{im} \in x_m$, i.e., x_{im} is the i^{th} element of the m^{th} vector. The data vectors are shown in Fig. 4.2. For this dataset we chose a k state HMM, because there are k instances in each vector. For the continuous dataset in Fig. 4.2, the continuous patterns formed are shown in Fig. 4.3.

To construct an initial HMM we must choose the initial parameter values of A, B and π. For the parameter values of A, initially we choose randomly generated numbers

$$\begin{array}{cccc|c} x_{11} & x_{21} & \dots & x_{k1} & DataVector1 \\ x_{12} & x_{22} & \dots & x_{k2} & DataVector2 \\ x_{13} & x_{23} & \dots & x_{k3} & DataVector3 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ x_{1m} & x_{2m} & \dots & x_{km} & DataVectorm \end{array}$$

Figure 4.2: Data vectors in the dataset

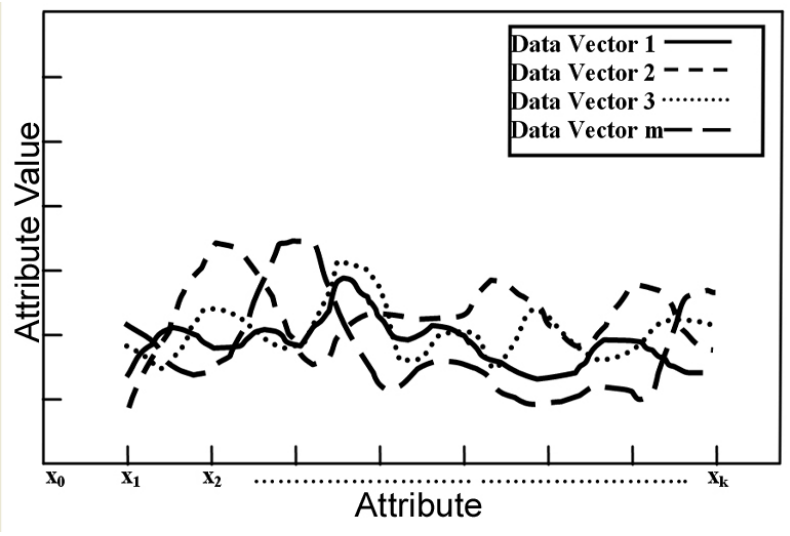


Figure 4.3: The continuous signal/pattern formed by the data vectors (see Fig. 4.2)

for the transition probabilities a_{ij}, such that, $\sum_i a_{ij} = 1$ for all j. Similarly, we choose the initial values of π such that, $\sum_i^N \pi_i = 1$. This is achieved by choosing a randomly generated number and then normalizing it. To obtain initial values for the observation emission probability matrix B we make use of the *segmental k-means algorithm* [146]. A description of this procedure follows.

Initially, we choose a random N group of data vectors (each data vector has the dimension k) out of a total of m data vectors and assign each of the N groups of data to one of the states in the HMM [146]. Thus, for the k dimensional data vectors of each of the group we obtain k number of means and covariance matrices. For the N groups of data there are a total of $N * k$ means and covariance matrices. Note that this initial choice of grouped/clustered vectors certainly does not create the final HMM, but it helps us to begin the re-estimation process using the given datasets. In our case, the total number of states is chosen as the dimension of the data vectors. Thus, the number of states N would be equal to k. The dataset is divided into k subgroups and corresponding mean, and covariance values for each of the attributes in each of the subgroups are calculated. The mean vector μ_i and the covariance matrix $\hat{V}_i$ for each of the states i are calculated using the following equations:

$$\hat{\mu}_i = \frac{1}{k_i} \sum_{D_t \in i} D_t \tag{4.1}$$

$$\hat{V}_i = \frac{1}{k_i} \sum_{D_t \in i} (D_t - \hat{\mu}_i)^T (D_t - \hat{\mu}_i) \tag{4.2}$$

Here, D = Continuous observation/data sequence and $(D_t \in D)$

These means and covariance values are used to obtain the initial observation emission probability distributions for each of the training data vectors for each state as follows [146]:

$$\hat{b}_i(D_t) = \frac{1}{(2\pi)^{k/2}|\hat{V}|^{1/2}} exp[-\frac{1}{2}(D_t - \hat{\mu}_i)\hat{V}_i^{-1}(D_t - \hat{\mu}_i)^T] \tag{4.3}$$

where, $1 \leq i \leq k$.

The Baum-Welch algorithm described in the previous chapter is used to re-estimate the initially chosen parameter values of the HMM for the given training dataset. The re-estimated parameter values determine the final HMM. The likelihood function is represented by $\Pr(D|\lambda_{cdhmm})$, where D represents the continuous observation sequence and λ_{cdhmm} is the final HMM. The next step is to calculate the likelihood values.

To calculate the likelihood values for each of the data vectors x_i, where,$1 \leq i \leq m$, given the trained HMM, the forward algorithm described in Section 3.2.1 is used. For the m data vectors (each is k dimensional) we obtain m distinct likelihood values. The likelihood or logarithm of likelihood values (known as log-likelihood values) may be interpreted as follows. Consider two k-dimensional data vectors $D_w = \langle x_{1w}, x_{2w}, ..., x_{kw} \rangle$ and $D_r = \langle x_{1r}, x_{2r}, ..., x_{kr} \rangle$, where, $x_{ij} = i^{\text{th}}$ attribute of the j^{th} data vector in the dataset. If the log-likelihood values l_w and l_r corresponding to the data items D_w and D_r are close, then we may infer that the data items D_w and D_r should belong to the same cluster [166].

4.1.2 Number of Clusters in the Dataset

The Baum-Welch algorithm maximizes the probability that each data vector fits the model. The log-likelihood values for two data items represent the similarity between them [2,147]. As in any datasets, typically, there are two or more clusters, it is reasonable to expect that *close log-likelihood* values will gather into one group, while the remaining log-likelihood values will appear at a distance from this group.

In order to determine the number of clusters in the dataset, we need to process the data based on their corresponding log-likelihood values. Figure 4.4 displays the log-likelihood values along the vertical axis and the respective data labels (horizontal axis) for a sample

dataset. The data label refers to the record number (i.e., if x_m represent the m^{th} data vector, m would be called as the record number/index) of the data item in the dataset. As it stands, this plot does not provide any useful information. We, therefore, group the log-likelihood values into several bins (windows) in order of magnitude ranging from minimum to maximum. These bin frequencies for the benchmark Ringnorm data are displayed in Fig. 4.5. The dataset has been described in details in Section 4.2. A pseudo code for distributing the log-likelihood values into bins is given in Fig. 4.8.

Figure 4.6 and 4.7 display the log-likelihood values for another two classical data sets Iris data and Thyroid. An inspection of the plotted bin frequencies in Fig. 4.5 or the log-likelihood values in Fig. 4.6 and 4.7 suggests that there are distinct breaks or changes in the curves, which lead us to think of dividing lines for possible clusters. Despite the subjective nature of such an inspection and placement of segmentation line(s), when a change in log-likelihood values exceed some threshold (e.g., an inflection point) we can identify the resulting partition. Further, we notice in the lower part of Fig. 4.7 there are many gaps in the curve that may suggest a number of possible clusters. However, since the number of points in this region is not significant, we may combine all points into one cluster and consider a total of three clusters dividing the data.

4.1.3 Labeling Data for Clusters

In Section 4.1.2, the log-likelihood values were initially partitioned into a window of fixed size (width 1 in this case). The next task is to label the data vectors (items) into clusters. To do this we examine the frequency of data vectors in each of the windows. Initially we use a minimum threshold frequency ψ as a means of identifying the natural partitions in the dataset. Then we merge windows on either side of the partition into a variable-sized window representing a cluster. The aim is to find the minimum number of windows (variable size) such that all the similar data vectors fall into their respective bins.

Initially we consider that the empty window/bin among other bins indicates the edges of the cluster. If there is no empty window, a threshold value ψ (minimum frequency) is subtracted from the frequencies of all windows to generate some empty windows. When the number of empty windows is more than the number of clusters identified in Section 4.1.2, some of the windows (empty as well as non-empty) in close neighborhood are merged. The pseudo code for labeling data to clusters is given in Fig. 4.9 and the experimental results from using this procedure are detailed in Section 4.2.

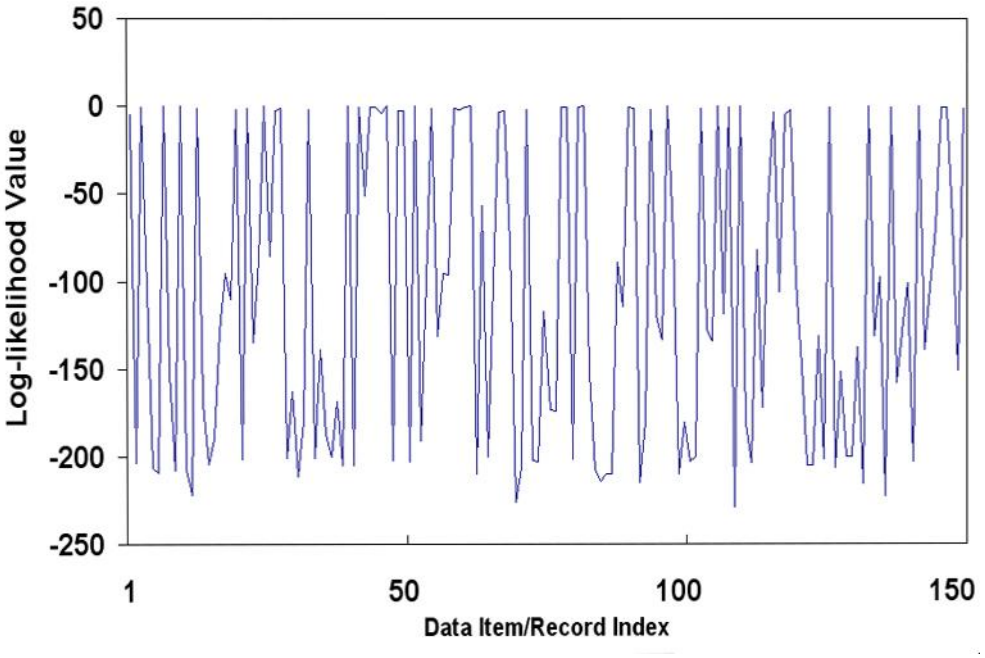


Figure 4.4: Log-likelihood values of data items/patterns

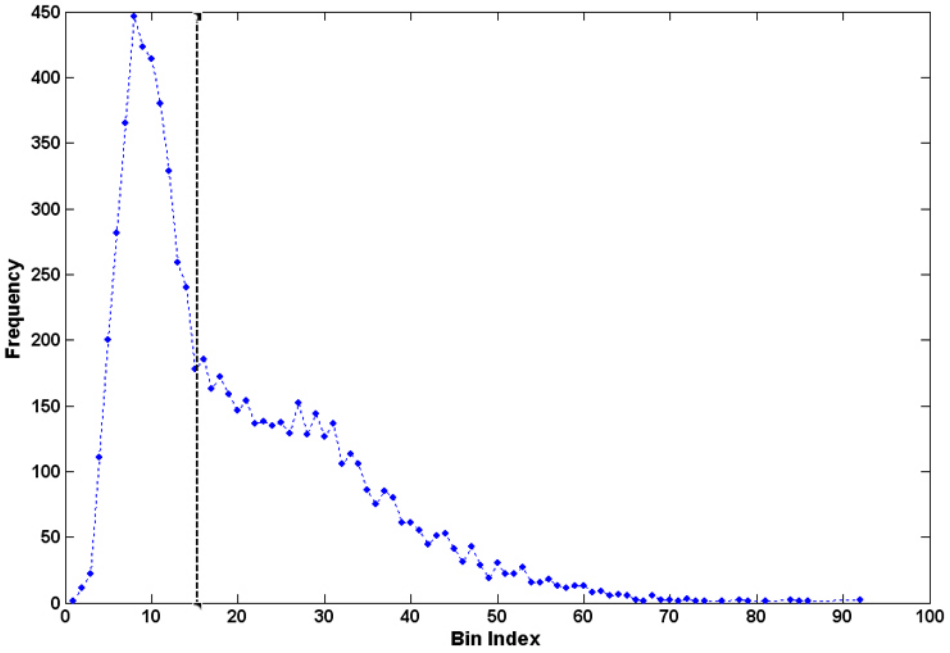


Figure 4.5: Ringnorm bin frequency chart

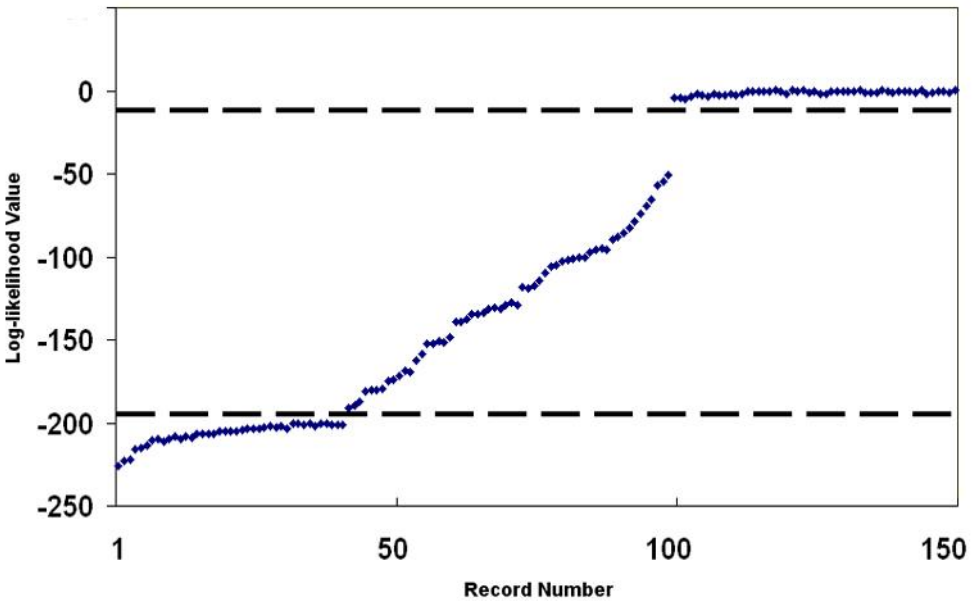


Figure 4.6: Number of clusters (Iris Data)

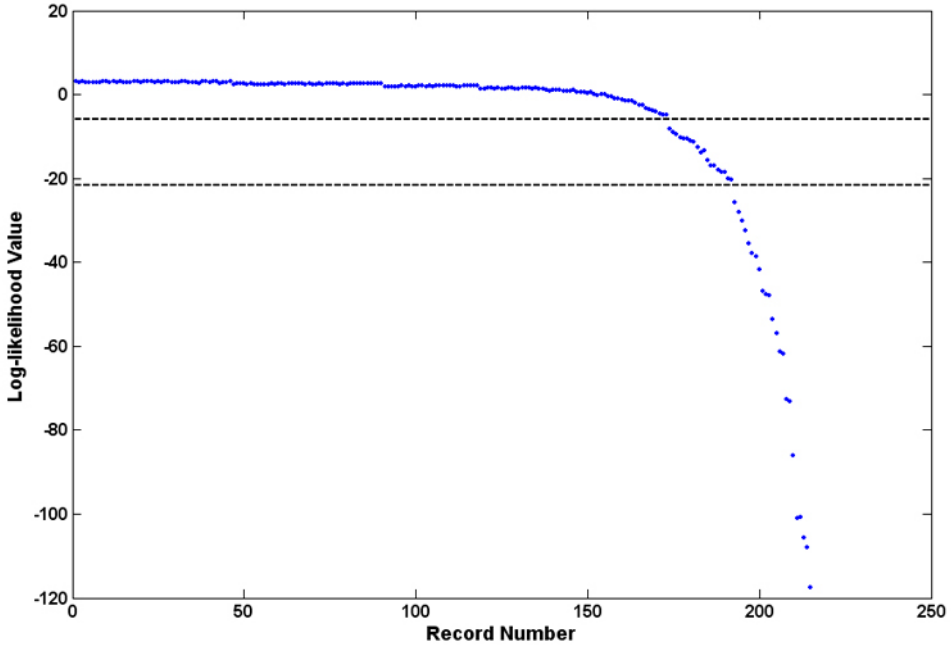


Figure 4.7: Number of clusters (Thyroid Data)

1. Get LL values for each of the data item in the dataset;
2. Set: window_size=θ;
3. Set: minimum_LL=minimum LL value;
4. Set: maximum_LL=maximum LL value;
5. Set: window_ptr=minimum_LL and i=1;
6. Repeat steps until window_ptr¡ maximum_LL;
7. Set: window(i)=data items which produces LL value in between window_ptr and window_ptr+window_size;
8. Set: window_ptr=window_ptr+window_size;
9. Increase i by 1;

Figure 4.8: Pseudo code for grouping similar data items

1. Set: threshold_frequency=ψ
2. Repeat step 3 for i=1 to total_ number_ of_ window
3. Subtract ψ from the window(i)
4. Repeat step 5 and 6 until last_window is reached
5. Locate the empty windows and combine subsequent empty windows (if any)
6. Combine subsequent occupied windows
7. (a) Find the maximum gap between two occupied windows and point it as a current_cluster edge
   (b) Locate the occupied windows from 1st window to the current_ cluster edge as the first cluster
   (c) Collect data items in the windows pointed as cluster in step
   (d) Label these data item to the cluster
   (e) Set: previous_cluster edge=current_cluster edge;
8. Repeat step 9 until the desired number of clusters is found or the last window is reached
9. (a) Find the next maximum window gap and point it as current_cluster edge
   (b) Locate the occupied windows from previous cluster_edge to the cluster edge as the next cluster
   (c) Collect data items in the windows pointed as cluster in step
   (d) Label these data item to the cluster
   (e) Set: previous_cluster edge=current_cluster edge;

Figure 4.9: Pseudo code describing the steps for labeling and allocating data to clusters

4.2 Experiment and Result

4.2.1 Performance evaluation of HMM based data clustering

Dataset

To evaluate the performance of the HMM based clustering described in the Section 4.1, we consider a number of benchmark datasets which include: Fisher's Iris dataset [167], Thyroid dataset [168], Ringnorm dataset [169] and the KDD 1999 Intrusion Detection System (IDS) dataset [170]. Details about these datasets are given below.

Fisher's Iris dataset [167] consists of sepal and petal measurements. The dataset consists of 50 objects from each of the three species, Iris-virginica, Iris-versicolor, Iris-setosa. Each object in the dataset is described by four attributes: sepal length, sepal width, petal length, petal width i.e., the data item is four-dimensional.

The Thyroid dataset [168] consists of a complete medical record, including anamnesis, scans etc., to help diagnosis whether a patient's thyroid belongs to the class euthyroidism, hypothyroidism or hyperthyroidism, and was first used by Coomans [171]. There are three classes in the dataset: normal, hyperthyroid; and hypothyroid and five attributes: T3-resin uptake test, total serum thyroxin, total serum triodothyronine, basal thyroid-stimulating hormone (TSH) and maximal absolute difference of TSH value followed by some condition. The dataset consists of 215 samples: 150 in the normal class, 35 in hyper class and the remaining 30 in the hypo class. In our model we have considered variables with numerical values only.

The Ringnorm dataset is a classic benchmark dataset, which is an implementation of Lei Breiman's ringnorm example [169]. The dataset contains 20 attributes and 2 classes. The attributes of each class are found from a multivariate normal distribution, while the classes are made to be different to each other by differing the mean and covariance matrix. Out of 7400 samples, 3736 samples belong to class 1 and the rest of the dataset belong to class 2.

The KDD (1999) dataset [170] is used to classify and detect attack/normal network behavior. In the dataset there are 41 attributes, consisting of 34 continuous and the remaining 7 are categorical information. The data are labeled as normal (non-attack) and the rest of the data are classified into other classes as attacks. We used 11 attributes from a total of 41. The chosen 11 continuous attributes were found to contain significantly larger values compared to other continuous attributes. We did not consider the categorical at-

Table 4.1: Misclassification percentage for some benchmark data

Dataset	No. of Predictors	No. of clusters	Misclassification percentage			
			HMM	SOM	k-means	Fuzzy k-means
Iris Data	4	3	5.33	8.667	15.33	10.67
Thyroid	5	3	12.09	11.1628	22.9	26.3
Ringnorm	20	2	13.43	21.4189	35.878	23.80
KDD 1999	11	2	1.05	1.24	1.27	1.34

tributes at this stage, as most of the algorithms used in our proposed model can handle only numerical values. The 11 attributes considered are: duration, src_bytes, dst_bytes, count, srv_count, same_srv_count, dst_host_count, dst_host_srv_count, dst_host_same_srv_count, dest_host_same_src_port_rate, dst_host_srv_rerror_rate. The dataset has 4,898,431 records, where 3,925,651 (80.1%) records are labeled as attacks and the rest of the data is labeled as normal/non-attack. However, to evaluate the proposed method we consider a small part of this dataset consisting of 26,829 data. In this dataset, there are 25,620 labeled as attacks.

Results

The results obtained for each of the benchmark datasets were compared with three other well-known data clustering algorithms: k-means [160], SOM (supervised) [161, 162], and fuzzy c-means [133, 134]. Table 4.1 displays the performance results (in terms of misclassification rate) for each dataset− model combination. For the k-means and fuzzy c-means algorithms, we supplied the relevant number of clusters for each dataset. For the k-means algorithm a 10 iteration was chosen to stop the clustering program. For the Iris data set, the plot in Fig. 4.6 was used to identify the number of clusters to be used.

4.3 Summary

In this chapter, a data clustering algorithm based on a single HMM has been proposed to: (i) identify the number of clusters in a dataset (both visually and algorithmically); and (ii) label the data item to its respective clusters. Our model is implemented in three steps. First, using a single HMM, the log-likelihood values are calculated for a given dataset. Second, similar log-likelihood values are grouped into windows of fixed size. The windows

are then visually scanned to identify logical partitions in the dataset. Finally, labeling of data items to their respective clusters is performed.

Our model provides a means of identifying possible clusters in a given dataset. The plotting of log-likelihood values allows for the identification of partitions in the dataset, thereby providing the necessary information to label the data items to their respective clusters. A key component of the model is the examination of similarity measures between data items based on their log-likelihood values. It is the differences in log-likelihood values that eventually produce clusters of different sizes. The results reported in Tab. 4.1, clearly establish that our model outperforms other well-known data clustering models for the benchmark datasets considered.

An important assumption on which the HMM is based is that there is some relationship (dependency between subsequent attributes) between the attributes of particular data items in the dataset considered. This in turn provides a measure of similarity (log-likelihood values) between the data items. This is in contrast to most of the existing clustering methods, which assume that each attribute of the data items is independent of each other. The HMM based method can clearly identify the possible number of clusters in the dataset if there is a little or no overlap between two adjacent clusters. This strength, in identifying cluster numbers from scattered data, clearly differentiates the proposed method from most of the existing clustering techniques.

Despite the fact that our model has outperformed other data clustering models for the given datasets, we have identified that the performance of the HMM-based technique may be degraded (in terms of misclassification percentage) when the clusters are overlapping (that is, where it is difficult to clearly identify cluster edges). To address this problem, further sensitivity analysis is required in terms of the threshold frequency levels used in the fixed variable sized windows. Having said this, it is worth mentioning that the problem of identifying exact partitioning is an ongoing research challenge for most of the clustering methods.

In the next chapter we will attempt to use the property of sorting the data patterns based on HMM-loglikelihood values for forecasting financial time series data. This approach attempts to overcome the subjective issue of finding the cluster edges and can be used to obtain a better forecast, given the historical dataset.

Chapter 5

HMM-based financial time series forecasting

In this chapter, we extend the HMM-based data pattern identification approach discussed in the previous chapter, to generate a forecast for financial time series data. The motivation for this approach is based on the strong statistical foundation on which HMMs are based, and the corresponding computational efficiency will provide a framework for developing a robust forecasting method for financial time series. Furthermore, to avoid choosing the initial value for the HMM randomly, we have incorporated an evolutionary search algorithm GA, along with a non-linear transformation function ANN. Details of the above mentioned studies have been published in the conference proceedings [172] and the journal [173].

Financial Time Series Forecasting

Forecasting stock prices or financial markets has been one of the biggest challenges to the CI community. By its nature, the stock market is mostly complex (non-linear) and volatile. The rate of price fluctuations in such series depends on many factors; for example, equity, interest rate, securities, options, warrants, merger and ownership of large financial corporations or companies, etc. Human traders cannot consistently win in such markets. Therefore, developing CI systems for this kind of forecasting requires an iterative process of knowledge discovery and system improvement through data mining, knowledge engineering, theoretical and data-driven modeling, as well as trial and error experimentation.

HMMs have been used in analyzing and predicting time series phenomena. HMMs have been extensively used in areas like speech recognition [2, 174, 175], DNA sequencing [3],

and ECG analysis [176]. Somewhat surprisingly, their use in predicting stock market time series has been limited. One study by Shi and Weigend [157] introduced an HMM to predict changes in the trajectories of financial market time series data. In related work, Zhang [158] modified the HMM training scheme to improve the prediction accuracy of Shi and Weigend's model.

In this chapter, we use a single HMM in a novel way to forecast "one-day ahead" stock prices. Then we propose and develop a fusion model, by combining the HMM with an ANN and a GA to improve the quality of the forecasts. Here, the ANN is used to transform the input observation sequence of the HMM and the GA is used to optimize the initial parameters of the HMM. This optimized HMM is then used to identify similar data patterns from the historical data. Two alternative methods are examined for constructing the "one-day ahead" forecast value based on the HMM identified patterns. In the first approach, we interpolate the neighbouring values of the identified data patterns to calculate the forecast value. In the second approach, we locate a number of similar data patterns from the historical data and then use a weighted average of the difference values between neighbouring data items. This value is then added to the current value of the variable of interest to obtain the corresponding forecast value.

The remainder of the chapter is organized as follows. Details of the proposed method are provided in Section 5.1 and 5.2. Section 5.3 lists the experimental results obtained using the model. Finally, Section 5.4 summarizes the chapter.

5.1 Using HMM for stock market forecasting

In this section, we develop an initial HMM-based technique for stock market forecasting. We forecast the exact future value in the stock data, as opposed to simply identifying the upward or downward trend. There are three steps in this technique. Initially, an HMM structure is built and the HMM parameter values are re-estimated using a training dataset. HMM-loglikelihood values are obtained for each of the data items in the dataset using the trained HMM. Next, we locate the past day(s) stock behavior that is similar to that of the current day using the obtained loglikelihood value. Then the difference values of the variable of interest between the matched day and that of the next to the matched day are calculated. This difference value is added with the value of the variable of interest of the current day. This value is the forecast value for next to current day. Figure 5.1 illustrates

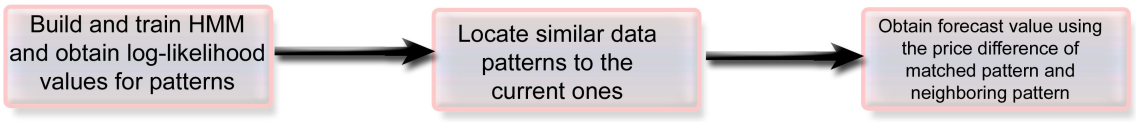


Figure 5.1: The steps for forecasting using HMM

the steps in our model. It should be noted that the observation sequence consists of daily stock prices – *open*, *high*, *low* and *close*: i.e., each data pattern in the dataset comprises of the daily *open*, *high*, *low* and *closing* stock prices. These features are used as predictors for the next day's stock price.

5.1.1 Build and train HMM

Build the Initial HMM

To simplify the implementation, we have used an HMM with 4 states to model the observation sequences/patterns with 4 input features – the opening price, closing price, highest price, and the lowest price. The next day's closing price is taken as the target price associated with the four input features.

For the prior probability π_i, a random number was chosen and normalized so that $\sum_{i=1}^{N} \pi_i = 1$. Given that the dataset is continuous, following the methodology in chapter 3, a Gaussian distribution was initially chosen as the observation emission probability density function. Each of the states in the HMM represents some independent part of the training data patterns. To select the subset of observation patterns that is to be attached to a specific state, we adopt the means used in *k-means segmental algorithm* [146], where the total data items/patterns are divided into 4 groups randomly (because here we choose 4-state HMM). Then each of the groups are attached with individual states in the HMM, as in the previous chapter. Thus the observation emission probability distribution for each of the training data items/patterns for each of the states is calculated using Eq. (4.3).

Parameter re-estimation/Train the HMM

The initial values of the parameters of the HMM are re-estimated using the Baum-Welch expectation maximization algorithm [149] and the training dataset. Details about this algorithm are described in Chapter 3. The approach of choosing the initial HMM structure along with the initial parameter values are discussed in the previous chapter.

Input: C = Current day's data pattern

Output: Day_m = Last time the day has a similar pattern

Process:

Calculate log-likelihood values L_c for C using trained HMM.
Search from previous set of log-likelihood values $\{L\} \epsilon L_c \approx L_m$ to find Day_m, where L_m = log-likelihood value of m^{th} data pattern from past dataset which matches with L_c.

Figure 5.2: Pseudo code to locate a similar data pattern to that of the current one

Obtain log-likelihood values

Using the trained HMM, log-likelihood values for each of the data patterns in the dataset (including training and test dataset) are calculated. For a total, M_{total} data pattern in the dataset, M_{total} log-likelihood values are obtained. For calculating log-likelihood values for each of the data item/patterns D_i $(1 \leq i \leq M_{total})$ of M_{total} data patterns, the forward algorithm described in Chapter 3 is used.

5.1.2 Locate similar data pattern

To obtain the forecast value of the next day's closing price, we locate the past day(s) stock pattern/behavior, which is similar to that of the current day c using the obtained log-likelihood value for the data pattern/sequence containing the daily $open, high, low and close$ price. To clarify, let us assume the log-likelihood value for the observation pattern on day c is L_c. Now from the total M_{total} log-likelihood value we locate the log-likelihood value L_m which is close (i.e., the values of L_c and L_m are the same within a tolerance level) to that of L_c. Note that we find the matched data pattern from the historical dataset that produces the log-likelihood value L_m. A pseudo code to locate similar data pattern in that of the current day's stock behaviour is shown in Fig. 5.2.

For example, an HMM was trained using the daily stock data of an airline company for the period 18 December 2002 to 29 September 2004 to predict the closing price on 30 September 2004. The trained HMM produced log-likelihood value -9.4594 for the stock price on the current day at 30 September 2004. Using this trained HMM and the past data, we located a (closer) likelihood value -9.4555 on 01 July 2003. Figure 5.3 shows the similarities between these two data patterns/vectors (stock prices on 30 September 2004 and 01 July 2003) and Tab. 5.1 reports the exact values of each of the features in the

Table 5.1: Current day's data vector and past data vector matched with that of current day for the daily stock data of an Airline company

	Opening price	High price	Low price	Closing price	Predicted closing price (30 Sep 2004)	Actual closing price (30 Sep 2004)
Today's data 29 Sep 2004	\$13.63	\$13.73	\$13.49	\$13.62		
Matched data pattern using HMM 01 Jul 2003	\$17.1	\$17.2	\$16.83	\$17.13	\$13.85	\$13.85
Next day's data 02 Jul 2003				\$17.36		

respective data vectors.

5.1.3 Obtain forecast value

To obtain the forecast value of the variable of interest, given the past historical data along with the data pattern on the current day, we have located the m^{th} day's observation sequence/patterns which produced the same log-likelihood value L_c. The price difference, $diff$, of the variable of interest (closing price in this case), on day m and $m+1$ is calculated. Using Eq. (5.1), the forecast value of the variable of interest on day $c+1$ is obtained.

$$\text{Forecast value of the } \textit{interestedvariable} \text{ on day } c+1 = \text{The value } (\textit{of the interested variable}) \text{ on day } c + diff \tag{5.1}$$

For the above mentioned example, the calculated difference between the closing prices on 01 July 2003 and the next day 02 July 2003 is \$17.36 − \$17.13 = \$0.23. This difference is added to the closing price on 29 September 2004 to obtain the forecast closing price for 30 September 2004. Table 5.1 shows the predicted and the actual prices of the stock on 30 September 2004.

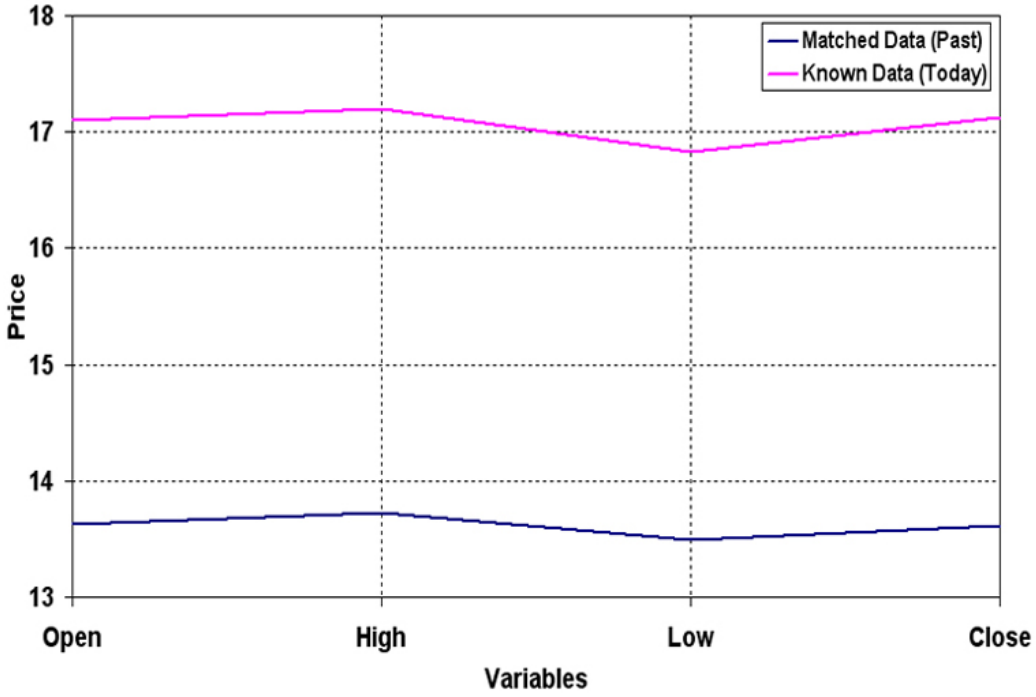


Figure 5.3: The graphical representation of the two data patterns on the current day's stock data and stock data pattern from the past that matched the current day's pattern

In the subsequent sections, we refer to this initial HMM based forecast method as HiMMI (Hidden Markov Model followed by Interpolation).

Challenge

In the HiMMI, we have generated an HMM structure by randomly choosing all of its parameter values. To model time series data with a single HMM alone might not produce good performance. This may be attributed to poor parameter initialization of the HMM and consequently poor training of the HMM. In addition, correlations between the training observation sequence values might lead to degraded performance. In addition, the simple interpolation mechanism relies on a single observation sequence. The challenge, therefore, is to develop a more robust model and improve the forecast accuracy.

5.2 The fusion model of GA, ANN and HMM

It is well-known that the performance of the HMM depends on the initial values of the HMM parameters [177–179] and the input observation sequences. Therefore, to meet

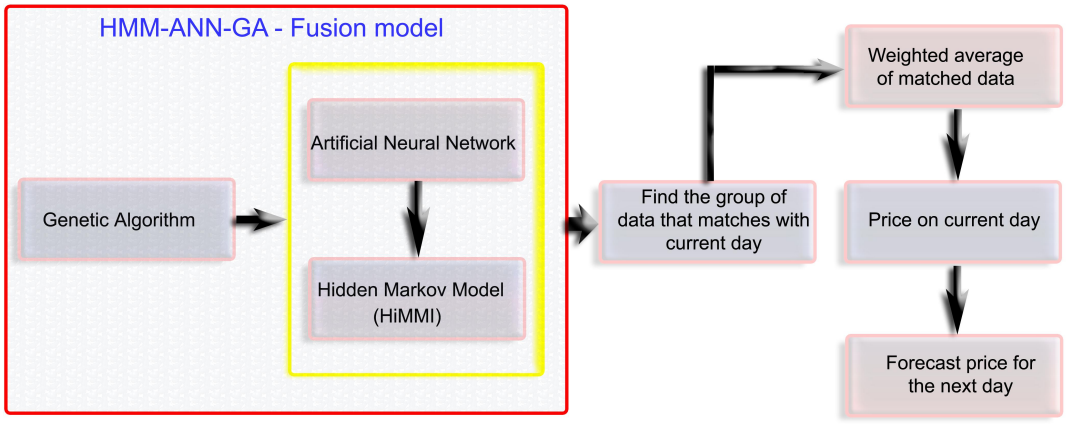


Figure 5.4: The steps in the fusion model of GA, ANN and HMM for forecasting

the challenge of developing an accurate "one day ahead" predictor for stock market time series data, we focus on developing strategies for estimating HMM parameters and pre-processing the input observation sequences. In the fusion model (see Fig. 5.4 for a high level overview), a hybrid of soft computing with HMM is presented. ANN and GA both are well established and researched SC methods. Our motivation in using only ANN and GA from a list of SC paradigms including, Ant Colony Optimization, Artificial Immune System, Swarm Optimization etc. in our fusion model was based on their simplicity and their proven application records. Here, an ANN is employed as a "black-box" to introduce noise to the observation sequences so that they may be better fitted with the HMM. That is, the ANN is used to transform the daily stock prices into an independent sets of uncorrelated values, which then become the input values for the HMM. The GA is then used to find the optimal initial parameters for the HMM given the transformed observation sequences. The fusion model is likely to find a number of alternative data items from the historical data, which display similar behavior (in terms of price fluctuations) to that of the current day. A weighted average of the price difference for each of the identified matching data items is then calculated. This weighted average is added to the current day's price. The resulting value determines the "one day ahead" forecast value.

In the following subsections, further details related to the optimization of HMM parameters and the algorithms used to construct the forecast using the weighted average of matched data are presented.

5.2.1 Optimization of the HMM

In the fusion model, the ANN and the GA processes are executed iteratively until a specified stopping criterion is reached (either a fixed number of iterations or the GA execution time has been exceeded). The specific implementations details are described below.

Linking of the ANN and HMM

There are strong dependencies between the observation sequences and the re-estimation criteria of the HMM [180]. Therefore, it is not unreasonable to expect that a more efficient optimization (re-estimation) process for the HMM parameters may be achieved by transforming the observation sequences. For example, to obtain an optimized HMM for pattern matching, Bengio *et al.* [180] used ANNs to transform the actual observations, which were then fed into the HMM as an input vector. After the re-estimation process, significant improvements were noted in the performance of the HMM. Following on from this work, we now explain in more detail how the observation sequences and the optimization of HMM are related to each other.

Let us assume that the transformed observation vector at a point in time t is Y_t. These vectors of observation sequences act as the inputs to the HMM. Thus the properties of the HMM are as follows:

- Y_1^T = the whole observation sequence
- Y_t = an input observation at time t
- a_{ij} = the transition probability from state i to state j
- b_i = the emission probability from state i and
- the probability that the HMM generates Y_t in state S_t at time t is denoted as $b_{i,t} = \Pr(Y_t|S_t = i)$

The algorithms described in Chapter 3 are used recursively to compute the following probabilities for the partial observation sequences up to time t with some boundary conditions assumed:

$$\begin{aligned}\alpha_{i,t} &= \Pr(Y_1^t \quad and \quad S_t = i|HMM) \\ &= b_{i,t}\sum_j a_{j,i}\alpha_{j,t-1}\end{aligned} \tag{5.2}$$

$$\begin{aligned}\beta_{i,t} &= \Pr(Y_{t+1}^T|S_t = i \quad and \quad HMM) \\ &= \sum_j a_{i,j}b_{j,t+1}\beta_{j,t+1}\end{aligned} \tag{5.3}$$

In Eq. (5.2) and Eq. (5.3), the variables $\alpha_{i,t}$ and $\beta_{i,t}$ are the forward and backward variables as defined in Chapter 3.

The re-estimation (or the optimization) of the HMM parameters can be carried out considering different criteria and uses for the probabilities obtained in Eq. (5.2) and (5.3). Typically, the maximum likelihood (ML), maximum mutual information (MMI) or maximum likelihood estimation (MLE) are used (see [181] for an overview).

If the MLE method is used for an HMM, then $b_{i,t}$ is assumed Gaussian mixture given by Eq.(5.4):

$$b_{i,t} = \sum_k \frac{Z_k}{((2\pi)^n|\sum_k|)^{\frac{1}{2}}} exp(-\frac{1}{2}(Y_t - \mu_k)\sum_k^{-1}(Y_t - \mu_k)^T) \tag{5.4}$$

where, n is the number of observation features of the HMM.

The transition probabilities a_{ij}, normal distribution mean vectors μ_k , covariance matrices $\sum_k$, and gains Z_k can be estimated as described in Chapter 3 and in [147]. As the optimization criterion $'C'$ used for the HMM depends on Y (the output produced by the ANN) it is possible to express $'C'$ as a function of Y and derive the following equation, using the chain rule:

$$\frac{\delta C}{\delta Y_{j,t}} = \sum_i \frac{\delta C}{\delta b_{i,t}}\frac{\delta b_{i,t}}{\delta Y_{j,t}} \tag{5.5}$$

Equation (5.5) shows that the optimization criterion C depends on the mutual correspondence (or, we can say, the degree of suitability between the input sequences and the observation emission probabilities) between the input sequence Y_{jt} (j^{th} observation at time t) and the observation emission probabilities $b_{i,t}$.

In our fusion model we consider the dependency of the optimization criterion of the HMM on the observation sequence. However, we do not follow the global optimization methodology described in [180]. As an alternative we employ a GA to obtain optimized initial parameter values of the HMM, which, after the training, best fit with the trans-

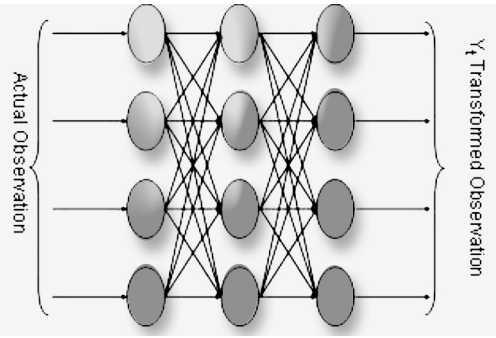


Figure 5.5: An ANN architecture to transform the correlated data features to uncorrelated

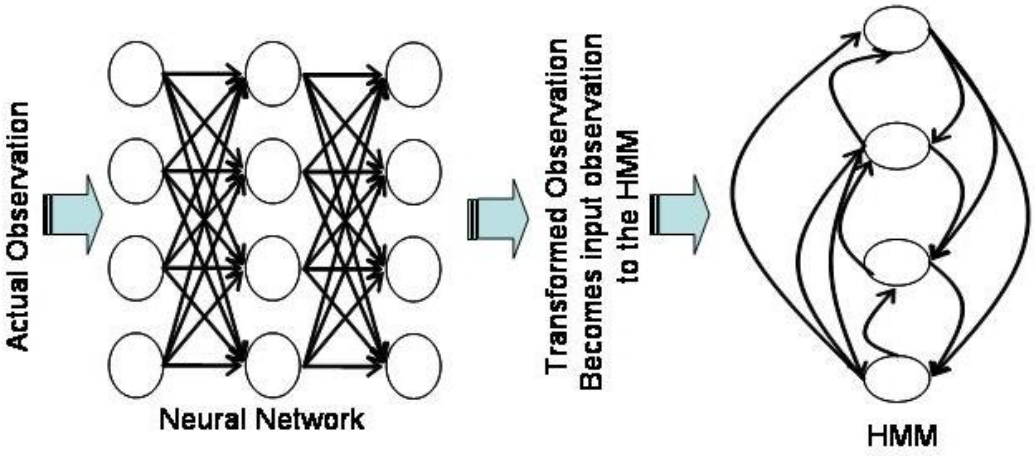


Figure 5.6: The linking of ANN and HMM. The transformed observations using the ANN are fed as input to the HMM

formed observation sequences Y. In summary, in order to combine an ANN with the HMM system, we follow these steps:

1. Create an ANN structure randomly consisting of n input nodes and n output nodes (where n is the number of predictors).

2. Initialize the weights of the ANN randomly.

3. The actual observation vectors represent the input values of the ANN.

4. The output vector Y_t produced by the ANN at time t is fed to the HMM as input observation vector.

Figure 5.5 and 5.6 illustrate the steps in the above integration process. Figure 5.7 and

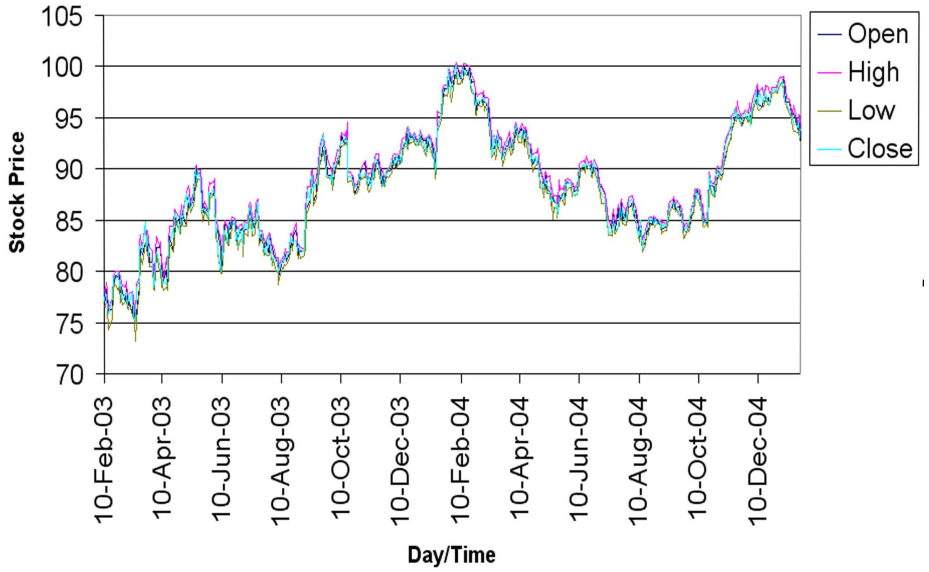


Figure 5.7: The actual observations which are highly correlated to each other

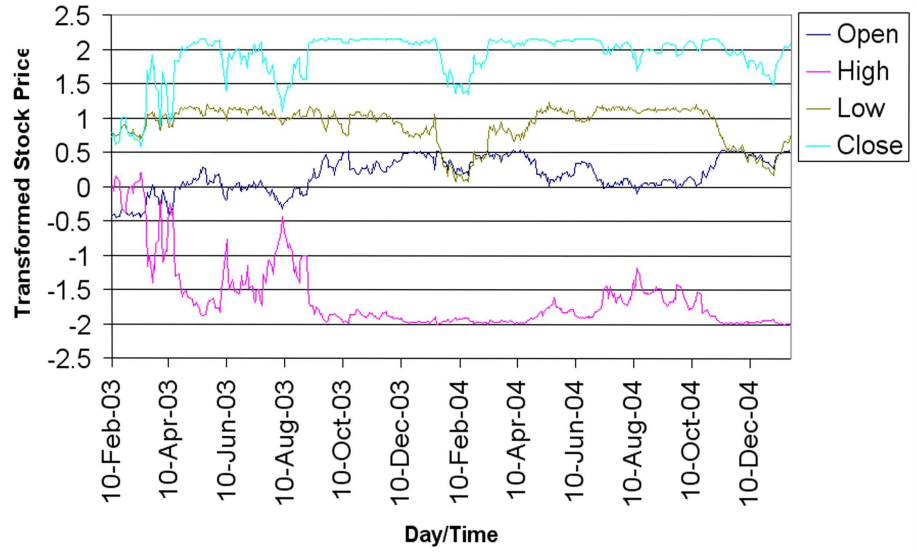


Figure 5.8: The observations after the transformation using the ANN which become uncorrelated

5.8 present a snapshot of the actual observation sequences for a typical stock and the transformed observation sequences using the ANN respectively.

Linking of the GA and HMM

In our HiMMI model, the values of the transition matrix, observation probability matrix and prior probability matrix were chosen randomly. In addition, the parameters of the HMM were re-estimated using the Baum-Welch expectation maximization algorithm.

Stock market time series, and the corresponding transformed data in Section 5.2.1, are continuous and thus it is a difficult task to identify the underlying data distribution. For a continuous HMM, the observation emission probability distribution can be represented by Gaussian distribution (Eq. 5.4). Still we cannot assume the initial values of the mean and covariance matrices for the Gaussian distribution precisely. This poses additional challenges that must be addressed.

In the fusion model, the GA is used to optimize the HMM initial parameter values for a given input sequence, generated by the ANN (see Fig. 5.9). Here, the GA is used to evolve the optimized initial values of the HMM parameters, so that after training, the HMM is a more accurate model for the observation sequences transformed by the ANN. There are three parameters in the HMM to be optimized:

1. Observation emission probability matrix;
2. State transition probability matrix; and
3. Prior probability matrix.

If we use a canonical GA to evolve values for all the parameters at a time, the length of the chromosome becomes very large, making the overall process very computationally expensive. The alternative way is to divide the problem into three parts as is done in cooperative co-evolutionary models. Here, one parameter value of the HMM is optimized at a time, while the other parameters values are kept fixed. Figure 5.9 shows the process of optimizing the fusion model. The specific steps to obtain initial values of the observation emission probability matrix are:

1. Initially choose the parameter values randomly.
2. Execute the GA to obtain initial values of the observation emission probability matrix keeping the other initial parameter values as are found in step 1.
3. Execute the GA to obtain initial values of the state transition probability matrix keeping the initial observation emission probability matrix values obtained in step 2 and the prior probability values generated in step 1.

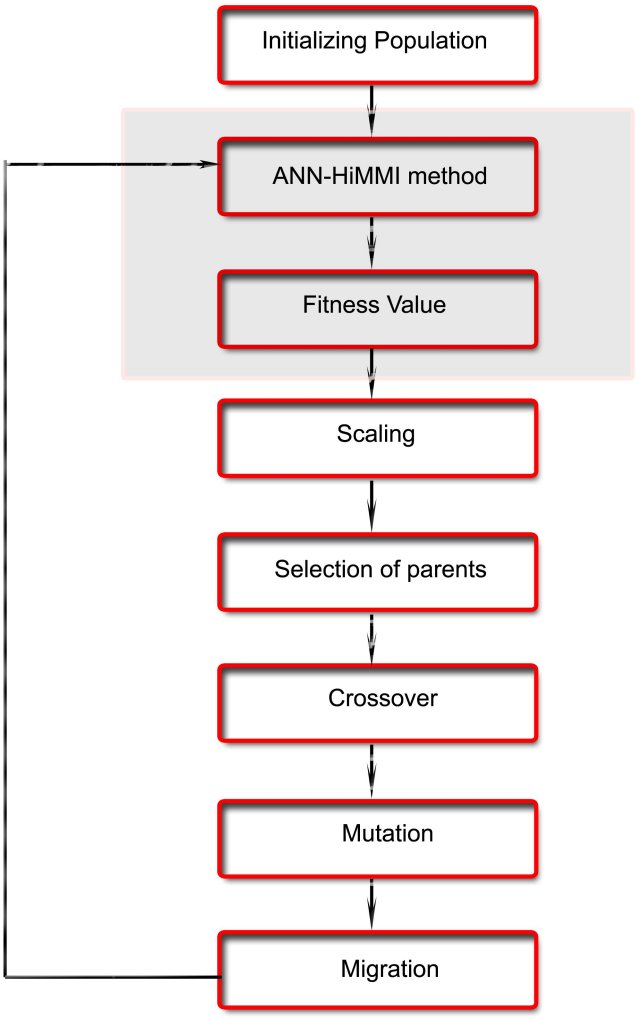


Figure 5.9: Steps in the GA to optimize the initial parameter values of the HMM used in the ANN-HiMMI model

4. Execute the GA to obtain initial values of the prior probability matrix keeping the initial observation emission probability matrix values obtained in step 2 and the state transition probability matrix values obtained in step 3.

5. If the resulting fitness value converges, go to step 2.

We have adopted a real-encoded chromosome for the GA. The length of the chromosome is equal to the number of parameters to be optimized. That is, each parameter to be optimized has a real-encoding. The fitness (or objective) function is simply the Mean Absolute Percentage Error (MAPE) of the ANN-HMM forecast method - a minimization problem. The specific algorithm steps are:

1. Initialize population.
2. Evaluate population.
3. Train the HMM by using each chromosome of the population as initial emission probability matrix and the transformed observation sequences.
4. Obtain forecast value for the validation dataset by finding the similar data vector and calculating the price difference.
5. Calculate the MAPE value for the validation dataset.
6. If termination condition is not met go to the next step else exit.
7. Select parents, based on the MAPE produced, for the next population.
8. Perform crossover and mutation.
9. Evaluate population.
10. Go to step 3.

5.2.2 Weighted average forecast

In the HiMMI model, a single data item that matches the current day's stock behavior is identified from historical data and the corresponding price difference is calculated. This is done assuming that, if the stock behavior on m^{th} day is found to be the same as that of current day c the stock behavior on day $c+1$ will be same as that of day $m+1$. However, this approach does not take into consideration recent stock market behavior, which has a profound influence on the future direction of the market. Many statistical forecasting techniques (e.g. Random walk, Moving Average, Auto Regression, Weighted Moving Average, etc.) do consider recent price fluctuations in the stock market. Thus it is reasonable to consider incorporating such trends into our fusion model.

Here, we utilize a technique that attaches more importance to matching data items from the recent past to higher rather than older matching data. To implement this technique, we need to identify a range of data vectors from the past training dataset that have produced log-likelihood values close (within a fixed threshold) to that of current data vector. That is, a group of days are identified where the price fluctuation was similar to that of the current day. Now, for each of the matched days, the price differences between those days

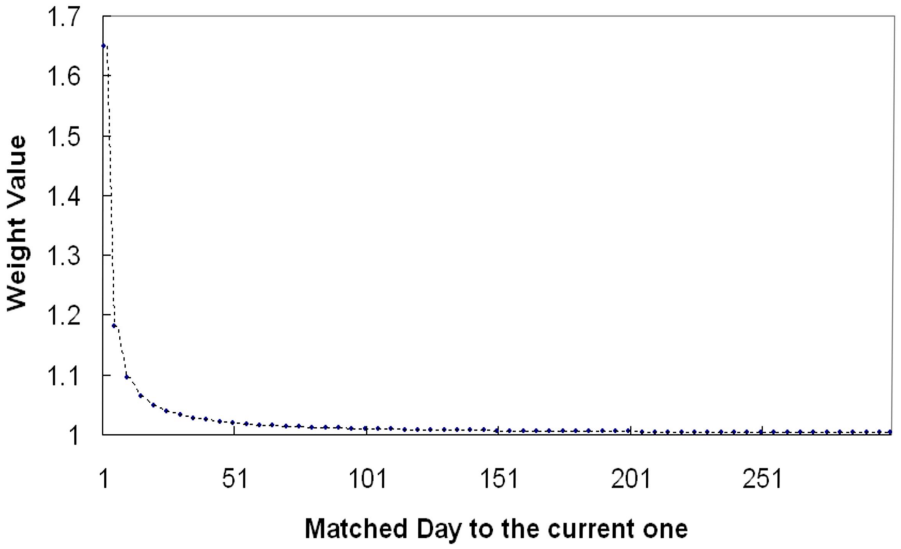


Figure 5.10: The weights assigned to the day. The less value in matched day indicates the found matched pattern is close to current day in time

and "*one day ahead*" neighboring days are calculated. Then the weighted average of these price differences are calculated using Eq. (5.6).

Let us assume that the training dataset is ordered in the following sequence: *a, b, c, d, e, f, g, h, i, j* etc., where j is the current day and a is from the distant past. Suppose the day *a, c, d, h* and i matched with that of day j. Denoting the price difference between days x and $x+1$ is by $diff_x$ and, assigning proportionally more weight to recent matches, a weighted average of the price differences is calculated. The plot in Fig. 5.10 illustrates how more weights are assigned to recent days than to distant past days. The following equation calculates the weighted average of the price differences:

$$wd_i = \frac{\sum_m w_m * diff_m}{\sum_m w_m} \tag{5.6}$$

where, i = index number of current day,

m = index number of matched day,

w_m = weight assigned to the day m using the equation $w_m = exp(\frac{1}{i-m+1})$,

wd_i = weighted average of price difference for current day i,

$diff_m$ = price difference between day m and $m+1$.

Thus, a forecast value for day $i+1$, fp_{i+1}, is constructed as in the Eq. (5.7).

$$fp_{i+1} = p_i + wd_i \tag{5.7}$$

where, p_i = Price on current day i.

5.3 Experiment and Result

5.3.1 Stock Market Forecasting using the HiMMI

To evaluate the performance of the HiMMI, we consider a number of stocks drawn from the airlines sector and IT sector respectively: namely British Airlines, Delta Airlines, Ryanair Airlines, Apple Computer Inc, International Business Machines Corporation (IBM) and Dell Inc. For each company, four attributes open, high, low and close price from the daily stock market prices were used to form the observation vector. The forecast variable is the next day's closing price.

Training and Test data

In order to evaluate the forecast performance of the HiMMI, the dataset is divided into two sets: 1) Training set and 2) Test set. Table 5.2 shows the information of the training and test datasets.

Table 5.2: Training and Test dataset

Stock Type	Stock Name	Training Data		Test Data	
		From	To	From	To
Airline Sector	British Airlines	17-Sep, 2002	10-Sep, 2004	11-Sep, 2004	20-Jan, 2005
	Delta Airlines	27-Dec, 2002	31-Aug, 2004	01-Sep, 2004	17-Nov, 2004
	Ryanair Airlines	06-May, 2003	06-Dec, 2004	07-Dec, 2004	17-Mar, 2005
IT Sector	Apple Computer Inc.	10-Feb, 2003	10-Sep, 2004	13-Sep, 2004	21-Jan, 2005
	IBM Corporation	10-Feb, 2003	10-Sep, 2004	13-Sep, 2004	21-Jan, 2005
	Dell Inc.	10-Feb, 2003	10-Sep, 2004	13-Sep, 204	21-Jan, 2005

Experimental setup

The HMM in HiMMI was built by choosing the number of states equal to the number of predictor features. The initial parameter values of the HMM were chosen by selecting random numbers.

Performance metrics

The primary performance metric used in this study was the Mean Absolute Percentage Error (MAPE) in accuracy. This value was calculated by first taking the absolute deviation between the actual value and the forecast value. Then the total of the ratio of deviation value with its actual value is calculated. The percentage of the average of this total ratio is the mean absolute percentage error. The following equation shows the process of calculating the MAPE.

$$\text{Mean Absolute Percentage Error (MAPE)} = \frac{\sum_{i=1}^{r}(\frac{abs(y_i - p_i)}{y_i})}{r} * 100\% \tag{5.8}$$

where, r = total number of test data sequences,

y_i = actual stock price on day i, and

p_i = forecast stock price on day i.

Result

Column 1 in Tab. 5.3 shows the prediction accuracy of the HiMMI model in terms of Mean Absolute Percentage Error (MAPE). Figure 5.11 shows the prediction accuracy of the model and Fig. 5.12 shows the correlation among the predicated values and the actual values of the stock of Delta Airlines.

5.3.2 Stock Market Forecasting using the Fusion Model

To validate the effectiveness of the fusion model as a forecasting technique, we report the results of a number of experiments below.

Test data

The same dataset as in the previous experiment has been used to evaluate the forecast performance of the fusion model.

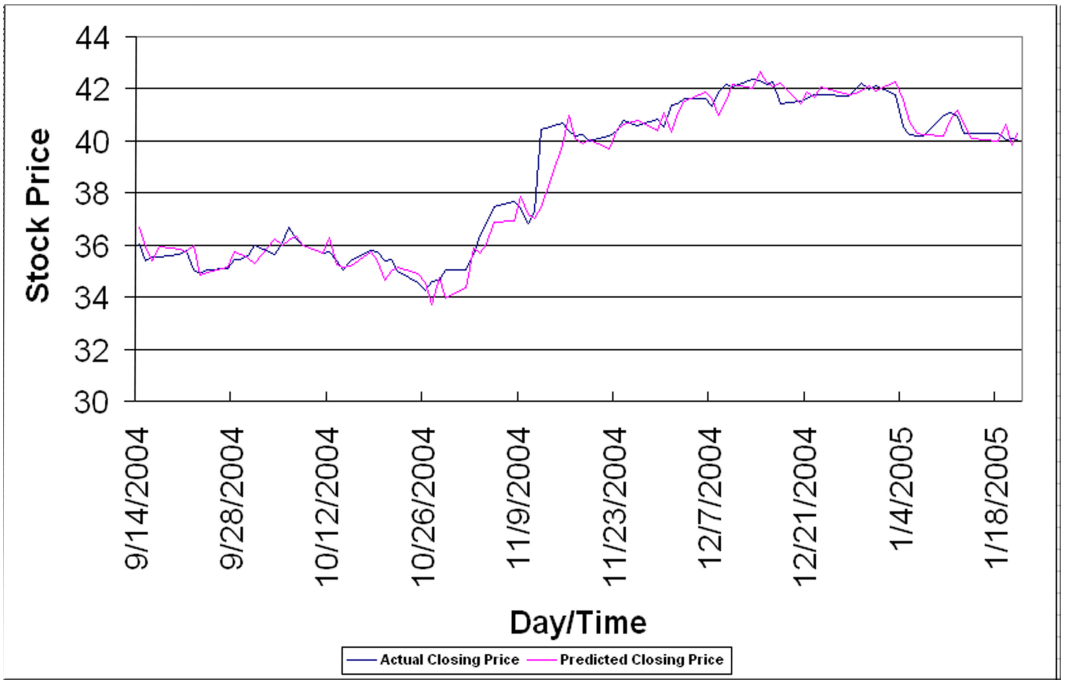


Figure 5.11: Actual vs Forecast(using HiMMI) closing stock prices of Delta Airlines

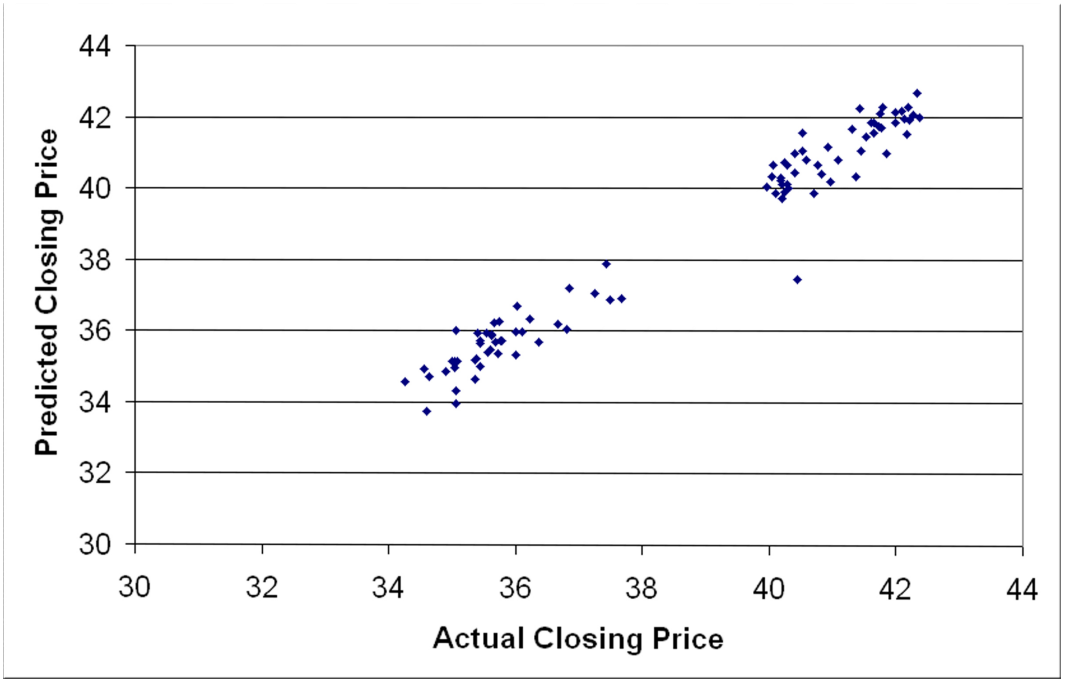


Figure 5.12: The correlation among the actual and forecast (using HiMMI) closing prices of the Delta Airlines

Experimental setup

The number of states in the HMM was set to the number of attributes in the observation vectors, as per the HiMMI.

In the ANN section of the model, we use a three-layer (an input layer, hidden layer and an output layer) feed-forward fully connected architecture as is shown in Fig. 5.5. There are four neurons in the hidden layer, while there must be four input neurons (due to four input variables) and four output neurons. The activation function for the hidden and output neurons were a tanh sigmoidal function and linear function respectively.

The HMM parameters were optimized using the GA. A Gaussian normal probability distribution was chosen for the emission probability distribution. The values for the prior probabilities and the state transition probabilities of the HMM were initialized to 1 / N, where N is the number of states. The GA chromosome encoded specific HMM probabilities and as such, varying lengths were used; i.e., the emission probability distribution was set to length 16 (4 states and 4 distinct variables); the prior probability was set to length 4, and the transition probability was set to length 16 respectively. Table D.1 in the Appendix lists the parameter values for GA used in the model.

Performance metrics

As in the previous experiment using HiMMI, in this experiment we have used the same performance metrics, MAPE, to evaluate the efficiency of the fusion model.

Results

Table 5.3 presents the experimental results for each of the six stocks considered. The table lists results for the base HiMMI (second column) and the two methods used to arrive at the final prediction using the fusion model – the interpolation method (column three) and the weighted average (column four). It is interesting to note the improved performance of the fusion-weighted average model.

We compare the performance of the HiMMI and the fusion model with that of the ANN and ARIMA model. We have used the same training dataset to train these models and obtained prediction values for the same test dataset. Column 4 in Tab. 5.4 shows the prediction accuracy for the test dataset using ANN. To obtain the best ANN architectures, we have used the EA and the results are produced using the ANN. The parameter values

of the respective EAs are provided in the Appendix.

In Tab. 5.4 column 5, we present comparative results between the fusion model and an ARIMA(p, d, q) model in terms of MAPE. To use the ARIMA model for the dataset described here, we first analyzed the series to obtain the order of p, d and q, (p – The autoregressive parameter, q – The moving average parameter, d – the number of differencing passes). As a result, the best possible ARIMA(p, d, q) model is built for each of the stock prices used in this experiment. The ACF and the PACF plots for each of the datasets are available in Appendix.

Table 5.3: Fusion model performance. The Mean Absolute Percentage Error (MAPE) in the forecast for unseen test dataset

Stock	HiMMI	Fusion Model	
		Interpolation	Weighted Average
British Airlines	2.629	2.450	1.646
Delta Airlines	6.850	6.650	5.529
Ryanair Airlines	1.928	1.920	1.377
Apple Computer Inc.	2.837	2.165	1.925
IBM Corporation	1.219	1.056	0.849
Dell Inc.	1.012	0.845	0.699

Table 5.4: Fusion accuracy comparison with the ARIMA. Mean Absolute Percentage Error (MAPE) in the forecast for the 91 sequential test dataset

Stock	HiMMI	The fusion model with weighted average	ANN	ARIMA
British Airlines	2.629	1.646	2.283	1.573
Delta Airlines	6.850	5.529	9.147	5.294
Ryanair Airlines	1.928	1.377	1.492	1.504
Apple Computer Inc.	2.837	1.925	5.491	1.801
IBM Corporation	1.219	0.849	1.132	0.972
Dell Inc.	1.012	0.699	3.049	0.660

5.4 Summary

In this chapter, we have developed two models for forecasting financial data, each of which is based on a single HMM. In the first model, called HiMMI, we located the pattern(s) from the past datasets that match with current stock price behavior and then interpolated

these two datasets with appropriate neighboring price elements and forecast the stock price for the next event/time unit. This HMM-based forecasting method performed well for the six daily stock prices considered in this thesis, as shown in Tab. 5.3 Column 1. Table 5.4 reports the forecast error using HiMMI and compares it with that of ANN and ARIMA. It is interesting to note that the performance of HiMMI is as good as that of the ANN.

In our second method, we developed a fusion model by combining the HMM with ANN and GA. In this fusion model an ANN is used to transform the input observation sequence of the HMM, and the GA is used to optimize the initial parameters of the HMM. This optimized HMM is then used to identify similar data patterns from the historical data.

In the fusion model, two alternative methods are examined for constructing the "one day ahead" forecast value based on the optimized HMM identified patterns. In the first approach, we interpolate the neighboring values of the identified data patterns to calculate the forecast value, as in HiMMI. In the second approach, we locate a number of similar data patterns from the historical data and then use a weighted average of the difference values between neighboring data items. This value is then added to the current value of the variable of interest to obtain the corresponding forecast value. The rationale behind this approach is that the data from the most recent past should have a greater impact on the final forecast value.

Using the fusion model we generated forecast values for the same six stocks that we used to evaluate the HiMMI. The same data is used to train ANN and another well established statistical tool, ARIMA. Then we produced forecast values for the test data using the respective forecast model, i.e., ANN and ARIMA. As we see from Tab. 5.4, the performance of our fusion model is better than that of ANN, as reflected in the MAPE measurement. The primary weakness with ANNs is the inability to properly explain the models. It has been already shown that the problem of designing and learning for feed-forward networks is NP-complete [22]. In contrast, the initial HMM-based forecast model HiMMI and the fusion model proposed in this chapter is explicable and has a solid statistical foundation.

Looking at Tab. 5.4, we find that the accuracy of the forecast value of the proposed fusion model is as good as that of the ARIMA model. While generating a forecast using ARIMA, we had to analyze the training dataset to choose the values of p, d, and q. As we know, when using the variable of interest, an autocorrelation coefficient function (ACF) is generated and a partial autocorrelation coefficient function (PACF) is also generated.

Then, analyzing both of these functions together, a decision is made about the values of the p, d, and q (see the ACF and PACF plots in the Appendix). These are carried out to examine whether there is any seasonality in the series. In fact, most statistical tools for forecasting are constrained by the underlying seasonality and non-stationarity (if any)(Tambi, 2005). In contrast, the proposed fusion model does not require any prior analysis of the dataset while using the statistical theories (as HMM provides a probabilistic framework for modeling a time series of multivariate observations).

In the first phase of the proposed fusion model, the ANN transforms the actual observations sequences to a set of independent values. Figure 5.8 and 5.7 show the transformed and the actual observation sequences. It is hard to differentiate the four sequences when they are in actual format (see Fig. 5.7). However, in the transformed figure (Fig. 5.8), the four individual sequences can be identified easily. Hence, it is more likely that the transformed observation sequences might play a better role in predicting the value of variable of interest. Simultaneously, the GA in Section 5.2.1 optimizes the initial parameters so they are well suited with the transformed observation sequences. The transformed independent values along with the optimized initial parameters help the HMM to find out the similar stock behavioral data from the historical dataset more efficiently and accurately. Table 5.3 shows the forecast accuracy improvement over the HiMMI (Hidden Markov Model followed by interpolation) after integrating the ANN and GA with the HMM. Equation (5.6) gives more weight to the recent matched stock data than to the far distant matched data. Figure 5.10 shows how more importance is given to the recent matched data. Applying the equation of weighted average to the integrated HMM-ANN-GA (fusion) model, the forecast accuracy of the method has been boosted significantly (see column 4 of Tab. 5.3).

In this chapter we have used HMM for obtaining similar patterns from historical datasets and developed two alternative methods to generate the "one day ahead" forecast value of some stocks. The performance of the developed models for forecasting non-linear time series data (see the Appendix for the non-linearity test of the considered time series data) is encouraging and is competitive with that of some contemporary time series forecasting methods. It has been noted that the characteristics of the daily stock prices of the six stocks considered in this chapter exhibit that the underlying non-stationarity can be transformed to a stationary process (as reflected by the ACF and PACF plots shown in the Appendix). An auto-regressive process can model such a system efficiently, as we see in the experimental results.

In the next chapter we shall use fuzzy logic to better handle the underlying non-stationarity (where it is difficult to remove the underlying non-stationarity) of any non-linear time series data, along with the HMM-based pattern matching technique discussed in Chapter 4.

Part III

HMM-based Fuzzy model generation

Chapter 6

HMM-based fuzzy model for time series forecasting

In this chapter, we present Fuzzy Logic (FL) in combination with HMM in order that the prediction accuracy is notably improved for non-stationary and non-linear datasets. Fuzzy modeling is a SC paradigm that has been used to analyze and predict multidimensional non-linear time series. The structure of a fuzzy model is usually chosen by bringing together the available expert knowledge for the system to be modeled, or by adopting some means of generating suitable rules from the numeric data available. The later approach is described as a data-driven model. In data-driven fuzzy models, the set of input-output relations from the available dataset are extracted. Typically, a two-stage process is used to build the fuzzy model: (i) Structure determination/Identification, and (ii) Parameter(s) Fine Tuning.

Structure determination, by generating appropriate and accurate fuzzy rules, is quite a challenge, in that even for a small number of data features in the dataset there is potentially a large number of rules that can be generated. There are several alternative methods that can generate fuzzy rules representing the input-output relationship that has been described in the literature [118,182,183]. However, a few studies have tried to reduce the complexity of the overall system by trying to attain an optimal number of fuzzy rules, with varying success. Moreover, in traditional data-driven fuzzy model identification techniques, the relationships among the data features in the data vector are not considered, and this invariably affects the resultant performance.

In this chapter, we present an HMM-based dataspace partition to generate fuzzy rules. The HMM-based data partitioning technique includes the relationship between data fea-

tures in the data vectors. Our approach combines the HMM's data partitioning approach with the generation of fuzzy logic for the prediction of multivariate time series data. Parameters of the generated fuzzy rules are further fine tuned using a gradient descent algorithm. We evaluated the performance of the model with one of the well-known non-linear and non-stationary benchmark data; namely the Mackey-Glass time series [62], and compared our results with other techniques reported in other recent studies. Furthermore, we prepared forecasts for the six stock that were considered in the previous chapter, to note the performance improvement of the proposed HMM-Fuzzy model compared to the HMM-based forecasting.

The remainder of this chapter is organized as follows. Section 6.1 introduces fuzzy rules and reviews some of the existing techniques for generating fuzzy rules. In Section 6.2, details of the HMM-Fuzzy model are described including the algorithms. Section 6.3 provides experimental results. Finally, in Section 6.4, we discuss the method and conclude the chapter.

6.1 Fuzzy Rules

A Fuzzy Rule is a linguistic representation of the relationship between the set of independent features and the set of dependent features. Fuzzy rules make use of fuzzy sets to logically describe relationships; e.g., the following rule makes use of the fuzzy linguistic term:

If x_1 is *very* cold and x_2 is *quite* low then y is *very* satisfactory.

This rule maps features x_1 and x_2 to y and uses the fuzzy terms *very*, *quite* and *very*. In order to determine the exact degree to which the feature's current conditions are satisfied, membership functions are used. Typically, a number of different types of membership functions are used to quantify the linguistic term in the rules. Some of the most common membership functions are [184]:

- Gaussian function

$$A_i(x_i) = e^{-\frac{1}{2}(\frac{x_i-\mu_i}{\sigma_i})^2} \tag{6.1}$$

where, A_i = Membership function for the i^{th} data feature – x_i in the dataset,
μ_i = Mean value of the i^{th} data feature – x_i in the dataset,

σ_i = Standard deviation value of the i^{th} data feature – x_i in the dataset.

- Bell shape function

$$A_i(x_i) = \frac{1}{1 + |\frac{x_i - c}{a}|^{2b}} \tag{6.2}$$

where, c = Center of the membership curve, usually the median of the i^{th} feature x_i in the dataset,

b = Any positive number,

a = Starting point of the curve, usually minimum value of the i^{th} feature x_i in the dataset.

- Triangular function

$$A_i(x_i) = max(min(\frac{x_i - a}{b - a}, 1, \frac{c - x_i}{c - b}), 0) \tag{6.3}$$

where, a = Minimum value of the i^{th} feature x_i in the dataset,

b = Maximum value of the i^{th} feature x_i in the dataset,

c = The peak of the triangular graph for the i^{th} feature x_i in the dataset.

- Trapezoidal function

$$A_i(x_i) = max(min(\frac{x_i - a}{b - a}, 1, \frac{d - x_i}{d - c}), 0) \tag{6.4}$$

where, a = Minimum value of the i^{th} feature x_i in the dataset,

b = Maximum value of the i^{th} feature x_i in the dataset,

c = Edge of the left shoulder of the trapezoidal graph for the i^{th} feature x_i in the dataset,

d = Edge of the right shoulder of the trapezoidal graph for the i^{th} feature x_i in the dataset.

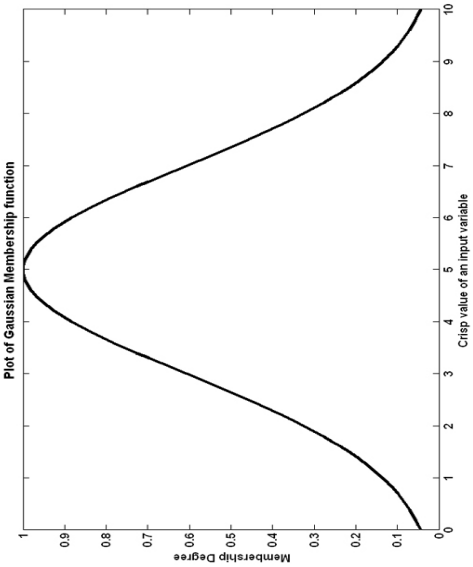


Figure 6.2: Graphical representation of Gaussian Membership function

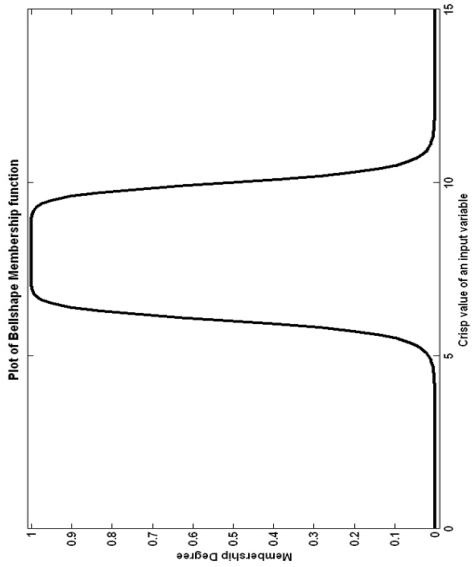


Figure 6.1: Graphical representation of Bellship Membership function

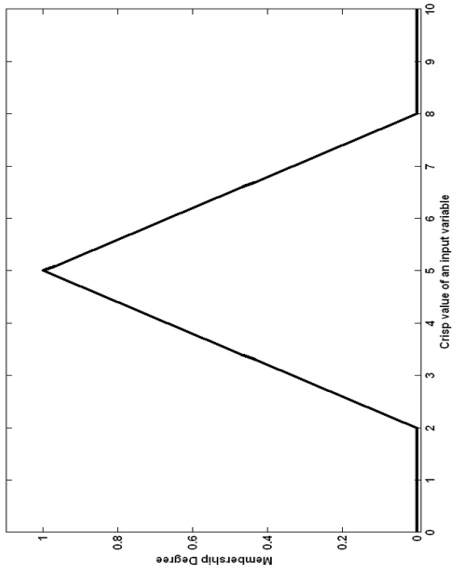


Figure 6.4: Graphical representation of Triangular Membership function

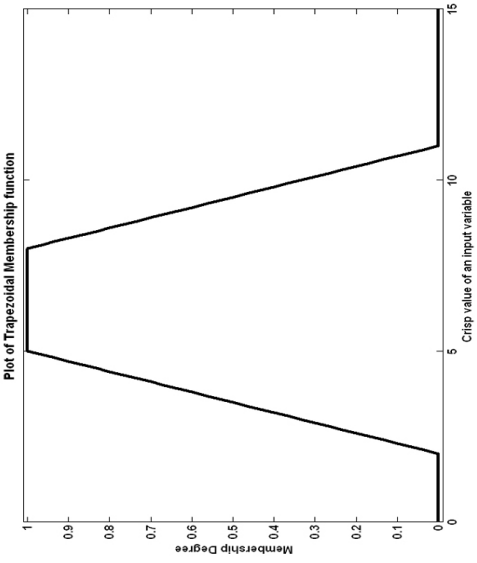


Figure 6.3: Graphical representation of Trapezoidal Membership function

6.1.1 Fuzzy Inference

Fuzzy inference is the process by which the mapping between fuzzy sets is done [39]. It combines the fuzzy rules and generates a solution. We describe some of the methodologies of fuzzy inference below.

Fuzzy System Models

In fuzzy rules, the relationships between variables are represented using fuzzy connectives and fuzzy inferences are generated with fuzzy implications. A standard syntax for a fuzzy rule is:

If antecedent proposition Then consequent proposition

There are two primary fuzzy models used, depending on the structure of the desired proposition/implication of the rule [132]:

- Mamdani fuzzy model [185],
- Takagi-Sugeno fuzzy model [186].

The distinguishing difference between these two models is that, in the Mamdani model, the consequent part is a fuzzy proposition, while in the TS model it is a crisp function of the antecedent variables.

Mamdani Fuzzy Model

Section 6.1 defined fuzzy rules as a method to represent relationships between input and output variables in linguistic terms. The general syntax to represent such a rule would be [185]:

$$\text{j'the Rule} : \text{If } \mathbf{x} \text{ is } A_j \text{ Then y is } B_j,\ j = 1, \cdot, c$$

where, c is the total number of rules in the fuzzy model [132], x is the antecedent (or input) vector and y is the consequent (output) variable. The symbols A_j and B_j are the linguistic representations about the status of the variables that are translated/quantified using membership functions $\mu_{A_j} : \mathcal{X} \rightarrow [0, 1]$ and $\mu_{B_j}(y) : \mathcal{Y} \rightarrow [0, 1]$, respectively.

In a fuzzy model, the fuzzy sets A_j in the antecedent space logically correspond to consequent fuzzy sets B_j. With fuzzy proposition, it is possible to reveal and describe

this relationship. For a k-dimensional input vector x, the fuzzy rule is represented using n fuzzy propositions, as in the following form:

j'the Rule: If x_1 is A_{j1} And x_2 is A_{j2} $\cdots$ And x_k is A_{x_jn} Then y is B_j

It may be noted that the single dimensional membership functions $A_{x_{ij}}, i = 1, \cdots, k, j = 1, \cdots c$ may possess different shapes, as shown in Fig 6.1, 6.2, 6.3 and 6.4.

Output fuzzy sets are derived by applying the fuzzy implication operator $A \rightarrow B$ and can be defined by the following formulae [132]:

1. **Minimum (Mamdani)** $\mu_{A\rightarrow B}(x, y) = min[\mu_A(x), mu_B(y)]$
2. **Product (Larsen)** $\mu_{A\rightarrow B}(x, y) = \mu_A(x)\dot{m}u_B(y)$
3. **Zadeh implication** $\mu_{A\rightarrow B}(x, y) = max\{min[\mu_A(x), mu_B(y)], 1 - \mu_A(x)$
4. **Lukasieqicz implication** $\mu_{A\rightarrow B}(x, y) = min[1, 1 - \mu_A(x) + mu_B(y)]$

The membership value $\mu_A(x)$ in the above expression is calculated by intersecting the cylindrical/bell-shape extensions of $A_{x_i}, i = 1, \cdots, k$. Two commonly used intersection operations are [132]:

1. **Minimum** $\mu_{A\rightarrow B}(x, y) = min[\mu_{A_{x_1}}(x), mu_{B_{A_{x_2}}}(x), \mu_{A_{x_3}}(x), \cdots, \mu_{A_{x_k}}(x)]$
2. **Product** $\mu_A(x) = \mu_{A_{x_1}}(x) \cdot \mu_{A_{x_2}}(x)$

For the sake of convenience, gradient-based optimization algorithms are used to fine tune fuzzy rule parameters, and product implication and product intersection are used with either Gaussian or bell-shaped membership functions to design the fuzzy model. In this model the consequent membership functions are presumed to be symmetric with centers at $y_j, j = 1, \cdots, c.$ with equal spread as in Eq. (6.5).

$$\mu_{A_j\rightarrow B_j}(x, y) = \mu_{A_j}(x) \cdot y_j \tag{6.5}$$

With regard to fuzzy models, the process of extrapolating crisp output from the fuzzy output is named 'defuzzification'. The defuzzification process in a Mamdani fuzzy model is determined by calculating the Center of Gravity (COG) as in Eq. (6.6).

$$y_{FS} = \frac{\sum_{j=1}^{c} \mu_{A_j}(x) y_j}{\sum_{j=1}^{c} \mu_{A_j}(x)} \tag{6.6}$$

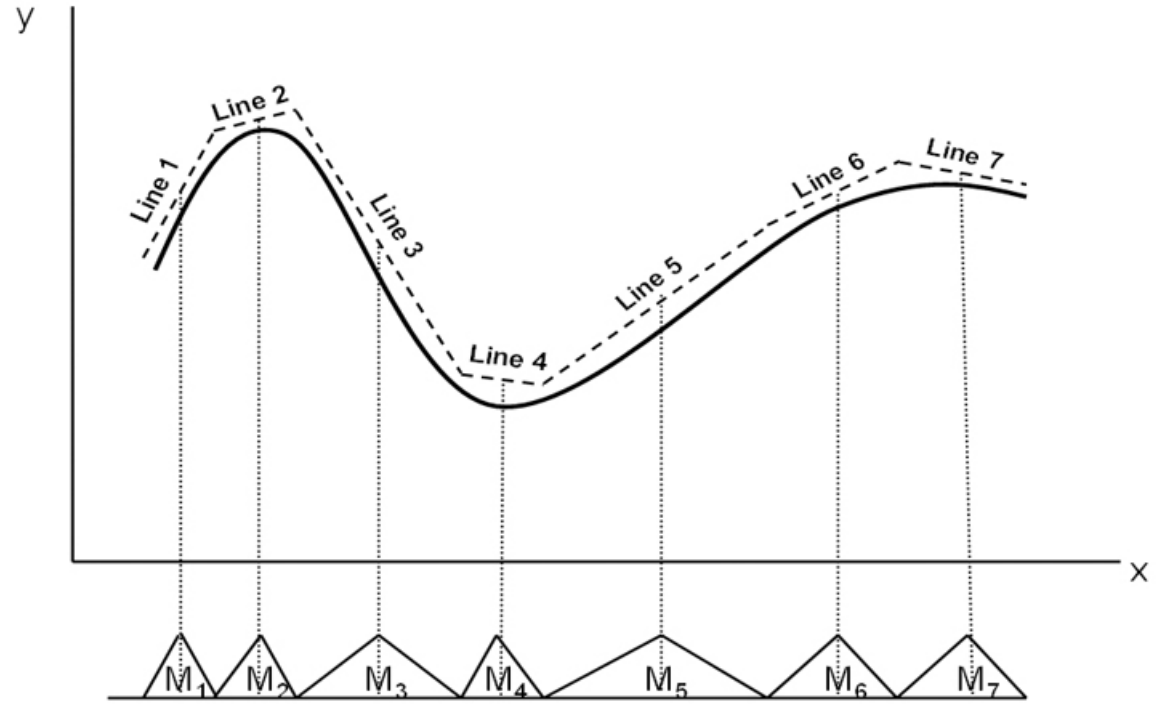


Figure 6.5: The structure of a TS model

In contexts involving the varying spread in consequent membership functions the defuzzification Center of Gravity (COG) is computed using the weighted COG method:

$$y_{FS} = \frac{\sum_{j=1}^{c} \mu_{A_j}(x) w_j y_j}{\sum_{j=1}^{c} \mu_{A_j}(x) w_j} \tag{6.7}$$

where, w_j is the area of the consequent fuzzy set B_j.

Takagi-Sugeno Fuzzy Model

Takagi and Sugeno [186] proposed a fuzzy system appropriately named the Takagi-Sugeno (TS) Fuzzy Model. The model comprises a set of fuzzy rules such that each rule has two parts: a premise and a consequence. The consequence is usually a linear combination of all variables or part of the variables in an input space, and is usually denoted as a function of the input variables. Non-linearity in the dataset is considered to be a combination of linear representations as illustrated in Fig. 6.5. As soon as representative straight lines are obtained for non-linear data, the TS fuzzy model is generated and the membership function

of each of the linear representations is derived as shown in the figure. The mathematical equation for each of the linear representations is a first-order polynomial, which is used to represent a fuzzy rule in the model. The representation syntax for such a fuzzy rule would be [132]:

$$\text{j'the Rule : If } \mathbf{x} \text{ is } A_j \text{ Then y is } b_{j0} + b_{j1}x_1 + b_{j2}x_2 + \cdots + b_{jn}x_n,\ j = 1, \cdot, c$$

Based on the above mathematical equation, each of the lines in Fig. 6.5 will be represented as follows (it may be noted that this is the same equation that represents a straight line in geometry):

$$\text{Line } i : y = b_i + b_i x_i \tag{6.8}$$

As per our discussion, TS-Fuzzy systems would each have consequence defining a linear mapping or, according to [187], what in geometrical terms is called a hyper plane in the input-output space. Ergo, the consequent part or defuzzification of each rule is computed to be a combination of the input variables represented in Eq. (6.9) [132].

$$z_{FS} = \frac{\sum_{j=1}^{c} \mu_{A_j}(x)(b_{j0} + b_{j1}x_1 + b_{j2}x_2 + \cdots + b_{jn}x_n)}{\sum_{j=1}^{c} \mu_{A_j}(x)} \tag{6.9}$$

The output produced by defuzzifying a TS-Fuzzy system is a crisp and precise value, which makes it different from the Mamdani-Fuzzy system.

6.1.2 Data-driven fuzzy model

To build a fuzzy model specifically designed to solve a non-linear problem, fuzzy rule generation is the first hurdle to be crossed. There are two ways to generate fuzzy rules:

1. By using available expert knowledge about the solution space; and

2. By analyzing available datasets.

A real life example of generating rules from expert knowledge is to automate a HVAC (Heating, Ventilation and Air-Conditioning) system to prevent windscreen fogging / misting. The set of rules would be as follows [188]:

Problems exist viz. non-linear time series modeling, robot movement navigation, that do not as yet have expert knowledge available, but datasets reflecting solutions are available. To build fuzzy systems for solving such problems, we require a means to generate fuzzy rules from the available dataset. Several methods exist that generate fuzzy rules

IF Humidity is HIGH, THEN Recire is LOW
IF Humidity is HIGH, THEN Blower is MEDIUM
IF Fog_ProbE is HIGH, THEN Recire is ZERO
IF Fog_Prob is HIGH, THNE Blower is HIGH
IF Fog_Prob is HIGH, THEN Target is HIGH
IF Fog_Prob is HIGH, THEN Target Mode is DEFROST
IF Fog_Prob is MED, AND Incar is WARM
THEN Mode is PNL/DEF
IF Fog_Prob is MED, AND Incar is COOL THEN Mode is FLR/DEF

Figure 6.6: The rules generated for an automotive HVAC system to prevent windscreen fogging [188]

from datasets [118, 189]. In order to efficiently generate fuzzy rules, a compaction of the complete system is required, and this is usually achieved by minimizing the number of rules without negatively affecting overall performance. Compacting fuzzy models brings into play an influential factor: the partitioning of the input space [132]. Clustering is one method used to partition dataspaces to generate fuzzy rules. Grid partitioning and tree partitioning are also used in data driven fuzzy models to generate fuzzy rules from the available datasets. These methods are described in the following.

1. **Clustering :** In clustering based rule generation, the number of clusters must be given prior to partitioning the dataset. Unless the structure of the available dataset is known, it is difficult to decide the number of clusters. Additionally, the performance of the fuzzy model is dependent on the suitability of the number of clusters, and this is because each cluster of data is used to generate a fuzzy rule.

2. **Grid partition :** In Grid partitioning approach, every input feature is used to generate independent membership functions and consequently this process suffers from exponential rule explosion as the number of input features increase. Moreover, this method could generate erroneous rules where no data instances actually occur. Referenced by [132], an example of a rule extraction method using this approach is the well-known Learning from Examples [135] and its modifications [136].

3. **Tree partition :** In Tree partitioning, more than one membership function is used per input feature, making the overall system unnecessarily complicated. Furthermore, the problem of generating rules for some empty regions still remains. Tree partitioning can be implemented using the CART approach [189].

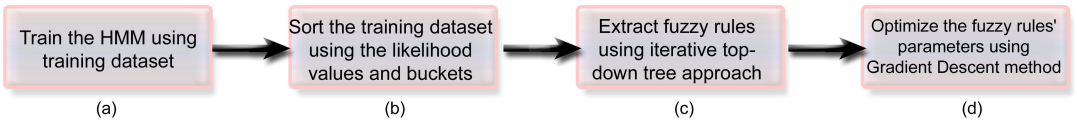


Figure 6.7: The block diagram of the HMM-Fuzzy model

All the existing rule generation methodologies that rely on data space partitioning work on the presumption that the total number of rules to be generated is determined before the actual building process commences. In this chapter, we present and develop an automated fuzzy rule generation method that does not require the number of clusters / partitions in the dataset to actually start building a data-driven fuzzy model. In the subsequent section we describe the proposed fuzzy model generation method.

6.2 HMM-Fuzzy model

This section describes the development of a fuzzy generation method based on the HMM-log-likelihood value from each of the data patterns in the dataset. This method is unique in that the number of clusters/partitions is not required prior to the fuzzy rule generation. On the other hand, it does require us to provide our desired error margin in the performance of the final fuzzy model. This model is an extended version of the HMM-based data clustering algorithm described in Chapter 4.

From a high level of abstraction, we may say that the HMM is used to partition the input space depending on the similarities in the input data sequence, as in Section 4.1 in Chapter 4. Following the steps in Section 4.1, an HMM is trained using the training dataset and then the HMM likelihood values calculated in the training stage (Fig. 6.7(a)) are ordered using a number of 'buckets' or 'bins'. Each of the 'buckets' contains similar patterns (Fig. 6.7(b)) that produces log-likelihood values within the range of the buckets start point and end point. A recursive divide and conquer algorithm (Fig. 6.7(c)) is then used to generate a set of fuzzy rules. Finally we optimize the fuzzy rule parameters using a gradient descent method (Fig. 6.7(d)).

6.2.1 Likelihood Values using the HMM

An initial HMM structure is built and the parameter values are re-estimated for a given dataset. It may be noted here that datasets fed into the HMM are usually arranged in

	Weather	Quality	Sale (Other)
Day 1	26	4	20
Day 2	24	3	44
Day 3	36	2	90
Day 4	23	4	16

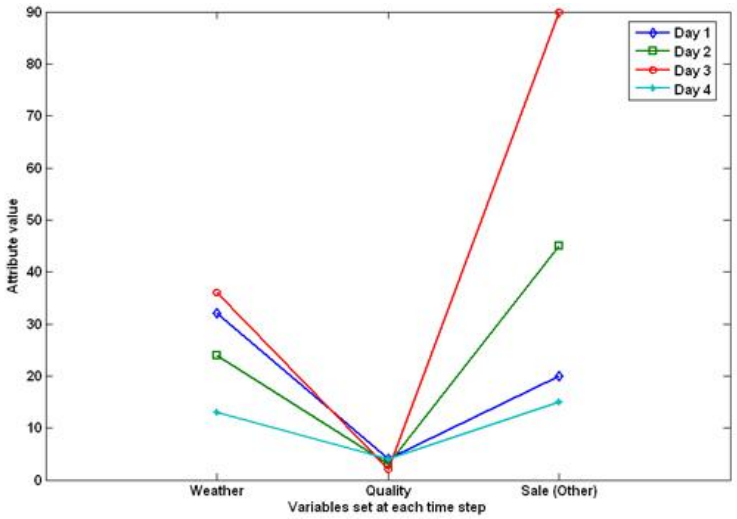


Figure 6.8: The data pattern formed by each of the data vectors

order that each data vector forms a pattern as described in Chapter 4. For multivariate time series data, each data vector must contain all the predictor/independent variables. To explain this using a hypothetical situation, let us consider the time series for the daily sales of a product; say, a fuzzy beverage. The sales figure, *amount*, of units sold may depend on factors as diverse as the weather, *weather*, the quality of the product, *quality*, competition marketing and sales, $sale_{other}$, price variations amongst similar products in the market, *price* etc. In this case, the set of predictor variables is $weather, quality, sale_{other}, price$. The values of these variables for each time unit would create a data vector for that particular time. At this stage the sequence of variables in each of the data vectors is assumed to appear consecutively, almost as if each data vector forms specific patterns, as shown in Fig. 6.8. Once the data pattern set is arranged in this manner, they are fed into the HMM to re-estimate the parameter values. The initial HMM is built as per the procedure in Chapter 4. Once the HMM is trained, this HMM is used to generate log-likelihood values for each of the data vectors in the dataset (comprising both training and testing data). To generate the log-likelihood values the forward algorithm described in Chapter 4 is used. For a total m number of data vectors/patterns m log-likelihood values are generated, each labeled with the index of the respective data vectors in the dataset.

6.2.2 Grouping of similar data sequences

The range of log-likelihood values *(l_1 to l_m, l_i=log-likelihood value produced for the ith data vector)* is split into equal sized buckets/bins. The contents of each bucket/bin would

1. bucket_size=θ;
2. startRange= minimum of the likelihood values;
3. endRange= maximum of the likelihood values;
4. i=startRange;
5. j=1;
6. while (i $\leq$endRange)

a bucket[j].start=i;

b bucket[j].end=i+bucket_size;

c bucket[j].data = find (dataLikelihood $\geq$i and dataLikelihood ¡ i+bucket_size)

d i=i+bucket_size;

e while end

Figure 6.9: The pseudo-code to split the range of log-likelihood into buckets/bins.

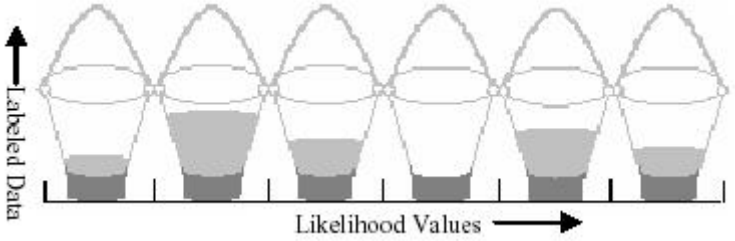


Figure 6.10: The bucketing approach to group data with similar likelihood values

produce similar log-likelihood values. The size of the bucket, θ, is a parameter of the model that is used to guide the rule extraction process. Figure 6.10 illustrates the bucketing process, and details of the algorithm are depicted in Fig. 6.9.

A bucket has a start point and an end point corresponding to the log-likelihood values. The start point corresponds to the immediate next log-likelihood value to the end point of the last bucket. The end point is the log-likelihood value that is at a θ distant from the start point. Figure 6.9 describes the bucket algorithm used to partition the data sequences. Figure 6.11 and 6.12 show bucketed data patterns grouped in two buckets, where the size

of the bucket is 0.5 (i.e., $\theta = 0.5$). The log-likelihood range for the bucket in Fig. 6.11 is from -3.5159 to -4.0159, whereas that for the bucket in Fig. 6.12 ranges from -4.5159 to -5.0159. To begin with the generating of fuzzy rules, the buckets are arranged in the ascending order of their range of log-likelihood values.

6.2.3 The Fuzzy Model

Once buckets have been created and the dataset partitioning operation is complete, the fuzzy rule extraction begins. Initially, rules are generated from the dataset. The rule generation process is briefly described below, followed by insight into the proposed top-down tree algorithm for the rule extraction.

For the consequent part of the fuzzy rules, we used the TS [186] model in which the output is calculated as the linear combination of weighted input variables. As described earlier in this chapter, the inference in the TS model is made from the fuzzy rule as follows:

if x is A_1 and y is B_1, then $z_1 = p_1x + q_1y + r_1$

where p_1, q_1 and r_1 are linear parameters.

Fuzzy Rule Extraction:

For the antecedent part of fuzzy rules, our approach chooses the Gaussian membership function. So, for k variables in a data vector/pattern, there exist k membership functions. For each of the input variables, we calculate the mean μ_{fuzzy} and the standard deviation σ_{fuzzy}. Then, the k^{th} (for the k^{th} input variable) membership function is:

$$M_{ik}(x_k) = e^{\frac{-1}{2}(\frac{x_k - \mu_{fuzzy_{ik}}}{\sigma_{fuzzy_{ik}}})^2} \tag{6.10}$$

where, M_{ik} = membership function for the rule i and data feature k,

$i_k = k^{\text{th}}$ input feature in the input space,

$\mu_{fuzzy_{ik}}$ = mean or the location of the k^{th} membership function in rule i, and

$\sigma_{fuzzy_{ik}}$ = the standard deviation or the steepness of the k^{th} membership function in rule i.

Following the standard representation of the fuzzy rule as described in Section 6.1.1, the generated i^{th} rule in linguistic form would be as:

if x_1 is M_{i1} and $\cdots$ and x_k is M_{ik} then (output) $p_{i1}x_1 + p_{i2}x_2 + ... + r_i$

($p_{i1}, p_{i2}, ..., r_i$ are linear parameters).

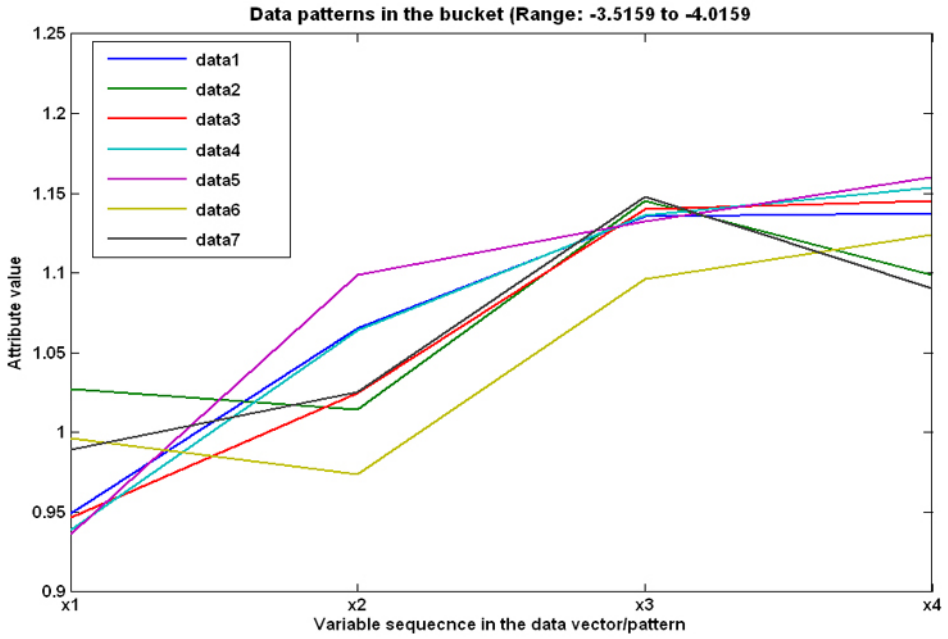


Figure 6.11: The bucketed data patterns in a specific bucket

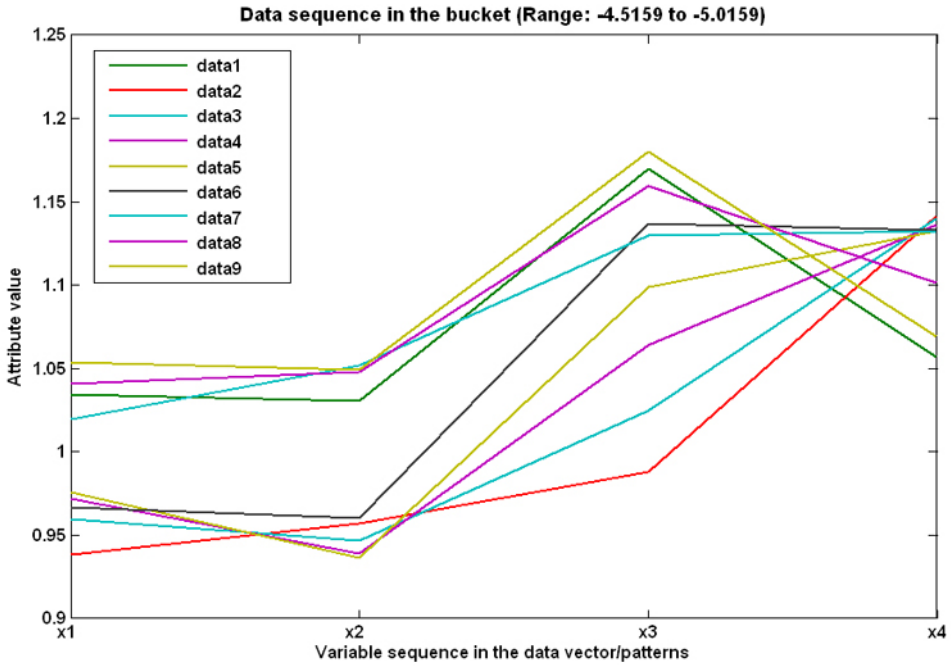


Figure 6.12: The bucketed data patterns in a different bucket than the bucket in Fig. 6.11

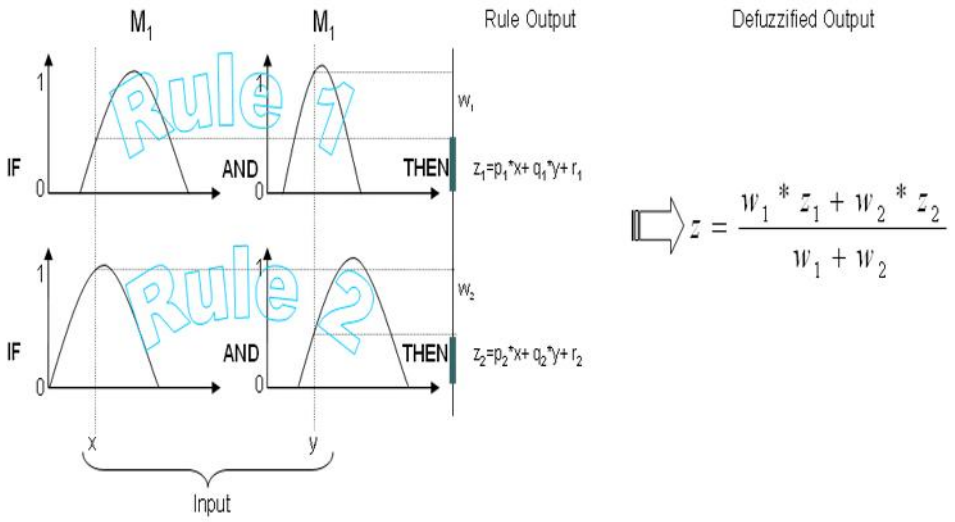


Figure 6.13: A graphical representation of the TS fuzzy rule (W_i represents the weights of each of the rule;typically set to 1)

A graphical representation of such two fuzzy rules along with the de-fuzzification process is shown in Fig. 6.13.

The top-down tree approach to generate the fuzzy rules

To begin, we create just one fuzzy rule to represent the entire input space of the training dataset. At this point, all log-likelihood values contained in the individual buckets may be perceived as belonging to one global bucket. This rule is then used to predict the variable of interest. The level of the rule's efficacy is determined based on the prediction error for the training dataset, which is in turn calculated in terms of Mean Squared Error (MSE).

If the prediction error for the training dataset is less than or equals a threshold value ξ, the algorithm is terminated and no further rules are extracted. On the other hand, if the prediction error is greater than ξ, then the input space is split into two parts with the help of the buckets produced in Section 6.2.2 (see Fig. 6.9). The splitting of the input space is achieved by dividing the total buckets into two equal parts. Data in the respective parts constitute the splitted input space. Each splitted partition has individual rules created for it. Finally, the total number of rules is increased by one. The prediction error for the training dataset is recalculated using the extracted rule set. Should the error threshold ξ not be reached then the buckets containing the datasets responsible for the left part of the rule are divided into two rules, and the process is iterated. Again, if the error threshold

ξ is still not met, the right part of the rule is partitioned and the process is undertaken again. This cycle continues until either the error threshold ξ is met or the number of rules equals the number of buckets.

Figure 6.15 shows the process of this partitioning while the algorithm in Fig. 6.14 describes the process for the rule extraction. Figure 6.16 and 6.18 shows close relationships among the divided data features for the Mackey-Glass time series data (details about this benchmark data are presented in Section 6.3) which reveals the strength of the data partitioning technique based on the divide and conquer/top-down tree approach of the bucketed/binned HMM-loglikelihood values. Figure 6.17 and 6.19 show rules generated from the divided parts respectively. It should be noted that the 2nd divided part in Fig. 6.17(a) has been further divided into two parts and hence three group of datasets are obtained in Fig. 6.19(a), which leads to generating three fuzzy rules. This rule extraction method follows the algorithm in Fig. 6.14.

6.2.4 Optimization of extracted Fuzzy rules

To optimize parameters for the extracted fuzzy rules, a gradient descent algorithm is employed. Our objective is to minimize the MSE for the training dataset. In the TS fuzzy model, every dataset has two parameters that need to be optimized. The first one is the nonlinear (premise) parameter, and the second one is the set of linear (consequence) parameters. In our model, we used the optimization technique ANFIS presented by Jang [27], where a gradient descent method along with the Least Squared Error (LSE) estimate is applied.

Error_Threshold=ξ; (ξ= desired error value)

Extract only one rule using the whole training dataset

error=calculate_error(data, rules);

if(error > error_threshold ξ)

divide the total number of buckets into two equal part and extract rules for each of the part

error=calculate_error(data, rules);

left_flag=true

right_flag=false

end if

while(error > error_threshold ξ)

if (left_flag=true)

divide the left part into two equal part using bucket number and

extract rules for each of the part

error=calculate_error(data, rules);

left_flag=false

right_flag=true

else

divide the right part into two equal part using bucket number

extract rules for each of the part

error=calculate_error(data, rules);

left_flag=true

right_flag=false

end if

end while

return rules

---

function error=calculate_error(data, rule)

simulate result using extracted rule

error=mse(produced_output-actual_output)

return error;

Figure 6.14: The algorithm for rule extraction using buckets.

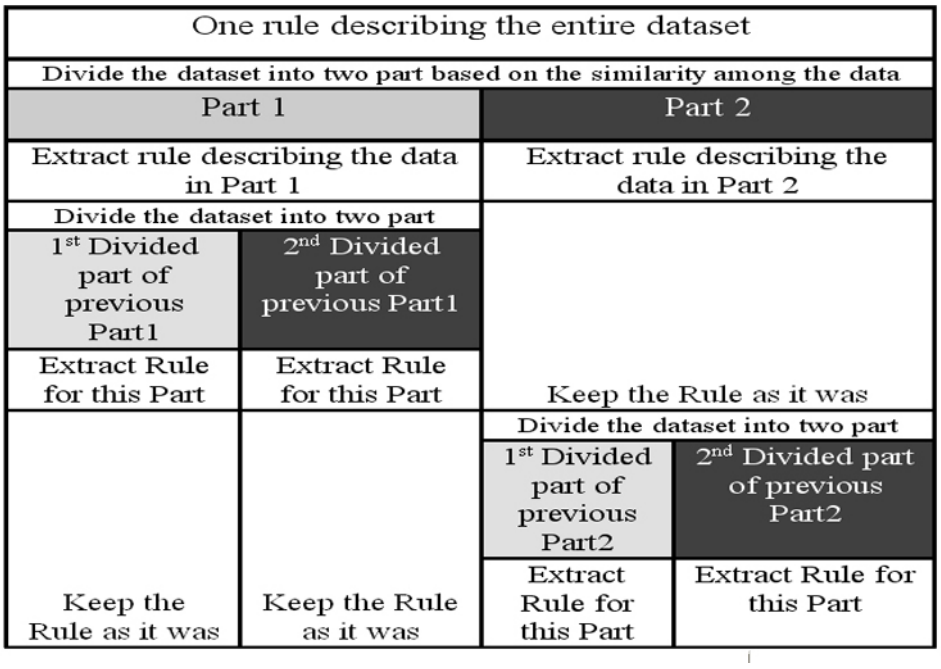


Figure 6.15: The top-down fuzzy rule creation

6.3 Experiment and Result

To investigate the performance of the HMM-Fuzzy model, we used several time series datasets. To start with, we used the well known Mackey-Glass dataset. We also evaluated our HMM-Fuzzy model by generating a forecast for the six stock prices that we used in Chapter 4. Descriptions and details of the experimental set up, datasets used, and the results obtained follow this paragraph.

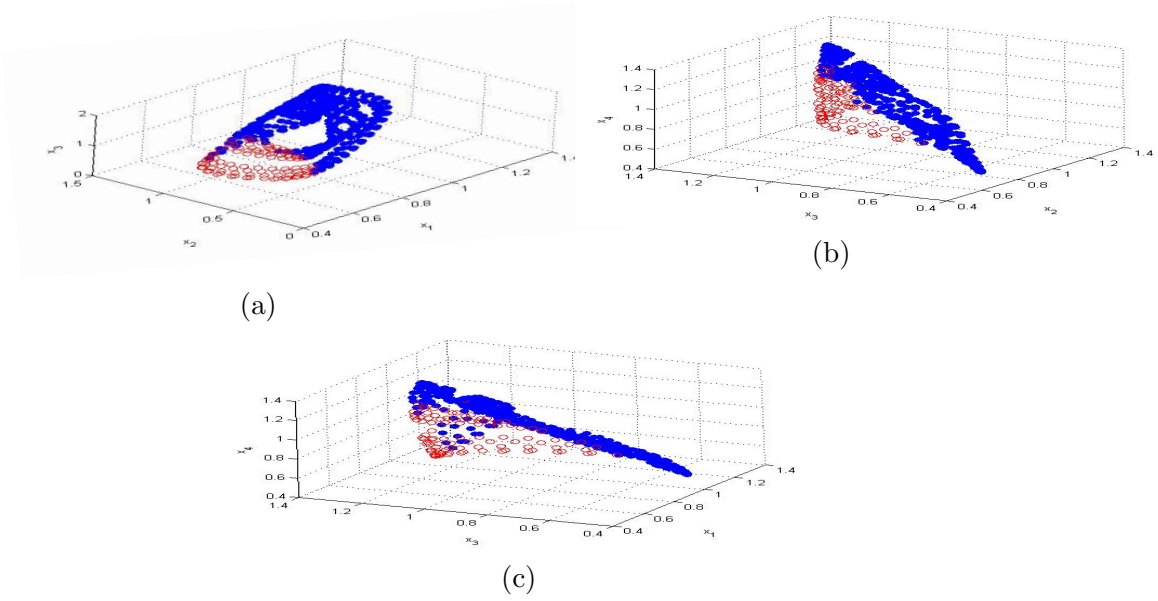
6.3.1 Time series prediction: Mackey-Glass Data

Data set

The well known Mackey-Glass chaotic time series data, originally presented by Mackey and Glass [62], was generated with the following differential equation τ:

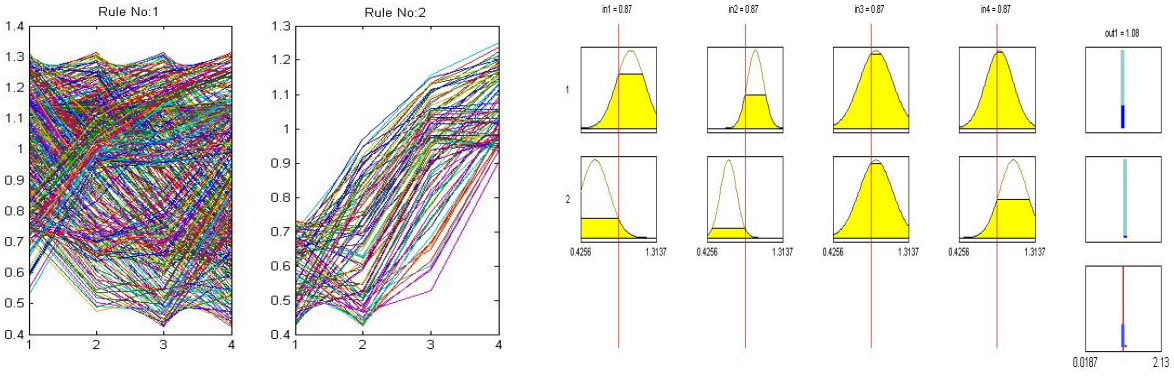
$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{1}0(t-\tau)} - 0.1x(t) \tag{6.11}$$

In the above equation, the values of $x(t-18), x(t-12), x(t-6)$ and $x(t)$ were used as predictor variables, while the value of $x(t+6)$ was taken to be the dependent variable.



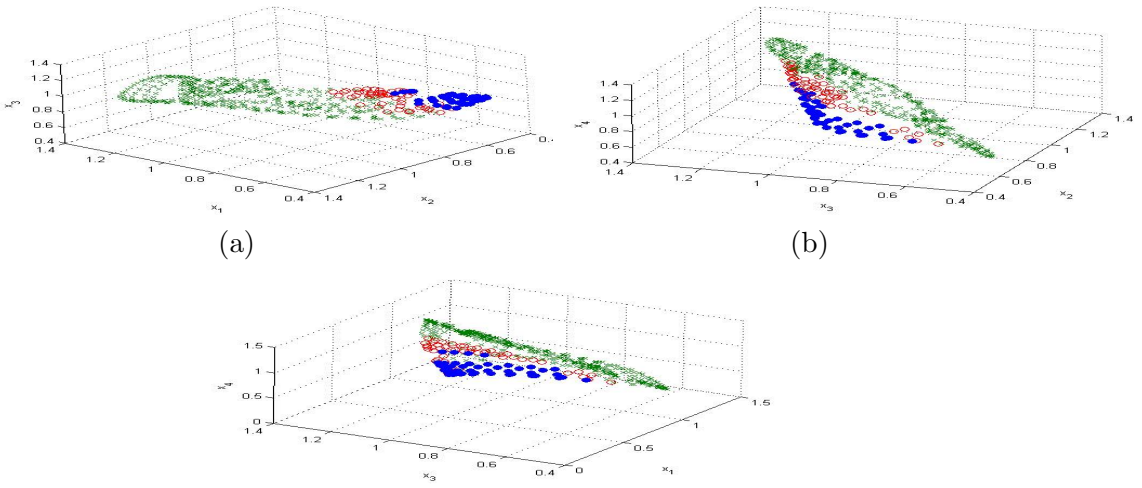
(a) (b) (c)

Figure 6.16: Using Mackey-Glass time series data: (a) The scatter-plot relating x_1, x_2, x_3 for the divided data into two parts. (b) The scatter-plot relating x_2, x_3, x_4 for the divided data into two parts. (c) The scatter-plot relating x_1, x_2, x_4 for the divided data into two parts.



(a) (b)

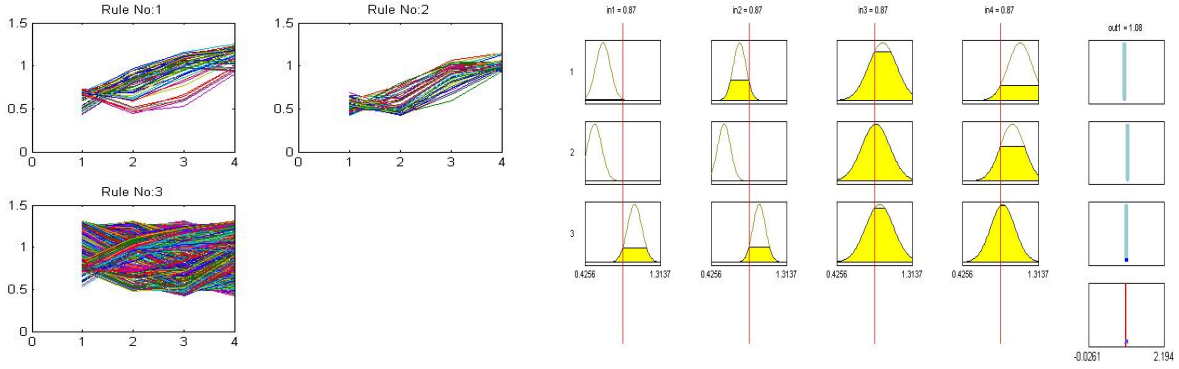
Figure 6.17: Using Mackey-Glass time series data: (a) The two parts of the dataset divided for generating two fuzzy rules. (b) Generated two fuzzy rules from the two divided part



(a) (b)

(c)

Figure 6.18: Using Mackey-Glass time series data: (a) The scatter-plot relating x_1, x_2, x_3 for the divided data into three parts. (b) The scatter-plot relating x_2, x_3, x_4 for the divided data into three parts. (c) The scatter-plot relating x_1, x_2, x_4 for the divided data into three parts.



(a) (b)

Figure 6.19: Using Mackey-Glass time series data: (a) The three parts of the dataset divided for generating three fuzzy rules. (b) Generated three fuzzy rules from the three divided parts

The Runge-Kutta method was used to generate 1000 data series; the initial conditions were $x(0) = 1.2$ and $x(t - \tau) = 0$ for $0 \leq t \leq \tau$. The first 500 data in the series were applied to the training process, whereas the remaining 500 data in the series were applied as test data.

HMM-Fuzzy Model parameters

For the considered model, the bucket size was set to $\theta = 0.5$. The actual bucketing was performed with the HMM log-likelihood values and the allowable margin of error was set to 0.000002. In order that the extracted rules be optimized, five hundred epochs were chosen during the execution of the gradient descent algorithm.

The number of states in the HMM was empirically chosen as the number of input features available to the dataset. Initial values of the transition probability matrix and the prior probability matrix were chosen at random for the HMM. Time series datasets are continuous, and hence the observation emission probability matrix of the HMM is chosen to follow normal distributions.

Rule creation

Figure 6.20 shows a plot of the error convergence rate in terms of MSE during the training period. After a certain point (say, rule 8 or 10) the value of the MSE increases as the number of fuzzy rules increases. But after that point, the MSE decreases with an increasing number of rules. We stop the process once the MSE meets the desired MSE (= 0.000002) The plot in Fig. 6.20 illustrates how the value of the MSE increases and then decreases in the 8–16 rule range, and after this point the value of the MSE approaches zero.

Figure 6.21 shows the final divided 32 parts of the training dataset obtained using the method described in this chapter. The similarities among the data patterns in each of the divided dataset are apparent from the figure. The respective generated fuzzy rules for each of the divided data sets are shown in Fig. 6.22.

Results

Figure 6.23 plots the prediction accuracy of the hybrid HMM-Fuzzy model. When inspected, the plot reveals that it is hard to differentiate between the predicted values and the actually occurring values. The inference would be that the model's ability to predict

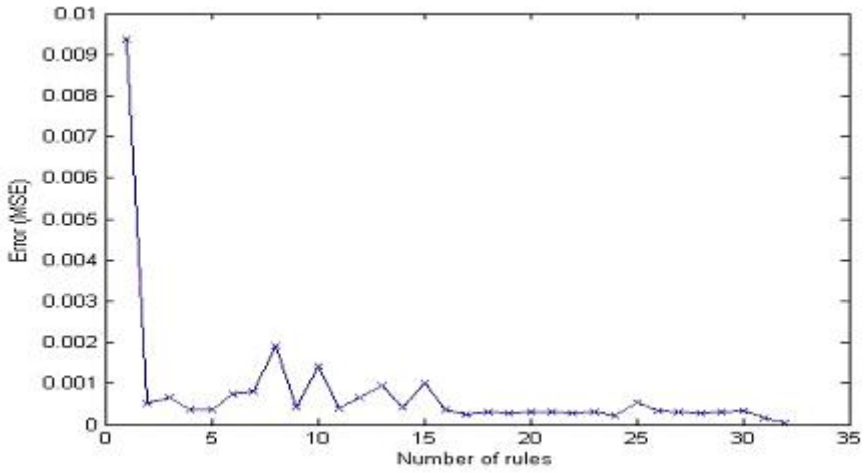


Figure 6.20: Training MSE convergence curve during the rule creation for Mackey-Glass time series data

the chaotic time series is quite good. The residual graph in Fig. 6.24 illustrates the negligible variance between actual values and predicted values. Table 6.1 compares our approach with various time series prediction techniques: e.g., ANFIS [27] and data driven linguistic modeling using relational fuzzy rules [131], for the same dataset. The term of comparison is the normalized root mean squared error (NRMSE) that is obtained by dividing the root mean squared error (RMSE) by the standard deviation of the test data.

6.3.2 Stock Market Forecasting

Dataset

To evaluate the performance of the HMM-Fuzzy model, we considered the daily prices of the six stocks that were used in Chapter 4. For the sake of consistency, we have used the same training and test dataset for all stocks considered here.

Experimental Setup

For each of the stocks, the bucket size was set to be $\theta = 0.5$; the actual bucketing was performed with log-likelihood values and the allowable margin of error was set in the range of 0.02 to 0.002. Before generating the rules, datasets were normalized between $+1$ to -1. In order that the extracted rules be optimized, a maximum five hundred epochs were chosen during the execution of the gradient descent algorithm.

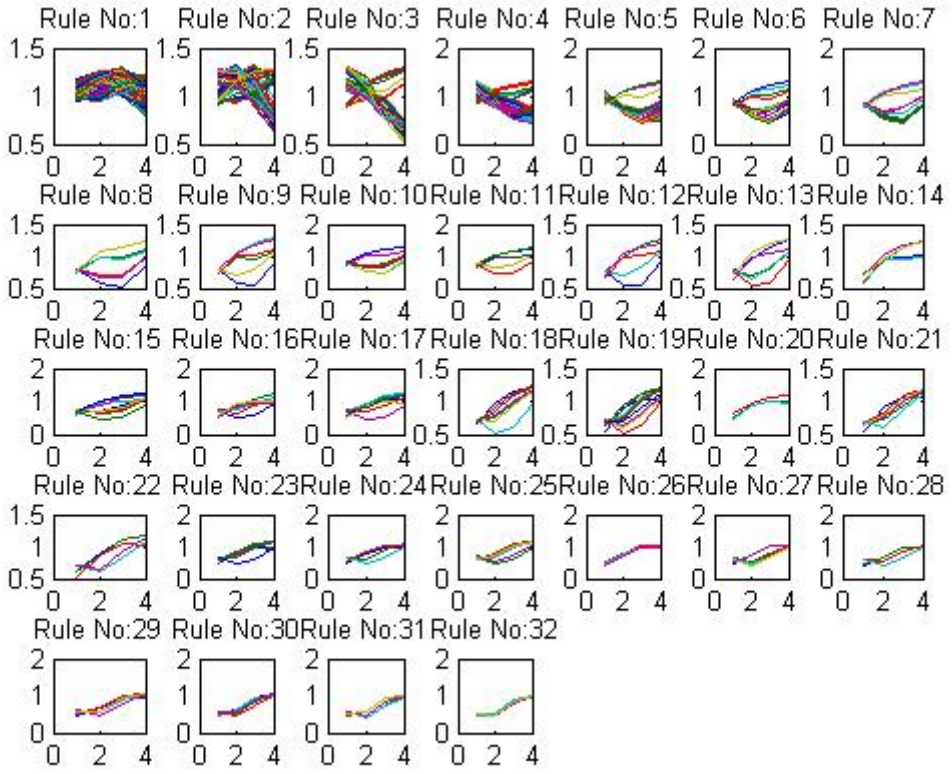


Figure 6.21: The divided data patterns of the training Mackey-Glass time series dataset

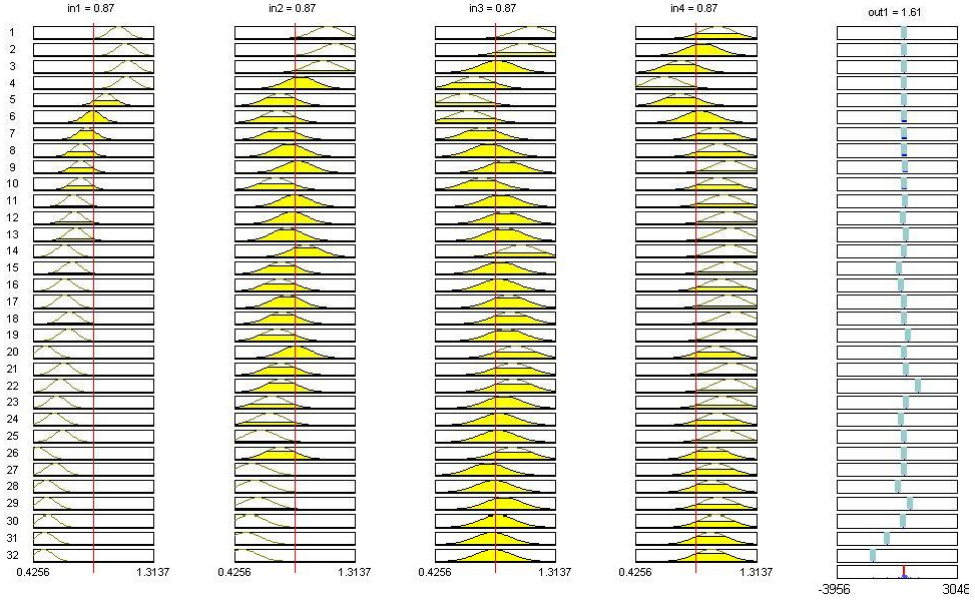


Figure 6.22: Fuzzy rules generated using each of the divided data patterns in Fig. 6.21

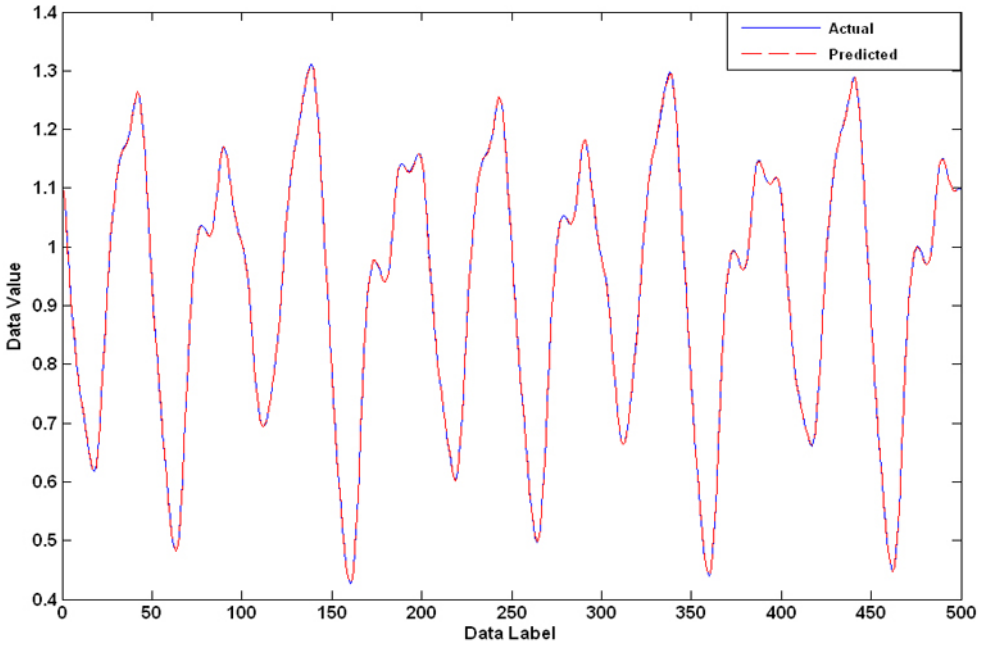


Figure 6.23: The Mackey-Glass time series and predicted value

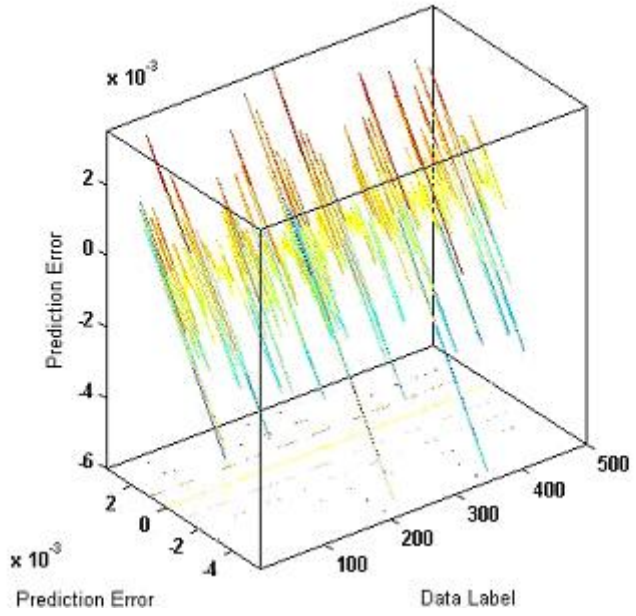


Figure 6.24: The absolute error (in prediction) for each of the time instance (Mackey-Glass time series data

Table 6.1: Prediction error for the Mackey-Glass dataset

Method	Prediction error (NRMSE)
HMM-Fuzzy Model	0.0055
GEFREX [190]	0.0061
ANFIS [27]	0.0074
Data-Driven Linguistic Modeling Using Relational Fuzzy Rules [131]	0.0090
Rule extraction using Subtractive clustering followed by optimization using Gradient Method(500 epochs) [107]	0.0084
SuPFuNIS(15 rules) [191]	0.0140
DENFIS(OFFLINE) [192]	0.0160
EPNet [99]	0.0200
TDDFN [193]	0.0250
GFPE [194]	0.0260
6th order polynomial [194]	0.0400
Cascade Correlation-NN model [194]	0.0600
Auto-Regressive Model [194]	0.1900
Linear Predictive Model [194]	0.5500

Following the previous experimental setup, the number of states in HMM for the stocks was empirically chosen as the number of input features available to the dataset. The initial parameter values of the HMM were chosen following the same steps as the previous experiment described in Subsection 6.3.1.

Results

Table 6.2 reports the forecast performance of the developed HMM-Fuzzy model for the six stocks in terms of the forecast error using the MAPE metric. To investigate the performance improvement, we also present the forecast performance obtained using the fusion HMM-based forecasting model developed in the previous chapter.

6.4 Summary

This chapter introduces an efficient data driven fuzzy model generation approach, namely the HMM-Fuzzy model. The model combines HMM's data partitioning approach with FL generation for the prediction of multivariate time series data. The HMM-based data partitioning technique includes the relationship between data features (because it uses

Table 6.2: HMM-Fuzzy model performance. The Mean Absolute Percentage Error (MAPE) in the forecast for unseen test dataset

Stock Name	Fusion HMM-ANN-GA with Weighted Average (MAPE)	HMM-Fuzzy (MAPE)	ARIMA (MAPE)
British Airlines	1.646	1.529	1.573
Delta Airlines	5.529	4.535	5.294
Ryanair Airlines	1.377	1.356	1.504
Apple Computer Inc.	1.925	1.796	1.801
IBM Corporation	0.849	0.779	0.972
Dell Inc.	0.699	0.405	0.660

the Markov process, where it is assumed that the current event/feature depends on the immediate past event/feature) in the data vectors. Hence, the generated fuzzy model uses a suitable number of fuzzy rules and produces a better performance. Parameters of the generated fuzzy rules are further fine tuned using a gradient descent algorithm.

We evaluated the performance of the model using the Mackey-Glass time series, which is one of the well-known benchmark data with non-linear and non-stationary characteristics. We compared our results with those of other techniques reported in contemporary studies. The results reported here are very encouraging. We have clearly demonstrated that the HMM-based fuzzy model is capable of predicting the non-linear and non-stationary time series data accurately. It should also be noted that at times the model may become complex, as the number of rules required to achieve a user defined minimum MSE increases. The work reported in this research demonstrates that hybridizing the strengths of HMM data partitioning for generating FL has the potential to achieve better performance when applied to the prediction of non-linear time series data.

The top-down tree approach (described in Section 6.2.3) is used to identify a 'reasonable' number of rules for a specific problem dataset in order that it meet the user defined MSE for the training dataset. The plot in Fig. 6.20 shows how prediction accuracy (in terms of MSE) for the Mackey-Glass training dataset converges as the number of rules increases up to a certain point. The cue to stop the rule extraction process, therefore, is the attainment of the minimal MSE for the training dataset.

Figure 6.23 shows the prediction accuracy of the hybrid model for the Mackey-Glass dataset. The actual sequence and the predicted sequence are very close. The maximum difference between the actual value and the predicted value is less than 2×10^{-3}, and

this is illustrated in Fig. 6.24. Table 6.1 compares the performance of the hybrid model with other existing models for the same dataset. It is evident that our model has outperformed a number of well-known alternate methods. The performance of our model prior to optimization is still better than that of certain existing time series prediction methods.

We also tested our model by preparing forecasts for the financial time series data of six stock prices. We notice that our model produces improved forecasts for the stock datasets (see Tab. 6.2 that are non-linear (see Appendix) but have weak non-stationarity (as reflected by the ACF and PACF plot shown in the Appendix). The HMM-Fuzzy model can generate a suitable fuzzy model to demonstrate the datasets and hence an improved forecasting performance is achieved for the financial time series with non-stationarity and non-linearity.

To evaluate the robustness of the developed HMM-Fuzzy approach, we applied it to the classification of a number of real world datasets. The model performed equally well in classifying the Wisconsin breast cancer dataset (see Appendix Tab. A.1) and the gait related problem (see Appendix Tab. A.2). The proposed HMM-fuzzy model not only performs better but also poses lower complexity levels by using a reduced number of rules.

We notice that tge HMM-Fuzzy model can achieve better forecast performance by generating an acceptable number of rules within the range of 1 to 10 for non-linear time series data where week non-stationarity exists. However, while predicting a time series data with strong non-stationarity (as reflected by ACF and PACF plots in the Appendix) as in the Mackey-Glass data, despite the better prediction performance, the generated fuzzy model becomes complex as the number of rules is very high, riching upto 32 (see Fig. 6.22) in our experiment. This is due to the inappropriate choice of the desired MSE, which has a potential role in keeping the number of fuzzy rules to a minimum. In the following chapter we shall introduce a hybrid HMM-multi-objective EA approach for building the fuzzy model for such a complex non-linear time series data prediction, where a range of appropriate MSE will be obtained with the incorporation of EA.

Chapter 7

HMM-Fuzzy with EA for time series forecasting

In this chapter, we develop and investigate a novel HMM-Fuzzy model with EA, by combining the HMM-Fuzzy model developed in the previous chapter with EA to predict a non-linear time series. In the previous chapter, we introduced the base model, where a single HMM was used to sort the data vectors (k-dimensional data) based on similar log-likelihood values. The rationale on which our model is based is that the underlying Markov process, which considers the relationship among the data features/observations, will help to generate appropriate fuzzy rules. Here, the multi-dimensional data are projected onto one-dimensional values represented by the HMM log-likelihood values for each of the k-dimensional data vectors. A divide and conquer approach is then used to generate a number of possible fuzzy rules to meet a predefined threshold level for the prediction accuracy for the training dataset. We empirically show that the HMM-Fuzzy model can prepare a better forecast for the non-linear time series dataset with either weak or strong non-stationarity. However we have also identified that the HMM-Fuzzy model generates a relatively large number of rules (for the Mackey-Glass dataset it generated 32 rules) for the dataset with strong non-stationarity. To improve the robustness of the model, we employ a multi-objective EA to find a range of compromise solutions between the optimal number of rules and prediction accuracy. This approach is proposed to counteract the 'over-fitting problem' associated with keeping a large number of fuzzy rules. In this chapter we have evaluated our proposed model using a number of known benchmark non-linear and non-stationary time series data including the Mackey-Glass data. The findings of this chapter have been submitted to appear in the journal [195].

The remainder of the chapter is organized as follows. In Section 7.1 we describe our HMM-Fuzzy model with EA, and place particular attention on describing the process of data partitioning using the HMM-based data pattern sorting approach and the process of combining multiobjective EA with the HMM-Fuzzy model. Section 7.2 presents experimental results on a number of non-linear and non-stationary benchmark data sets time series data. A comparison of the performance of the hyrbid fuzzy model with other time series prediction techniques is presented in this section. Finally, in Section 7.3 we conclude this chapter with a discussion of the model and its implication.

7.1 HMM-Fuzzy model with EA

We introduce a novel hybrid model by combining the HMM-Fuzzy model[1] with EA. Here, in our model, an HMM is used to sort the data vectors in the multivariate dataset and divide the input space into a number of subspaces to form fuzzy rules. The multiobjective EA is applied to obtain a range of solutions which 'trades-off' between the number of generated fuzzy rules and the prediction accuracy.

The model consists of four phases:

- **Phase 1**: The HMM is used to partition the input dataset based on the ordering of the calculated HMM-loglikelihood values.
- **Phase 2**: An iterative top-down(divide and conquer) algorithm is used to generate the minimum number of fuzzy rules to meet the pre-defined mean square error(MSE) for the training dataset.
- **Phase 3**: A gradient descent method is applied to fine tune the obtained model parameters.
- **Phase 4**: We refer to the model developed in phases 1 to 3 as the *base model*. A multiobjective EA is applied to the base model to choose a suitable MSE for the dataset.

It is noted that, before using the HMM to partition the input data vectors, the HMM is trained using the Baum-Welch algorithm described in Chapter 3 and the available training data vectors. Fig. 7.1 shows the overall hybrid HMM-fuzzy with EA model.

[1] A prototype of the model was initially developed in our previous chapter [196]

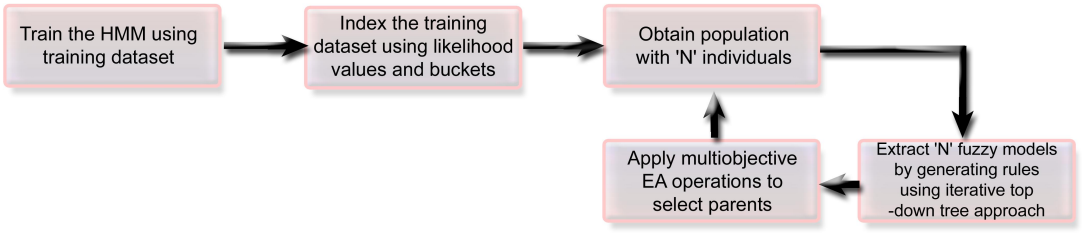


Figure 7.1: The hybrid HMM-EA model for generating fuzzy rules

Data vector 1 :	x_{11}	x_{21}	x_{31}	...	x_{k1}
Data vector 2 :	x_{12}	x_{22}	x_{32}	...	x_{k2}
Data vector 3 :	x_{13}	x_{23}	x_{33}	...	x_{k3}
.....	...	...	...	...	...
.....	...	...	...	...	...
Data vector n :	x_{1n}	x_{2n}	x_{3n}	...	x_{kn}

Figure 7.2: The set of data vectors formed by distinct variables to be used as observation of an HMM

7.1.1 Sort the training dataset

To partition the input dataspace, we first sort the data vectors/patterns using a single HMM, based on the similarities among the patterns. The initial HMM is built by using random parameter values. The approach of sorting data vectors using a number buckets, as is described in the previous chapter, is followed in this phase of the model. However, a detailed analysis of the effect of ordering the data variables in the training dataset on the data-pattern sorting approach is discussed below.

HMM as a data-pattern sorting tool

Typically, an HMM is used as a forecasting tool where the main requirement is that the data to be processed are sequential. In our model, a single HMM is used to sort the available training data based on the HMM-loglikelihood value. Here, the HMM is used as a pattern matching tool only, where no time dependency is assumed among the data variables (features). Each data vector that is fed into the HMM are formed using a number of distinct variables (see Fig. 7.2). Given the trained HMM, the probability of generating a data pattern is calculated using the forward algorithm [147] presented in Chapter 3.

In Chapter 3 and onward, we have used a single HMM in such an approach to sort and group the similar data vectors in the data set. The approach for sorting data vectors

$$\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ . \\ x_k \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & \dots & x_k \end{array} \\ \begin{bmatrix} \sigma_{1,1} & \sigma_{1,2} & \sigma_{1,3} & \dots & \sigma_{1,k} \\ \sigma_{2,1} & \sigma_{2,2} & \sigma_{2,3} & \dots & \sigma_{2,k} \\ \sigma_{3,1} & \sigma_{3,2} & \sigma_{3,3} & \dots & \sigma_{3,k} \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ \sigma_{k,1} & \sigma_{k,2} & \sigma_{k,3} & \dots & \sigma_{k,k} \end{bmatrix} \end{array}$$

Figure 7.3: Covariance Matrix for a dataset containing $k - dimensional$ 'k' data.

$$\begin{array}{c} \\ x_2 \\ x_1 \\ x_3 \\ . \\ . \\ . \\ . \\ x_k \end{array} \begin{array}{c} \begin{array}{ccccc} x_2 & x_1 & x_3 & \dots & x_k \end{array} \\ \begin{bmatrix} \sigma_{2,2} & \sigma_{2,1} & \sigma_{2,3} & \dots & \sigma_{2,k} \\ \sigma_{1,2} & \sigma_{1,1} & \sigma_{1,3} & \dots & \sigma_{1,k} \\ \sigma_{3,2} & \sigma_{3,1} & \sigma_{3,3} & \dots & \sigma_{3,k} \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ \sigma_{k,2} & \sigma_{k,1} & \sigma_{k,3} & \dots & \sigma_{k,k} \end{bmatrix} \end{array}$$

Figure 7.4: Covariance Matrix with changing the ordering of variables for the same dataset used in Fig. 7.3

based on HMM-loglikelihood values is valid for a multivariate data vector, such that each data vector can be represented as a data pattern/sequence. The question is, what if we change the ordering of the variables in the data vectors? The following discussion explains the effect of the change in ordering of the variables.

For a multivariate dataset consisting of k variables, the covariance matrix would be as shown in Fig. 7.3. Each of the elements of the covariance matrix represents the relationship between two variables along the vertical axis and along the horizontal axis. The relationships between the variables would be unchanged with the changed ordering of representation of the variables. For instance, if we change the ordering of the variables in Fig. 7.3 as $\langle x_2, x_1, x_3, ..., x_k \rangle$ instead of $\langle x_1, x_2, x_3, ..., x_k \rangle$ the relationships between the variables would be the same, except that the look of the covariance matrix will be changed according to the change of representation of the ordering of the variables (see Fig. 7.4). The performance of the HMM used for sorting such data vectors is not likely to be affected by the ordering of variables presented to it. Here the parameters of the HMM would be re-estimated during the training period, taking into consideration the covariance matrix of the dataset and the changes in data values. But caution should be taken such that the

ordering of the variables in each data vector remains consistent throughout the process of the training and testing/practical application.

7.1.2 Fuzzy rule generation

In this phase of the model, we divide the dataset into a number of bins/buckets based on the HMM-loglikelihood values, which follows the rule generation step described in the previous chapter. Initially a rule is generated assuming the dataset is undivided. In the process of rule generation, we calculate the mean μ_{x_i} and standard deviation σ_{x_i} to define the membership function for each feature x_i in the dataset as follows:

$$M_{x_i} = e^{-\frac{1}{2}(\frac{x_i - \mu_{x_i}}{\sigma_{x_i}})^2} \tag{7.1}$$

The prediction error for the training data vectors is calculated using the generated fuzzy rule. A Mean Squared Error (MSE) is used to quantify the performance of the developed model for the training dataset. If the error is greater than a predefined threshold error (ξ_{MSE}) for the training dataset, the input space is divided into two subspaces to generate two fuzzy rules. This is achieved by dividing the total bins of data vectors into two parts. Data vectors contained in each divided part are used to form fuzzy rules. The training error is calculated and compared with the (ξ_{MSE}). The divide and conquer method continues until the threshold error (ξ_{MSE}) is met by the generated fuzzy model.

7.1.3 Parameter fine tuning

The parameters of the generated fuzzy rules are further fine tuned using a gradient descent algorithm and training dataset. At this stage, the mean and standard deviation for each of the membership functions of all fuzzy rules is fixed more precisely so that it can predict with better accuracy. We follow the gradient descent methodology as in ANFIS [27].

7.1.4 Multiobjective EA applied to the HMM-Fuzzy model

The base HMM-Fuzzy model described in Chapter 6 uses an iterative divide and conquer method to divide the input data-space into a number of data subspaces (cf. Section 7.1.2). Each subspace is used to generate a fuzzy rule. We use the desired MSE for the training dataset given the generated fuzzy rules as the stopping criteria for the iterative process. This approach may suffer from the 'over fitting problem' when the desired MSE is a very

small number. It is possible that the maximum number of rules are generated for the given MSE. Once the fuzzy model is built, it may not perform well for the unknown test dataset because of the high number of fuzzy rules. A good/reasonable desired MSE could solve the problem of generating the rules up to a reasonable number and solve the over fitting problem as well.

Let us consider the training data set $X = \{\hat{X}_i = \langle x_1, x_2, ..., x_k \rangle\}$. The single data vector $\hat{X}_i$ is an element of the space $R^n \times R$ for each i, $1 \leq i \leq n$ and n =total number of data vectors in the dataset. In the HMM-Fuzzy model the HMM maps the training dataset X to the space of fuzzy systems based on the similarity among the data vectors and meeting the predefined desired MSE, ξ. For a given problem, 10 rules are used which produce a training MSE 0.023 and produce a test MSE 0.75. If the number of rules increases, the training MSE becomes 0.012 and the test MSE becomes 0.74. A further increase in the rule number produces training MSE 0.001 and test MSE 1.33. On the other hand, if the rule decreases the training MSE becomes 0.053 and the test MSE becomes 0.98. Hence, a suitable desired training MSE, ξ can produce a compact fuzzy model along with improved accuracy for the unseen test dataset. Therefore, the problem of generating an efficient and compact fuzzy model can be expressed as a multiobjective optimization problem, where there is always a trade-off between the generated number of fuzzy rules and the MSE. In this case, the two objectives are:

1. Prediction/classification accuracy – high values required, but the model should not suffer from the over fitting problem.

2. Minimum number of fuzzy rules – a compact fuzzy model is required

In this chapter we have used a multiobjective EA [197, 198] to find a range of alternative solutions for the given problem. The EA is used to generate a diverse set of points distributed along the non-dominated (or Pareto optimal) front. These solutions are optimal in the wider sense, in that no other solution in the search space is superior to them when all abovementioned objectives are considered. Figure 7.5 shows one such range of optimal solutions obtained using the multiobjective EA with the two abovementioned objectives.

An outline of the hybrid process is presented below:

1. Divide the training dataset into two parts: training and validation.

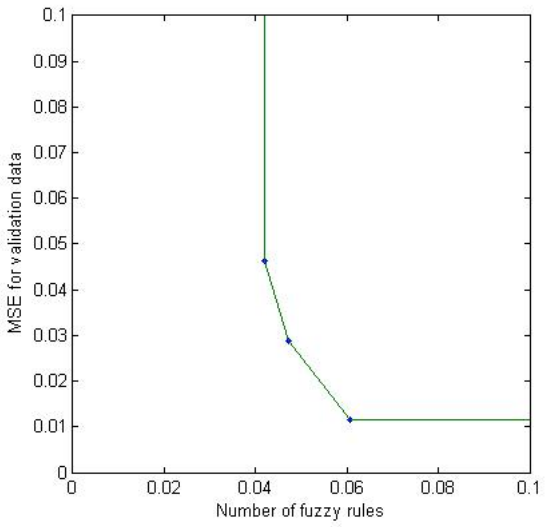


Figure 7.5: The range of optimal solution obtained using Multiobjective EA. Here, we scale the two cost functions: (1) accuracy (MSE) and (2) Fuzzy rules in the range of 0 to 0.1.

2. Generate the initial population P with N individuals (Each individual contains the desired MSE for the validation dataset)

3. Generate N fuzzy models using our HMM-Fuzzy approach and training dataset, then calculate the corresponding MSE values for each individual in the population.

4. Sort the N fuzzy models according to MSE and the number of fuzzy rules generated for each of the N models. That is, use a non-dominated sort based on the given objectives.

5. Select individuals from the non-dominated set as parents, then apply the standard genetic operators (crossover and mutation).

6. Repeat steps 3 and 5 until a predefined time limit T_{stop}

7. Finally, consider those individuals on the Pareto-optimal front as the range of optimal solutions for the problem.

7.2 Experimental Results

To investigate the performance of our proposed hybrid HMM-Fuzzy with EA model, a number of experiments were conducted. Three benchmark time series data sets, the Mackey-

Table 7.1: Comparison of prediction accuracies for the Mackey-Glass data

	No. of Rules	NRMSE
HMM-Fuzzy with EA	**8**	**0.0054**
	13	**0.0048**
HMM-Fuzzy [196]	31	0.0055
ANFIS [27]	16	0.0074
Data-Driven Linguistic Modeling Using Relational Fuzzy Rules [131]	6	0.0090
SuPFuNIS [191]	15	0.0140
DENFIS (OFFLINE) [192]	-	0.0160
EPNet [99]	-	0.0200
Rule extraction using Subtractive clustering followed by optimization using Gradient Descent Method [107]	16	0.0084

Glass dataset [62], Box-Jenkins gas furnace dataset [58] and Gas-Mileage dataset [199] were used in the initial experiments.

7.2.1 Mackey-Glass Data

The Mackey-Glass dataset is a well-known benchmark time series data introduced by Mackey and Glass [62]. The detail of this dataset is provided in Chapter 5. In that chapter, we tested our base HMM-Fuzzy model using this dataset, but we did not test the model during the training time using the validation data. However, in this experiment, the first 500 data vectors of the 1000 generated data are used for training the forecasting method and the remaining 500 data are kept to test the method. From the training dataset, the first 450 data were used for training the model and the next 50 were used for validation.

Table 7.1 shows the results of some well known prediction methods tested on Mackey-Glass data. Using the base model, 31 suitable rules were found corresponding to a 0.0055 NRMSE (Normalized Root Mean Squared Error) accuracy level for the test data. When the Multiobjective EA was used with validation data, significant improvement in performance was noted. Table 7.1 shows improvement in both the accuracy and complexity of results.

To investigate what effect the ordering of the variables has on the HMM-Fuzzy model, we tested the model by interchanging the first two predictor variables in the Mackey-Glass dataset. We achieved an NRMSE up to 0.0049 using the interchanged sequence of

variables. This was achieved using 15 rules. The average error (for at least 5 runs) for this experiment is achieved $0.0062 \overset{+}{-} 0.0010$ using $12.40 \overset{+}{-} 1.5166$ fuzzy rules, while the average error with keeping the order of variables unchanged is achieved as $0.0057 \overset{+}{-} 0.00058$ using $12.80 \overset{+}{-} 0.8367$ rules. This reveals that there is no effect from the ordering of variables in the data vector on the performance of the model which we have discussed in Section 7.1.1.

7.2.2 Box-Jenkins Gas Furnace data

The Box-Jenkins gas furnace data [58] is one of the widely used benchmark data in testing the efficiency of a prediction method.

In this gas furnace, air and methane were combined in order to obtain a mixture of gases which contained CO_2 (Carbon dioxide). The methane gas feed rate forms the input series $u(k)$ and the gas outflow $y(k)$ is the output series, which is the concentration of CO_2 at the outlet. 296 successive pairs of observations were collected from the continuous records at 9-second intervals to form the entire gas-furnace dataset.

In this experiment we chose the following input variables to the fuzzy model to make it consistent with some of the existing study [131, 194]:

$u(k-1), u(k-2), u(k-3), y(k-1), y(k-2), y(k-3)$ for the output variable $y(k)$

To be consistent with other studies we trained and tested all the 296 data vectors and there was no validation dataset. Table 7.2 compares the performance of the proposed HMM-Fuzzy with EA model against some of the prediction methods reported in recent studies.

7.2.3 Automobile Gas Mileage data

The automobile gas mileage prediction, contains six numerical attributes that specify the automobile type - number of cylinders, displacement, horse power, weight, acceleration, model year - has been used to model the fuel consumption for a variety of vehicles. This is a nonlinear regression problem that relates fuel consumption with the respective vehicle characteristics. This dataset has been used in a number of studies [131, 189]. To maintain consistency with previous studies, in our experiment we also removed the six samples with missing values out of 398 samples. The remaining 392 samples were then divided randomly into two equal parts. In producing the fuel consumption value, we used the two

[2] Collected from [131]

Table 7.2: Comparison of prediction accuracies for the Box-Jenkins Gas Furnace data

Model	No. of Rules	MSE
HMM-Fuzzy with EA	**2**	**0.0454**
	4	**0.0371**
SI model with transformed inputs [200][2]	2	0.0480
Data-Driven Linguistic Modeling Using Relational Fuzzy Rules [131]	2	0.0550
SI model [194]	2	0.0550
Sugeno' model with OLS [201][2]	2	0.0660
Sugeno's model [202][2]	2	0.0680
Sugeno and Yasukaw [203]	6	0.1900
Chiu's fuzzy model [204]	3	0.0720
ARMA [58][2]	N/A	0.2020

Table 7.3: Comparison of prediction accuracies for the Automobile Gas Mileage data

Model	No. of Rules	RMSE
HMM-Fuzzy with EA	**2**	2.9700
	4	2.8000
Neuro Computing and Soft Computing approach [189]	4	2.9700
Data-Driven Linguistic Modeling Using Relational Fuzzy Rules [131]	3	2.8500

predictors, weight and model year, identified in [189]. Table 7.3 compares the performance of our model with other methods reported in previous studies on the same data.

7.3 Summary

In this chapter, we have developed an efficient data-driven fuzzy rule generation method for predicting future values of nonlinear time series. The proposed method is capable of generating a minimum number of fuzzy rules with improved prediction accuracy. Many of the existing data-driven fuzzy models are constrained by user defined parameters; e.g., number of clusters, radius of clusters or increased computational complexity due to generation of a large number of fuzzy rules. Our proposed method is not constrained by user parameters (e.g., number of fuzzy rules, number of clusters or radius of each clusters).

In an earlier work, Gaweda and Zurada [131] developed a model that required extensive analysis of the relationship between the available data and desired output for the time

series, prior to generating an efficient fuzzy model. In contrast, in our model, we use HMM to find similar data patterns from the training dataset. The HMM groups the similar data patterns based on the fluctuations among the predictor variables in the training dataset. A key advantage of our approach is that it is not necessary to analyze the training dataset before using the model. Once the data patterns have been grouped into a number of small buckets/bins, we proceed towards the generation of fuzzy rules. The divide and conquer technique for generation of fuzzy rules continues until the desired MSE is obtained for the validation dataset using the generated fuzzy model. This approach helps to keep the number of fuzzy rules as small as possible by meeting the desired MSE. This can lead to the generation of a large number of rules if we choose an impractical/unsuitable MSE for the dataset. It is possible for over fitting to occur – that is where a large number of rules are used in the fuzzy model. However, to overcome this problem we have used a multiobjective EA to find a range of comprise solutions between MSE and the number of fuzzy rules in the model.

By choosing a suitable MSE value, the prediction accuracy for the Mackey-Glass dataset improved by 12.73% compared with our base model described in Chapter 5. In addition, the number of required rules was significantly reduced, from 31 to 8. This performance is the best so far for the same dataset with a minimum number of fuzzy rules.

The efficiency of the HMM-Fuzzy model for time series prediction was further tested on a number of benchmark datasets. For the Box-Jenkins gas furnace dataset, by using 2 fuzzy rules as in another study [131,200], our model achieved at least 5% higher prediction accuracy. Furthermore, our model generated the optimal number of rules to be 4, which achieved at least a 22.7% better prediction.

On the automobile gas mileage prediction problem, our model performed better than other models using 4 rules, as shown in Tab. 7.3. For this dataset, the performance of our HMM-fuzzy model with EA was as good as that reported using a neuro-computing and soft computing approach [189]. When comparing the neuro-fuzzy model [189] with our model, we see that our model achieved this accuracy using only 2 rules, while the neuro-fuzzy model used 4 rules. Our model achieved the best performance (NRMSE=2.80) using 4 rules, which is better than the data-driven linguistic fuzzy model [131] (NRMSE=2.85).

In general, by choosing an appropriate MSE level, the base HMM-Fuzzy model has performed significantly better for all the time series benchmark datasets considered. This fine tuning may be attributed to the data pattern matching ability of the HMM. In most of

the data driven fuzzy rule generation methods, either the data sets are analyzed extensively to determine the appropriate number of fuzzy rules, or some existing clustering method is employed to divide the input data space. However, none of these methods takes into account the relationships among the data features which have a strong influence on the output feature. HMM data partitioning followed by fuzzy rule generation is able to obtain a minimum number of fuzzy rules to generate better time series predictions.

Chapter 8

Conclusion

In this thesis, a Hidden Markov Model (HMM) is combined with different model ideas from statistics and computational intelligence for time series forecasting. Typically, the HMM is used to recognize similar data patterns or to forecast trends in a given time series. What distinguishes this research from earlier work is that rather than building HMMs to forecast the basic movement (or shift) in the time series data, we forecast the (near) exact value of the next event in the time series.

In this research, we have developed novel approaches to forecast time series using a single HMM. Initially, we have used a single HMM to identify similar patterns from the historical dataset and interpolate the differences between the matched pattern and neighboring pattern to forecast the next value in the time series. To improve on the forecast accuracy, a combination of an artificial neural network (ANN) and evolutionary algorithm (EA) is used. An ANN is used to transform the correlated observation sequences into uncorrelated observation sequences and the EA is used to obtain the optimal initial parameter values of the HMM. An automatic data-driven fuzzy model generation method based on a single HMM is developed to further improve the forecast accuracy for the time series data. We have evaluated the performance of our models using daily stock prices for six stocks listed on the Australian Security Exchange. We have also examined the alternative hybrid combinations of the HMM-Fuzzy system using three benchmark chaotic time series. The results show that our model outperforms other well-known time series forecasting techniques. It is worth noting that forecasting time series data has been researched for a long time with varying success and it would be notably significant even if a little improvement in performance is achieved.

8.1 Contribution

The major contributions in this thesis are to develop single HMM-based hybrid soft computing methods for generating exact or near exact future values of time series. The developed methods are easy to interpret, computationally inexpensive and generate forecasts with improved accuracy.

1. In Chapter 4, a single HMM is used to find the similar data patterns. In exploring the capability of a single HMM in identifying similar data patterns and grouping into clusters, our experiments using 4 benchmark datasets clearly demonstrate that the approach is rich in pattern identification. Empirical results using a number of benchmark datasets clearly demonstrate the efficiency of the proposed approach when compared to that of some of the well known techniques. Furthermore, the approach facilitates the grouping of the data without giving any prior information about the structure of the respective dataset, which clearly is an unsupervised technique.

2. In Chapter 5, an initial HMM-based forecasting method, HiMMI, is developed. The idea of using a single HMM, as shown in Chapter 4, has been used to predict the time series data in contrast to the traditional approach of the HMM, which is to detect data patterns or trends over time. The forecast accuracy of the HiMMI is comparable to that of the ANN and the ARIMA for the six financial time series data considered.

3. In the fusion model, the limitations of the HiMMI are solved by hybridizing the ANN and the GA with it. In addition, we introduced a weighted average formula to prepare the future value as a forecast. Each and every step in the HMM-based forecasting is transparent like a 'white box', and it does not require domain expertise to use them. The only information required from a user is the training dataset along with the predictor time series data, and hence the developed model is not constrained by user given parameters. As evident from the experimental results, the improvement in the forecast accuracy over that of the HiMMI is altogether impressive.

   An analysis of the dataset reveals that they do not have substantial underlying seasonality (see Appendix for the ACF and the PACF plots) and hence can be efficiently modeled using autoregressive processes. However, our fusion model produced a com-

petitive performance even with the best performance of the autoregressive processes for these datasets.

4. In Chapter 6 we have developed the HMM-Fuzzy model, which is able to generate a minimum number of fuzzy rules with improved prediction accuracy. A key advantage of this approach is that it is not necessary to analyze the training dataset before using the model, as it is necessary for some existing data driven models such as [131]. Furthermore, this new approach is not constrained with user defined parameters, such as the number of clusters to divide the dataset or the radious of clusters. We have applied the HMM-fuzzy model to the well known Mackey-Glass time series data along with the six stock prices, and the empirical results clearly demonstrate the efficiency of the hybrid approach by generating improved forecast accuracy compared to the HMM-based forecasting methods. Moreover, the performance of the HMM-Fuzzy model for Mackey-Glass data is quite impressive when we compare it with the other existing fuzzy systems. A 38.9% performance improvement is achieved when compared to one of the recently developed data driven fuzzy models. It is interesting to note that forecasting accuracy using the HMM-Fuzzy model outperforms the ARIMA model for the six financial time series data.

5. In Chapter 7 we presented the integration of the HMM-Fuzzy model and EAs. This work may be viewed as an adaptive computational framework for automatic generation of the fuzzy model with an optimal number of fuzzy rules and improved accuracy. In terms of RMSE, both the HMM-Fuzzy and HMM-Fuzzy with EA outperformed the conventional design of fuzzy systems using contemporary techniques. For comparison purposes, the empirical results from conventionally designed HMM-Fuzzy models were compared with those of the ANFIS (Adaptive Neuro-Fuzzy Inference System), implementing a Takagi-Sugeno fuzzy inference system. For the Mackey-Glass time series data, the performance of the HMM-Fuzzy with EA is superior to that of the HMM-Fuzzy model using a reduced number of fuzzy rules. It is interesting to note that our model performed at least 40% better in terms of the NRMSE when compared to other data-driven fuzzy systems, namely SuPFuNIS, data-driven linguistic modeling using relational fuzzy rules, and DENFIS.

   Experimental results demonstrate the superior performance of HMM-Fuzzy with EA for the other two known benchmark time series data. Referring to the empirical

results of HMM-Fuzzy with EA, for the automobile gas mileage data the hybrid model performed marginally better. This might be due to the comparatively weak non-stationarity underlying the time series data and perhaps this could not be well represented using a Takagi-Sugeno fuzzy inference system.

8.2 Future Research Directions

To choose the best parameter values and best structure of an HMM, a variety of hybridizations exist (e.g. by using ANN and FL to generate the observation emission probabilities) apart from that of using the Gaussian distribution. It might be possible to improve the performances of the HMM-based forecasting methodologies and the HMM-Fuzzy models by using those techniques to choose the parameter values, instead of using the Gaussian distribution. A further evolutionary approach could be used to choose the best HMM architecture along with choosing the optimal initial parameter values of the HMM, with the penalty of increased computational complexity.

We use a fixed bin/bucket size while grouping the similar data patterns prior to the generating the fuzzy model. It would be interesting to explore the effect on the performance if the size is made adaptive to the varying dataset. While generating fuzzy rules using the top-down tree approach, the dataset is divided into two parts by dividing the total number of bins/buckets. It would be also interesting if the division could be done in an adaptive manner where the total number of bins would be divided into two parts with varying size (in terms of the number of bins/buckets) in each part, rather than being equal.

In a traditional fuzzy model all the rules have the same size; i.e., the consequent part of each of the rules uses the same number of dependent variables. It is quite logical that all the dependent variables might not have a significant role in producing the exact output value for all the rules. Hence, a varying size of rules using the most significant variables in the respective rule could be useful both for an improvement in performance and reduction in computation time and complexity.

Further hybridization studies of the HMM with other intelligent techniques such as Support Vector Machines [164], Artificial Immune Systems [205], Bayesian methods [206], Rough sets [207] and popular case-based reasoning systems, may be of interest.

References

[1] B. L. Bowerman and R. T. O'Connell, *Time Series Forecasting Unified Concepts ans Computer Implementation*, 2nd ed. Duxbury Press, Boston, 1987.

[2] X. Huang, Y. Ariki, and M. Jack, *Hidden Markov Models for speech recognition*. Edinburgh University Press, 1990.

[3] L. W. K. Cheung, "Use of runs statistics for pattern recognition in genomic dna sequences," *Journal of Computational Biology*, pp. 107–124, 2004.

[4] E. J. Hannan, *Time series analysis from Methuen's monographs on applied probability and statistics*. Science paperbacks and Methuen & Co. Ltd., 1960.

[5] X. Wang, "Characteristic-based forecasting for time series data," Ph.D. dissertation, Faculty of Information Technology, Monash University, July 2005.

[6] S. Makriadakis, S. C. Wheelwright, and R. Hyndman, *Forecasting: Methods and Applications*, 3rd ed. New York: John Wiley { Sons, Inc, 1998.

[7] W. Feller, *An introduction to Probability Theory and Its Applications*. Volume 1, ISBN: 0-471-25708-7, 1968.

[8] T. Bollerslev, "Generalized autorregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, pp. 307–327, 1986.

[9] R. Durrett and H. Kesten, *Random Walks, Brownian Motion and Interacting Particle Systems (Progress in Probability)*. Birkhauser Verlag AG, 1991.

[10] R. Joseph and S. Maxwell, *Commodity Futures Trading With Moving Averages*. Speer Books, 1975.

[11] W. Mendenhall and T. L. Sincich, *A Second Course in Statistics: Regression Analysis, Sixth Edition*, 6th ed. Prentice Hall, 2003.

[12] P. Burridge, *Forecasting and signal extraction in autoregressive-moving average models (Warwick economic research papers).* Dept. of Economics, University of Warwick, 1986.

[13] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, 2nd ed. Springer VerLag, March 2003.

[14] A. K. Bera and M. L. Higgins, "Arch models: Properties, estimation and testing," *Journal of Economic Surveys*, vol. 7, no. 4, pp. 307–366, 1993.

[15] URL, "Market fluctuation forecast – a fuzzy time series analysis approach to data mining," Online, 4 Februrary 2007, http://www.zaptron.com/literature/timeseries.htm.

[16] S. Haykin, *Adaptive filter theory*, 2nd ed. Englewood Cliffs, NJ Prentice Hall, 1991.

[17] R. K. Begg and M. R. Hassan, *Artificial Neural Networks in Smart Home.* Chapter 9 in Designing Smart Home, Editted by: Juan C. Augusto and Chris D. Nugent, Springer-Verlag, Burlin, 2006.

[18] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Communications of the ACM*, vol. 37, no. 3, pp. 77–84, 1994.

[19] D. G. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1989.

[20] S. D. Balkin and J. K. Ord, "Automatic neural network modeling for univariate time series," *Journal of Forecasting*, vol. 16, pp. 509–515, 2000.

[21] J. S. Armstrong and K. C. Green, "For researchers in forecasting (evidence-based forecasting)," http://www.forecastingprinciples.com/m3-competition.html.

[22] A. L. Blum and R. L. Rivest, "Training a 3-node neural networks is np- complete," *Neural Networks*, vol. 5, pp. 117–127, 1992.

[23] P. C. Nayak, K. P. Sidheer, and K. S. Ramasastri, "Fuzzy computing based rainfall-runoff model for real time flood forecasting," *Hydrological Processes*, vol. 19, pp. 955–968, 2005.

[24] R. Simutis, "Fuzzy logic based stock trading system," *Proceedings of the IEEE/IAFE/INFORMS 2000 Conference on Computational Intelligence for Financial Engineering*, pp. 19–21, 2000.

[25] S.-M. Chen and J.-R. Hwang, "Temparature prediction using fuzzy time series," *IEEE Transactions on Systems, Man and Cybernatics-Part B: Cybernatics*, vol. 30, no. 2, pp. 263–275, 2000.

[26] C.-H. Cheng, T.-L. Chen, and C.-H. Chiang, "Trend-weighted fuzzy time-series model for taiex forecasting," *Proceedings of ICONIP*, vol. III, pp. 469–477, 2006.

[27] J. R. Jang, "Anfis: Adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernatics*, vol. 23, pp. 51–63, 1993.

[28] E. Andersson, D. Bock, and M. Frisen, "Detection of turning points in business cycles," *Journal of statistics, Goteborg University, Sweden*, pp. 93–108, 2004.

[29] A. S. Weigend and S. Shi, "Predicting daily probability distributions of s&p500 returns," *Journal of Forecasting*, pp. 375–392, 2000.

[30] E. Trentin and M. Gori, "A survey of hybrid ANN/HMM models for automatic speech recognition," *Neurocomputing*, vol. 37, pp. 91–126, 2001.

[31] H. Bourlard and N. Morgan, *Connectionist Speech Recognition. A Hybrid Approach,*. The Kluwer International Series in Engineering and Computer Science, 1994, vol. 247.

[32] E. Levin, "Word recognition using hidden control neural architecture," in *Internationa Conference on Acoustics, Speech and Signal Processing*, 1990, pp. 433–436.

[33] P. Haffner, M. Franzini, and A. Waibel, "Integraing time alignment and neural networks for high performance continuous speech recognition," in *International Conference on Acoustics, Speech and Signal Processing*, 1991, pp. 105–108.

[34] S. Haykin, *Neural Networks: A Comprehensive Foundation.* Macmillan College Publishing Company Inc., 1994.

[35] S. R. y Cajal, *Histologie du systeme nerveux de l'homme et des vertebras.* Paris: Malonie; Edition Franxaise Revue: Tome I. 1952; Tome II. 1955, Madrid: Consejo Superior de Investigaciones Cientificas, 1911.

[36] M. H. Hassoun, *Fundamentals of Artificial Neural Network*. Massachusetts Institute of Technology, USA, 1995.

[37] H. Takagi, *Intelligent Systems: Fuzzy Logic, Neural Networks and Genetic Algorithms*. Norwell, Massachusets, USA: Kluwer Academic Publishers, 1997, ch. Introduction to Fuzzy Systems, Neural Networks, and Genetic Algorithms.

[38] D. Rumelhart and J. McClelland, *Parallel Distributed Processing*. MIT Press, 1986.

[39] J. M. Mendel, "Fuzzy logic systems for engineering: A tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, 1995.

[40] T. Munakata and Y. Jani, "Fuzzy systems; an overview," *Communications of the ACM*, vol. 37, no. 3, pp. 68–76, 1994.

[41] D. Dubios and H. Prade, *Possibility theory: An approach to Computerized Processing of uncertainty*. Plenum, New York, 1988.

[42] G. J. Klir and T. A. Floger, *Fuzzy Sers, Uncertainty and Infromation*. Prentice-Hall, Englewood Cliffs, 1988.

[43] B. Kosko, *Neural Networks and Fuzzy Systems*. Prentice-Hall, Englewood Cliffs, 1992.

[44] C. C. Lee, "Fuzzy lofic controller," *IEEE Transactions on System, Man and Cybernatics*, vol. Part I and II, pp. 404–435, 1996.

[45] T. Terano, K. Asai, and M. Sugeno, *Fuzzy Systems Theory and Its Applications*. Academic Press, San Diego, Calif, 1992.

[46] C. M. School of Computer Science, "What is fuzzy logic?" Online, April 1993, http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq-doc-2.html.

[47] R. L. Haupt and S. E. Haupt, *Practical genetic algorithms*. New York : Wiley, 1998.

[48] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *Journal of forecasting*, vol. 5, pp. 559–583, 1989.

[49] S. M. Stigler, "Laplace, fisher, and the discovery of the concept of sufficiency," pp. 439–445, 1973.

[50] N. Amjady, "Short-term hourly load forecasting using time-series modeling with peak load estimation capabilty," pp. 498–505, 2001.

[51] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 35, pp. 35–45, 1997.

[52] M. J. C. Hu, *Application of the adaline system to weather forecasting.* Master Thesis, Technical Report 6775-1. Stanford Electronics Laboratories, Stanford, CA, 1964.

[53] D. Lowe and A. R. Webb, "Time series prediction by adaptive networks: A dynamical systems perspective." *IEE proceedings-F138 (1)*, pp. 17–24, 1990.

[54] B. E. Rosen, "Neural networks moving averages for time series prediction," *SPIE*, vol. 1966, no. 2, pp. 448–456, 1993.

[55] T. Hill, M. O'Connor, and W. Remus, "Neural network models for time series forecasts," *Management Science*, vol. 42, no. 7, pp. 1082–1092, 1996.

[56] M. Nelson, T. Hiill, W. Remus, and M. O'Connor, "Time series forecasting using neural networks: Should the data be deseasonalized first?" *Journal of Forecasting*, vol. 18, pp. 359–367, 1999.

[57] J. Faraway and C. Chatfield, "Time series forecasting with neural networks: a comparative study using the airline data," vol. 47, no. 2, pp. 231–250, 1998.

[58] J. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control.* Holden-Day, San Fransisco, CA, 1970,1976 Third Edition published in 1994.

[59] A. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: prediction and system modeling." *Technical Report LA-UR-87-2662, Los Alamos National Laboratory*, 1987.

[60] A. Lapedes and R. Farber, "How neural netts work." *Neural Information Processing Systems, Editted by D.Z. Anderson, American Institute of Physics, New York*, pp. 442–456, 1988.

[61] P. Gartside and M. 0450, "Chaos: Logistics map," Online, 30 June 2007, http://pear.math.pitt.edu/mathzilla/Examples/chaos/logmap.mhtml.

[62] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287–289, 1977.

[63] A. B. Geva, "Scalenet–multiscale neural-network architecture for time series prediction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1471–1482, 1998.

[64] K. A. de Oliveira, A. Vannucci, and E. C. da Silva, "Using artificial neural networks to forecast chaotic time series," *Physica A*, vol. 284, no. 5, pp. 393–404, 2000.

[65] M. Hanon, "A two-dimensional mapping with a strange attractor," *Communications in Mathematical Physics*, vol. 50, pp. 69–77, 1976.

[66] G. Mihalakakou, M. Santamouris, and D. N. Asimakopoulos, "The total solar radiation time series simualtion in athens using neural networks," *Theoretical and applied climatology*, vol. 66, pp. 185–197, 2000.

[67] R. D. Jones, Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee, and P. S. Lewis, "Function approximation and time series prediction with neural networks," *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 649–665, 1990.

[68] D. Y. C. Chan and D. Prager, "Analysis of time series by neural networks," *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 1, pp. 355–360, 1994.

[69] I. Ginzburg and D. Horn, "Learnability of time series," *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 3, pp. 2653–2657, 1991.

[70] I. Ginzburg and D. Horn, "Learning the rule of a time series," *International Journal of Neural Systems*, vol. 3, no. 2, pp. 167–177, 1992.

[71] I. Poli and R. D. Jones, "A neural net model for prediction," *Journal of American Statistical Association*, vol. 89, no. 425, pp. 117–121, 1994.

[72] J. W. Taylor and R. Buizza, "Neural network load forecasting with weather ensemble predictions," *IEEE Transactions on power systems*, vol. 17, no. 3, pp. 626–632, 2002.

[73] R. Baratti, B. Cannas, A. Fanni, M. Pintus, G. M. Sechi, and N. Toreno, "River flow forecast for reservoir management through neural networks," *Neurocomputing*, vol. 55, pp. 421–437, 2003.

[74] W. Huang, B. Xu, and A. Chan-Hilton, "Forecasting flows in apalachicola river using neural networks," *Hydrological Processes*, vol. 18, pp. 2545–2564, 2004.

[75] A. G. Bakirtzis, J. B. Theocharis, S. J. Kiartzis, and K. J. Satsios, "Short term load forecasting using fuzzy neural networks," *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1518–1524, 1995.

[76] P. K. Dash, G. Ramakrishna, A. C. Liew, and S. Rahman, "Fuzzy neural networks for time-series forecasting of electric load," *IEE Proceedings of Generation, Transmission and Distribution*, vol. 142, no. 5, pp. 535–544, 1995.

[77] V. P. S J Kiartzis, A G Bakirtzis, "Short-term load forecasting using neural network," *Electric Power Systems Research*, vol. 33, pp. 1–6, 1995.

[78] G. Tkacz, "Neural network forecasting of canadian gdp growth," *International Journal of Forecasting*, vol. 17, pp. 57–69, 2001.

[79] L. Yu, S. Wang, and K. K. Lai, "A novel nonlinear ensemble forecasting model incorporating GLAR and ANN for foreign exchange rates," *COmputers and Operations Research*, vol. 32, pp. 2523–2541, 2005.

[80] J. Yao and C. L. Tan, "A case study on using neural networks to perform technical forecasting of forex," *Neurocomputing*, vol. 34, pp. 79–98, 2000.

[81] A. Abraham, "Analysis of hybrid soft and hard computing techniques for forex monitoring systems," *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'02).*, pp. 1616–1622, 2002.

[82] M. Z. Susac, *Neural Networks in Investments Prifibality Predictions.* Doctorial Dissertation, University of Zagreb, Faculty of Organization and Informatics, 1993.

[83] H. White, "Economic prediction using neural networks: The case of ibm daily stock returns," *Proceedings of the second annual IEEE conference on neural networks*, vol. II, pp. 451–458, 1988.

[84] C.-C. C. Wong, M.-C. Chan, and C.-C. Lam, "Financial time series forecasting by neural network using conjugate gradient learning algorithm and multiple linear regression weight initialization," Society for Computational Eco-

nomics, Computing in Economics and Finance 2000 61, July 2000, available at http://ideas.repec.org/p/sce/scecf0/61.html.

[85] W. Leigh, R. Purvis, and J. M. Ragusa, "Forecasting the nyse composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support," *Decision Support Systems*, vol. 32, pp. 361–377, 2002.

[86] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. Wiley, 1973.

[87] T. Kolarik and G. Rudorfer, "Time series forecasting using neural networks," *Time Series and Neural Networks*, pp. 86–94, 1994.

[88] R. J. Kuo, L. C. Lee, and C. F. Lee, "Integration of artificial nn and fuzzy delphi for stock market forecasting," *Proceedings of IEEE International Conference on Systems, Man and Cybernatics*, vol. 2, pp. 1073–1078, 1996.

[89] A. Atiya, N. Talaat, and S. Shaheen, "An efficient stock market forecasting model using neural networks," *Proceedings of International Conference on Neural Networks*, pp. 2112–2115, 1997.

[90] T. Kimoto, k. Asakawa, M. Yoda, and M. Takeoka, "Stock market prediction system with modular neural networks," *Proceedings of the international joint conference on neural networks(IJCNN)*, vol. 1, pp. 1–6, 1990.

[91] W.-C. Chiang, T. L. Urban, and G. W. Baldridge, "A neural network approach to mutual fund net asset value forecasting," *Omega International Journal of Management Science*, vol. 24, no. 2, pp. 205–215, 1996.

[92] T. B. Trafalis, "Artificial neural networks applied to financial forecasting," *C. H. Dagi Dagli, A. L. Buczak, J. Ghosh, M. J. Embrechts, & O. Ersoy (Eds.), Smart engineering systems: neural networks, fuzzy logic, data mining, and evolutionary programming. Proceedings of the artificial neural networks in engineering conference (ANNIE'99)*, pp. 1049–1054, 1999.

[93] J. H. Choi, M. K. Lee, and M. W. Rhee, "Trading s&p500 stock index futures using a neural network," *Proceedings of the 3rd annual international conference on artificial intelligence applications on wall street*, pp. 63–72, 1995.

[94] S. Wu and R. Lu, "Combining artificial neural networks and statistics for stock-market forecasting," *ACM Conference on Computer Science*, pp. 257–264, 1993.

[95] P. G. Harrald and M. Kamstra, "Evolving artificial neural networks to combine financial forecasts," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 40–52, 1997.

[96] J. V. Hansen and R. D. Nelson, "Neural networks and traditional time series methods: A synergistic combination in state economic forecasts," *IEEE Transactions on Neural Networks*, vol. 8, no. 4, pp. 863–873, 1997.

[97] N. G. Pavlidis, D. K. Tasoulis, and M. N. Vrahatis, "Financial forecasting through unsupervised clustering and evolutionary trained neural networks," *Proceedings of the Congress on Evolutionary Computation (CEC 2003)*, pp. 2314–2321, 2003.

[98] H. A. Mayer and R. Schaiger, "Evolutionary and coevolutionary approaches to time series prediction using generalized multi-layer perceptrons," *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 275–280, 1999.

[99] X. Yao and Y. Lin, "A new evolutionary sytem for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, pp. 694–713, 1997.

[100] I. DeFalco, A. Iazzetta, P. Natale, and E. Tarantino, "Evolutionary neural networks for nonlinear dynamics modeling," *Proceedings of 5th International Conference on Parallel Problem Solving from Nature*, pp. 593–602, 1998.

[101] K. Becker, B. Thull, H. Kasmacher-Leidinger, J. Stemmer, G. Rau, G. Kalff, and H. J. Zimmermann, "Design and validation of an intelligent patient monitoring and alarm system based on a fuzzy logic process model," *Artificial Intelligence in Medicine*, vol. 11, pp. 33–53, 1997.

[102] M. J. Cordova and J. M. Goldman, "Advanced clininical monitoring: similar scenario discrimination," *Proceedings of Medinfo*, vol. 95, pp. 880–884, 1995.

[103] B. Hayes-Roth, R. Washington, D. Ash, R. Hewett, A. Collinot, A. Vina, and A. Seiver, "Guardian: a prototype intelligent agent for intensive-care monitoring," *Artificial Intelligence in Medicine*, vol. 4, pp. 165–185, 1992.

[104] M. Wolf, M. Keel, K. Von-Siebenthal, H. U. Bucher, K. Geering, Y. Lehareinger, and P. Niederer, "Improved monitoring of preterm infants by fuzzy logic," *Technol Health Care*, p. 4, 1996.

[105] A. Lowe, M. J. Harrison, and R. W. Jones, "Diagnostic monitoring in anaesthesia using fuzzy trend templates for matching temporal patterns," *Artificial Intelligence in Medicine*, vol. 16, pp. 183–199, 1999.

[106] W. Tvarusko, M. Bentele, T. Misteli, R. Rudolf, C. Kaether, D. L. Spector, and H. H. Gerdes, "Time-resolved analysis and visualization of dynamic processes in living cells," *Proceedings of National Academy Sciences*, vol. 96, pp. 7950–7955, 1999.

[107] S. L. Chiu, "An efficient method for extracting fuzzy classification rules from high dimensional data," *Journal of Advanced Computational Intelligence*, vol. 1, pp. 1–7, 1997.

[108] S.-M. Chen, "Forecasting enrollments based on fuzzy time series," *Fuzzy Sets and Systems*, vol. 81, pp. 311–319, 1996.

[109] Q. Song and B. S. Chissom, "Fuzzy time series and its models," *Fuzzy Sets and Systems*, vol. 54, pp. 269–277, 1993.

[110] Q. Song and B. S. Chissom, "Forecasting enrollments with fuzzy time series- part i," *Fuzzy Sets and Systems*, vol. 54, pp. 1–9, 1993.

[111] J.-R. Hwang, D.-M. Chen, and C.-H. Lee, "Handling forecasting problems using fuzzy time series," *Fuzzy Sets and Systems*, vol. 100, pp. 217–228, 1998.

[112] Q. Song and B. S. Chissom, "Forecasting enrollments with fuzzy time series- part ii," *Fuzzy Sets and Systems*, vol. 62, pp. 1–8, 1994.

[113] K. Huarng, "Heuristic models of fuzzy time series for forecasting," *Fuzzy Sets and Systems*, vol. 123, pp. 369–386, 2001.

[114] W. W. L. Cheung, T. J. Pitcher, and D. Pauly, "A fuzzy logic expert system to estimate intrinsic extinction vulnerabilities of marine fishes to fishing," *Biological Conservation*, vol. 124, pp. 97–111, 2005.

[115] R. Froese and D. Pauly, "World wide web electronic publication," http://www.fishbase.org.

[116] S. Mackinson, "An adaptive fuzzy expert system for predicting structure, dynamics and distribution of herring shoals," *Ecological Modelling*, vol. 126, pp. 155–178, 2000.

[117] S. Mackinson, M. Vasconcellos, and N. Newlands, "A new approach to the analysis of stock-recruitment relationships:"model-free estimation" using fuzzy logic," *Canadian Journal of Fisheries and Acquat Science*, vol. 56, pp. 686–699, 1999.

[118] S.-M. Chen and S.-H. Lee, "A new method for generating fuzzy rules from numerical data for handling classification problems," *Applied Artificial Intelligence*, pp. 645–664, 2001.

[119] A. Fiordaliso, "A constrained takagi-sugeno fuzzy system that allows for better interpretation and analysis," *Fuzzy Sets and Systems*, vol. 118, pp. 307–318, 2001.

[120] U. Hubner, N. B. Abraham, and C. O. Weiss, "Dimensions and entropies of chaotic intensity pulsations in a single-mode far infrared nh_3 laser," *Phys. Rev.*, vol. A 40, pp. 6354–6365, 1989.

[121] K. C. Lee and H. S. Kim, "A fuzzy cognitive map-based bi-directional inference mechanism: an application to stock investment analysis," *Intelligent Systems in Accounting, Finance and Management*, vol. 6, pp. 41–57, 1997.

[122] R. J. Kuo, P. Wu, and C. P. Wang, "An intelligent sales forecasting system through integration of artificial neural networks and fuzzy neural networks with fuzzy weight elimination," *Neural Networks*, vol. 15, pp. 909–925, 2002.

[123] H. A. Linstone and M. Turoff, *The Delphi Method:Techniques and Applications*. New Jersey Institute of Technology, 2002.

[124] K. Huarng and H.-K. Yu, "A type 2 fuzzy time series model for stock index forecasting," *Physica A*, vol. 353, pp. 445–462, 2005.

[125] M. Setnes and O. J. H. van Drempt, "Fuzzy modeling in stock-market analysis," *Proceedings of the IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*, pp. 250–258, 1999.

[126] M. Tarrazo and L. Gutierrez, "Theory and methodology economic expectations, fuzzy sets and financial planning," *European Journal of Operational Research*, vol. 126, pp. 89–105, 2000.

[127] E. Vercher, J. D. Bermudez, and J. V. Segura, "Fuzzy portfolio optimization under downside risk measures," *Fuzzy Sets and Systems*, vol. 158, pp. 769–782, 2007.

[128] D. Petrovic, Y. Xie, and K. Burnham, "Fuzzy decision support system for demand forecasting with a learning mechanism," *Fuzzy Sets and Systems*, vol. 157, pp. 1713–1725, 2006.

[129] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity of processing information," *The pshychological Review*, vol. 63, pp. 81–97, 1956.

[130] M.-S. Kim, "Parallel-structure fuzzy system for sunspot cycle prediction in the railway systems," *Proceedings of FSKD*, pp. 919–928, 2006.

[131] A. E. Gaweda and J. M. Zurada, "Data-driven linguistic modeling using relational fuzzy rules," *IEEE Transactions on Fuzzy Systems*, vol. 11, pp. 121–134, 2003.

[132] J. Zurada, *Optimal data driven rule extraction using adatptive fuzzy-neural models.* PhD dissertation, University of Louisbille, 2002.

[133] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer Academic Press, 1981.

[134] J. C. Bezdek, R. Ehrlick, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comp. Geoscience*, vol. 10, pp. 191–203, 1984.

[135] L.-X. Wang and J. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on System, Man, Cybernetics*, vol. 22, pp. 1414–1427, 1992.

[136] J. Zurada and A. Lozowski, "Generating linguistic rules from data using neuro-fuzzy framework," *Proceedings of the Fourth International Conference on Soft Computing*, pp. 618–621, 1996.

[137] G. G. Szpiro, "Forecasting chaotic time series with genetic algorithms," *Physical Review E*, vol. 55, no. 3, pp. 2557–2568, 1997.

[138] P. Cortez, M. Rocha, and J. Neves, "Genetic and evolutionary algorithms for time series forecasting," *Proceedings of IEA/AIE*, pp. 393–402, 2001.

[139] S. Magfoud and G. Mani, "Financial forecasting using genetic algorithms," *Applied Artificial Intelligence*, vol. 10, pp. 543–565, 1996.

[140] B. Jeong, H.-S. Jung, and N.-K. Park, "A computerized causal forecasting system using genetic algorithms in supply chain management," *The Journal of Systems and Software*, vol. 60, pp. 223–237, 2002.

[141] J. Kohlmorgen, S. Lemm, K.-R. Muller, S. Liehr, and K. Pawelzik, "Fast change point detection in swithcing dynamics using a hidden Markov model of prediction experts," *Proceedings of Artificial Neural Networks*, pp. 204–209, 1999.

[142] R. Bhar and S. Hamori, "Hidden Markov models: Applications to financial economics," 2005.

[143] R. C. Vasko, A. El-Jaroudi Jr., and J. R. Boston, "An algorithm to derermine hidden Markov model topology," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1996, pp. 3577–3580.

[144] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proceedings of the IEEE*, vol. 73, no. 11, pp. 1551–1585, 1985.

[145] M. Karlsson, "Hidden Markov models," http://www.math.chalmers.se/~olleh/Markov_Karlsson.pdf.

[146] R. Dugad and U. B. Desai, "A tutorial on hidden Markov model," *Technical Report*, pp. 1–16, 1996.

[147] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[148] A. B. Poritz, "Hidden Markov models: a guided tour," *Proceedings of ICASSP*, pp. 7–13, 1988.

[149] L. E. Baum, T. Pitrie, G. Souls, and N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, pp. 164–171, 1970.

[150] H. Bahi and M. Sellami, "Combination of vector quantization and hidden Markov models for arabic speech recognition," *ACS/IEEE Proceedings of International Conference on Computer Systems and Applications*, p. 0096, 2001.

[151] L. E. Baum, "An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes," *Inequalities*, vol. 3, pp. 1–8, 1972.

[152] L. R. Liporace, "Maximum-likelihood estimation for multivariate observations of Markov sources," *IEEE Transactions on Infrimation theory*, vol. IT-28, pp. 729–734, 1982.

[153] J. Doob, "Stochastic processes," *John Wiley*, p. 10, 1953.

[154] C. P. Papageorgiou, "Mixed memory Markov models for time series analysis," *Proceedings of the IEEE/IAFE/INFORMS 2000 Conference on Computational Intelligence for Financial Engineering*, pp. 165–170, 1998.

[155] L. Saul and M. I. Jordan, "Mixed Markov models," *Proceedings of the 1997 Conference on Artificial Intelligence and Statistics*, 1997.

[156] N. Viovy and G. Saint, "Hidden Markov models applied to vegetation dynamics analysis using satellite remote sensing," *IEEE transactions on geoscience and remote sensing*, pp. 906–917, 1994.

[157] S. Shi and A. S. Weigend, "Taking time seriously: Hidden Markov experts applied to financial engineering," *Proceedings of the IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*, pp. 244–252, 1997.

[158] Y. Zhang, *Prediction of financial time series with Hidden Markov Models*, 2004.

[159] A. K. Jain, M. N. Murty, and P. J. Flunn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, pp. 264–323, 1999.

[160] J. A. Hartigan and M. A. Wong, "Algorithm as 136: a k-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[161] T. Kohonen, *Self Organization and Associative Memory (3rd edn)*. Springer, 1989.

[162] T. Kohonen, *Self Organizing Maps, Springer Series in Information Sciences.* Springer, 1989, vol. 30.

[163] R. Begg, R. Hassan, S. Taylor, and M. Palaniswami, "Artificial neural network models in the diagnosis of balance impairments," *Proceedings of International conference on Intekkigent Sensing and Information Processing*, pp. 518–522, 2005.

[164] V. N. Vapnik, "The nature of statistical learning theory," *Springer, New York*, 1990.

[165] L. Kuafman and P. J. Rousseuw, *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley, 1990.

[166] M. R. Hassan, B. Nath, and M. Kirley, "A data clustering algorithm based on single hidden Markov model," *Proceedings of the International Multiconference on Computer Science and Information Technology*, pp. 57–66, 2006.

[167] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annula Eugenics*, vol. II, pp. 179–188, 1936.

[168] "www.ics.uci.edu/pub/ml-rwpoa/machine-learning-database/."

[169] L. Breiman, "Bias, variance and arcing classifiers," *Technical Report 460, Statistics department, University of California*, April,1996.

[170] K. cup dataset, Online, October 1999, http://kdd.ics.uci.edu/databases/kddcup99 /kddcup99.html.

[171] D. Coomans, M. Broeckaert, M. Jonckheer, and D. L. Massart., "Comparison of multivariate discriminant techniques for clinical dataapplication to the thyroid functional state," *Methods of Information in Medicine*, vol. 22, pp. 93–101, 1983.

[172] M. R. Hassan and B. Nath, "Stock market forecasting using hiddent Markov model: A new approach," *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, pp. 192–196, 2005.

[173] M. R. Hassan, B. Nath, and M. Kirley, "A fusion model of HMM, ANN and GA for stock market forecasting," *Expert Systems with Applications*, vol. 33, no. 1, pp. 171–180, 2007.

[174] F. Jelinek, M. Kaufmann, and C. S. Mateo, *Self-organized language modelling for speech recognition.* Readings in Speech Recognition (Eds. Alex Waibel and Kai-Fu Lee), Morgan Kaufmann, San Mateo, California, 1990.

[175] H. Xie, P. Anreae, M. Zhang, and P. Warren, "Learning models for english speech recognition," *Proceedings of the 27th Conference on Australasian Computer Science*, pp. 323–329, 2004.

[176] D. A. Coast, R. M. Stern, G. G. Cano, and S. A. Briller, "An approach to cardiac arhythmia analysis using hidden Markov models," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 9, pp. 826–836, 1990.

[177] P. Smyth, "Clustering sequences with hidden Markov models," *G. Tesauro, D. Touretzky, T. Leen (Eds), Advances in neural information processing systems*, vol. 9, pp. 648–654, 1997.

[178] A. Sankar, "Experiments with gaussian merging-splitting algorithm for HMM training for speech recognition," *Proceedings of DARPA speech recognition workshop*, 2001. [Online]. Available: *www.nist.gov/speech/publications/darpa*98/*html*/*am*10/*am*10.*htm*

[179] S. Kwong and H. Qianhua, "The use of adaptive frame for speech recognition," *EURASIP Journal on Applied Signal Processing*, vol. 2, pp. 82–88, 2001.

[180] Y. Bengio, R. D. Mori, and R. Kompe, "Global optimization of a neural network-hidden Markov model hybrid," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.

[181] A. Nadas, D. Nahamoo, and M. A. Picheny, "On model-robust training method for speech recognition," *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. 77, pp. 257–285, 1988.

[182] P. P. Angelov and R. A. Buswell, "Automatic generation of fuzzy rule-based models from data by genetic algorithms," *Information Science*, pp. 17–31, 2003.

[183] X. Z. Wang, Y. D. Wang, X. F. Xu, W. D. Ling, and D. S. Yeung, "A new approach to fuzzy rule generation: fuzzy extension matrix," *Fuzzy Sets and Systems*, pp. 291–306, 2001.

[184] L. M. Reyneri, "Unification of neural and wavelet networks and fuzzy systems," *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 801–815, 1999.

[185] H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, pp. 1–13, 1975.

[186] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Transactions on System, Man and Cybernatics*, pp. 116–132, 1985.

[187] M. Mannle, "Identifying rule-based tsk fuzzy models," *Proceedings of EUFIT*, pp. 286–299, 1999.

[188] L. I. Davis Jr., R. W. Matterson, and G. A. Dage, "Method and control system for controlling an automotive hvac system to prevent fogging," *US Patent Number: 5516041, Assignee: Ford Motor Company*, 1996.

[189] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft-Computing: A Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ Prentice-Hall, 1997.

[190] M. Russo, "Genetic fuzzy learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 259–273, 2000.

[191] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system (supfunis)," *IEEE Transactions on Neural Networks*, vol. 13, pp. 578–599, 1993.

[192] N. K. Kasabov and Q. Song, "Denfis: Dynamic evolving neural-fuzzy inference system and its application for time series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, pp. 144–154, 2002.

[193] Y. Oysal, "Time delay dynamic fuzzy networks for time series prediction," *Proceedings of ICCS, LNCS 3514*, pp. 775–782, 2005.

[194] E. Kim, M. Park, S. Ji, and M. Park, "A new approach to fuzzy modeling," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 3, pp. 328–337, 1997.

[195] M. R. Hassan, B. Nath, and M. Kirley, "Fuzzy modeling using HMM and EA for time series prediction," *IEEE Transactions on Fuzzy Systems*, (Submitted).

[196] M. R. Hassan, B. Nath, and M. Kirley, "HMM based fuzzy model for time series forecasting," *Proceedings of World Congress on Computational Intelligence(WCCI2006)*, pp. 9963–9968, 2006.

[197] C. C. Coello, D. V. Veldhuizen, and G. Lamont, *EA for Solving Multi-Objective Problems*. Kluwer, 2002.

[198] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley and Sons, Ltd., Chichester, England, 2001.

[199] R. Quinlan, "Combining instance-based and model-based learning," *Proceedings on the Tenth International Conference of Machine Learning*, pp. 236–243, 1993.

[200] E. Kim, M. Park, S. Ji, , and M. Park, "A transformed input-domain approach to fuzzy modeling," *IEEE Transactions on Fuzzy Systems*, pp. 546–604, 1998.

[201] L. Wang and R. Langari, "Building sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques," *IEEE Transactions on Fuzzy Systems*, pp. 454–458, 1995.

[202] M. Sugeno and M. Tanaka, "Successive identification of a fuzzy model and its application to prediction of a complex system," *Fuzzy Sets Systems*, pp. 315–334, 1991.

[203] M. Sugeno and T. Yasukawa, "A fuzzy logic based approach to qualitative modeling," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, pp. 1–7, 1993.

[204] S. Chiu, "Selecting input variables in fuzzy models," *Journal of Intelligent and Fuzzy Systems*, vol. 4, pp. 243–256, 1996.

[205] L. N. de Castro and J. I. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. London, U.K.: Springer-Verlag, 2002.

[206] C. P. Robert, *The Bayesian choice: From decision-theoretic foundations to computational implementation*. New York: Springer, 2001.

[207] Z. Pawlak, *Rough Sets: Theoretical Aspects of Reasoning About Data*, Series D: System Theory, Knowledge Engineering and Problem solving. Kluwer Academic Publishers, 1991.

[208] M. R. Hassan, R. K. Begg, Y. Morsi, and K. Lynch, "HMM-Fuzzy model for breast cancer diagnosis," *Proceedings of 15th International Conference on Mechanics in Medicine and Biology*, 2006.

[209] C. L. Black and C. J. Merz, "Uci repository of machine learning databases," *Department of Information and Computer Sciences, University of California, Irvine*, 1988.

[210] H. A. Abbas, "An evolutionary artificial neural networks approach for breast cancer diagnosis," *Artificial Intelligence in Medicine*, vol. 25, pp. 265–281, 2001.

[211] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proceedings of the National Academy of Sciences*, vol. 87, pp. 9193–9196, 1990.

[212] D. B. Fogel, E. C. Wasson, and E. M. Boughton, "Evolving neural network for detecting breast cancer," *Cancer Letter*, vol. 96, no. 1, pp. 49–53, 1995.

[213] M. R. Hassan, R. K. Begg, S. Taylor, and D. K. Kumar, "HMM-Fuzzy model for recognition of gait changes due to trip related falls," *Proceedings of 28th IEEE International Conference on EMBS*, pp. 1216–1219, 2006.

[214] M. R. Hassan, R. K. Begg, and S. Taylor, "Fuzzy logic-based recognition of gait changes due to trip-related falls," *Proceedings of the 27th International IEEE-EMBS Conference*, pp. 4970–4973, 2005.

[215] R. K. Begg, M. Palaniswami, and B. Owen, "Support vector machines for automated gait classification," *IEEE Transactions on Biomedical Engineering*, vol. 52, pp. 828–838, 2005.

[216] A. Das and P. Das, "Chaotic analysis of foreign exchange rates," *Applied Mathematics and Computation*, vol. 185, pp. 388–396, 2007.

[217] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. Farmer, "Testing nonlinearity in time series: the method of surrogated data," *Physica D*, vol. 58, pp. 77–94, 1992.

[218] SPSS Inc., *SPSS trends 13.0*, SPSS Inc, 2004.

Part IV

Appendix

Appendix A

Data classification using HMM-Fuzzy model

A.1 Breast Cancer recognition

Dataset

This section describes the diagnoses and classification of breast cancer using the HMM-based fuzzy model. Findings reported in this section have been published in [208]. The dataset we used was collected from the UCI repository [209]. The dataset comprises 699 patterns with 9 attributes: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatic, normal nucleoli, and mitoses. The output included two classes: benign and malignant. Abbas [210] suggests that the original dataset was described by Wolberg and Mangasarian [211].

Training and Testing Datasets

To ensure that our measures were consistent with other investigations [210, 212], we removed 16 instances of missing values from the dataset and constructed a new dataset with 683 examples. The initial 400 instances in the new dataset were set aside for training purposes, whereas the remaining 283 were retained as the test set. To make sure that we could compare our performance with that reported by Abbas [210] and Fogel *et al.* [212], we executed the experiment a total of fifteen times. The desired MSE, to stop the rule creation process, was set to 0.08 and bucket sizes were fixed to 1. The experiment used a five state HMM.

Table A.1: Accuracy of the breast cancer classification

Method	Classification Accuracy
HMM-Fuzzy Model	98.163 ± 0.2389
MPANN	98.100 ± 0.5000
Fogel *et al.* [212]	98.100 ± 46.4000

Results

The average test error of the HMM-based Fuzzy model was calculated for the 15 runs. The average accuracy and the number of generated fuzzy rules across the 15 runs are presented as Tab.3. A small standard deviation, along with a manageable number of fuzzy rules implies a model's ability to classify breast cancer with reasonable computational complexity. Table 3 contrasts the performance of the HMM-based fuzzy model with other reported results like Abbas [210] and Fogel *et al.* [212].

A.2 Balance impairment problem identification

This section reports the use of the HMM-Fuzzy model for identifying the differences in gait between people with tendencies to fall and healthy people. The findings have been published in [213].

Participants

Our volunteers for the gait data collection included ten healthy older adults (H) and ten older adults with balance impairments (I). All the volunteers were > 65 years old and hailed from the local community and senior citizen clubs. Victoria University's Human Research Ethics Committee approved the informed-consent procedures undertaken by the volunteers. None of them had known injuries or abnormalities that could affect their normal gait.

Gait dataset

Data collection and gait features extraction can be found in [214], but for convenience's sake a brief outline of the procedure is included here.

Foot clearance data sets were collected during steady state self-selected walking on

a treadmill with a PEAK MOTUS 2D (Peak Technologies Inc, USA) motion analysis system. Two reflective markers were attached to each subject's left shoe, representing the fifth metatarsal head and the great toe. Each subject completed between ten and twenty minutes of normal walking at a self-selected comfortable walking speed. Marker positions and shoe dimensions were used to predict the position on the shoe traveling closest to the ground at the time MFC (Minimum Foot Clearance) occurred using a 2D geometric model of the foot. MFC was calculated by subtracting ground reference from the minimum vertical coordinate during the swing phase.

Features describing major statistical characteristics of MFC distributions were extracted and these included altogether nine features: mean, median (Q2), min, max, 25th quartile (Q1), 75th quartile (Q3), standard deviation (SD), skewness and kurtosis [214].

The input data that described individual gait patterns belonged to two classes: healthy and balance impaired (marked as 'fallers').

Result

To generalize and document the performance of the model a five fold cross validation test was adopted. Toward this purpose, the dataset was split up into subsets so that each one comprised data from two healthy people and two impaired people. Four subsets were used to extract the fuzzy rules. The parameters obtained after training were saved, and the remaining data subsets were tested against them. The results of the five fold cross validation tests were, in turn, used to obtain the receiver operating characteristics (ROC) area. ROC areas have been used on prior occasions [214, 215] by researchers to gauge the predictive ability of a classifier over a broad range of threshold values.

Table A.2 displays the overall test result for the cross validation scheme. From this table it may be observed that the overall accuracy is greater than the 91^{st} percentile considering that there are not many fuzzy rules. The ROC plot, again, shows the promise of high accuracy the HMM-Fuzzy model displays: an ROC area of > 0.95. ROC area is a better indicator of performance as it accounts for threshold variation from the default (0). The HMM-Fuzzy model provided both a better ROC area, and improved sensitivity and specificity results, with merely two fuzzy rules.

Table A.2: Accuracy of the gait problem classification

Method	Classification Accuracy	Sensitivity	Specificity	ROC area	Number of Rules
HMM-Fuzzy Model	91.3	0.884	0.95	0.954	2
Subtractive clustering based Fuzzy Model	89.3	0.870	0.92	0.930	15 to 18

Appendix B

Non-stationarity identification

B.1 Autocorrelation and Partial Autocorrelation function plots

In this section we present the plots of Autocorrelation (ACF) and Partial Autocorrelation (PACF) functions for the dataset considered in this thesis. ACF and PACF plots are useful in identifying the underlying non-stationarity in the dataset. While using ARIMA, these plots are especially useful to determine the order of AR, MA and the integration. What follows are some general guidelines to identify the process [218]:

- If the number of residual points in the ACF plot for the corresponding lags are significant (i.e. above the horizontal line shown in blue) then the plot indicates that the time series is non-stationary.
- If the residual points corresponding to the lags in ACF decline exponentially and one or more initial residual points (i.e. first one or two lags) in PACF are significant then this indicates that the time series can be well fitted using autoregression.
- If the first or more residual points in the ACF are significant and the residual points in the PACF decays exponentially then the time series can be modeled by moving average.
- A non-stationary time series data with moving average and autoregressive process exhibits a combination of the above mentioned ACF and PACF characteristics.

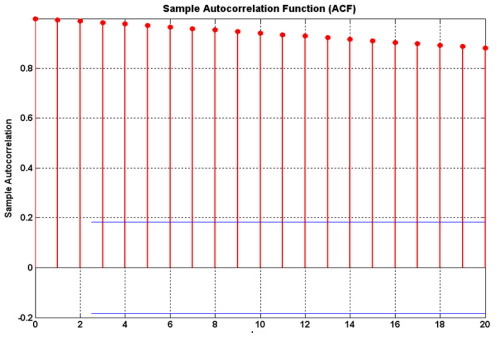


Figure B.1: ACF plot for daily stock (closing) prices of British Airlines

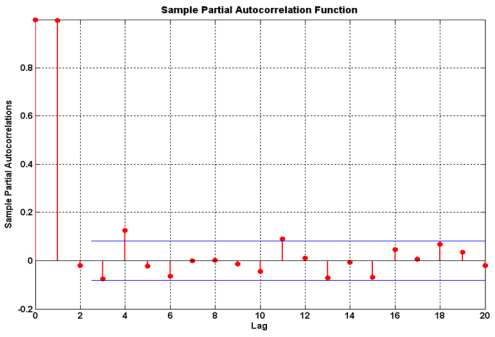


Figure B.2: PACF plot for daily stock (closing) prices of British Airlines

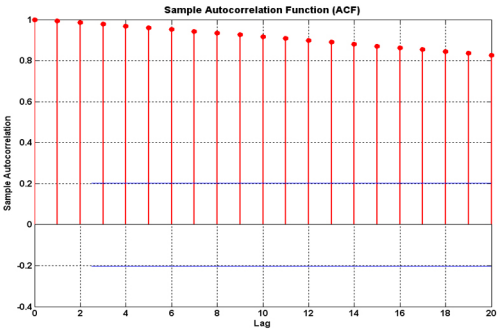


Figure B.3: ACF plot for daily stock (closing) prices of Delta Airlines

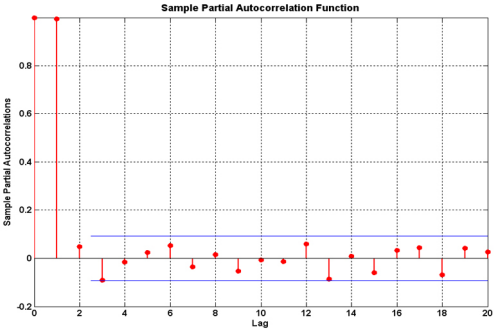


Figure B.4: PACF plot for daily stock (closing) prices of Delta Airlines

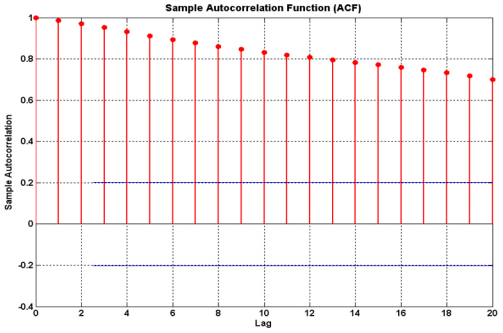


Figure B.5: ACF plot for daily stock (closing) prices of Ryanair Airlines

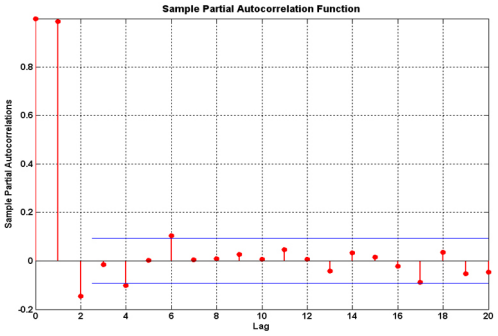


Figure B.6: PACF plot for daily stock (closing) prices of Ryanair Airlines

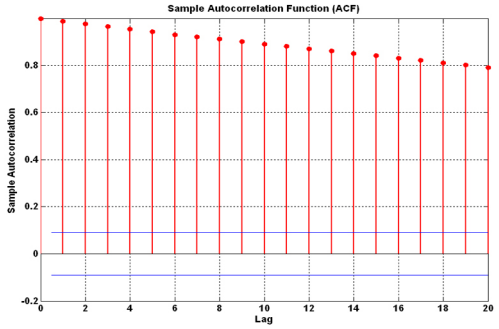


Figure B.7: ACF plot for daily stock (closing) prices of Apple Computer Inc.

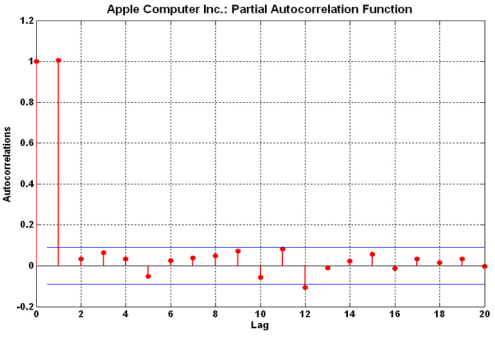


Figure B.8: PACF plot for daily stock (closing) prices of Apple Computer Inc.

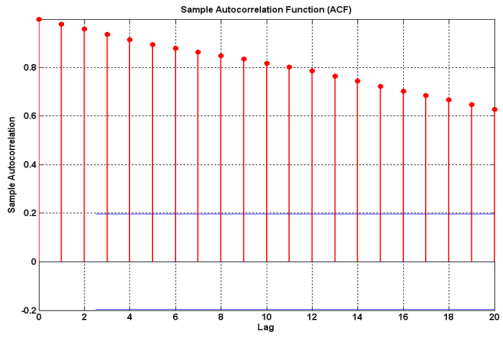


Figure B.9: ACF plot for daily stock (closing) prices of IBM Corporation

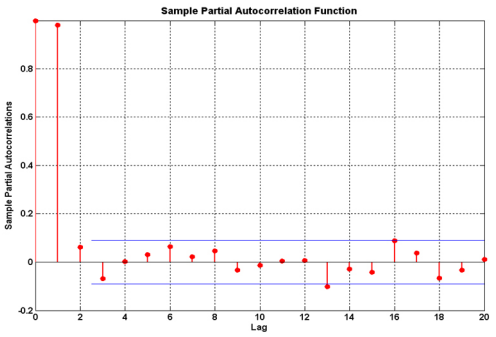


Figure B.10: PACF plot for daily stock (closing) prices of IBM Corporation

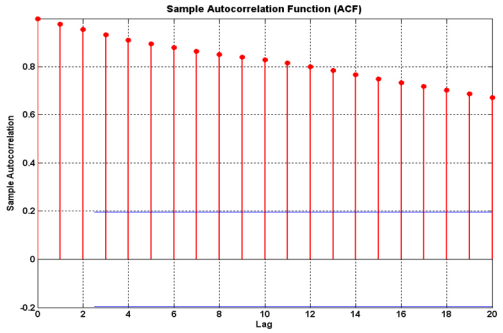


Figure B.11: ACF plot for daily stock (closing) prices of Dell Inc.

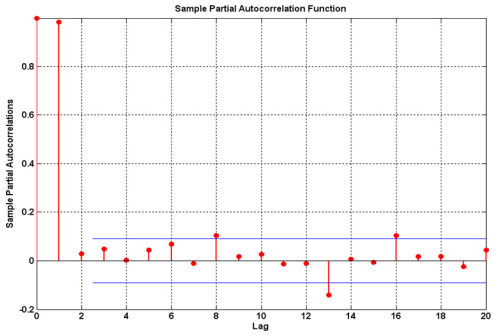


Figure B.12: PACF plot for daily stock (closing) prices of Dell Inc.

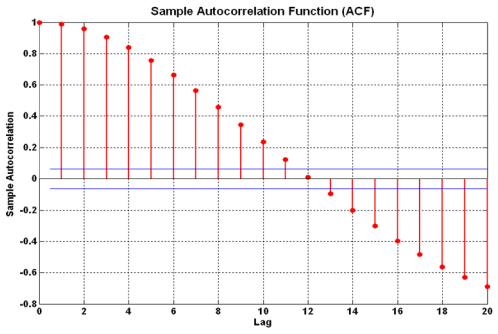


Figure B.13: ACF plot for Mackey-Glass dataset

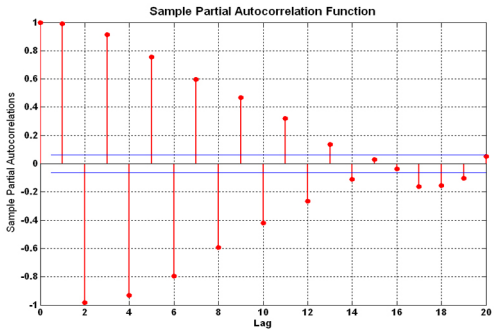


Figure B.14: PACF plot for Mackey-Glass dataset

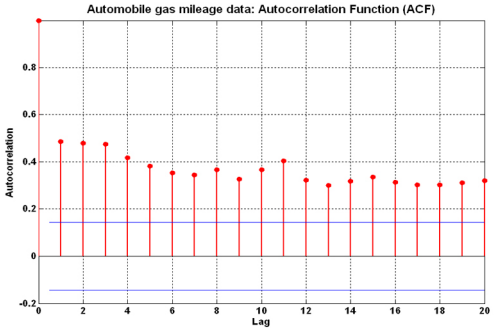


Figure B.15: ACF plot for Automobile Gas Mileage data

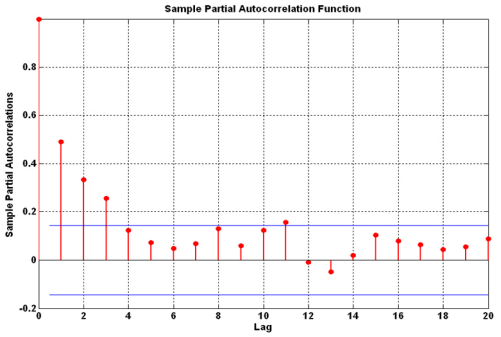


Figure B.16: PACF plot for Automobile Gas Mileage data

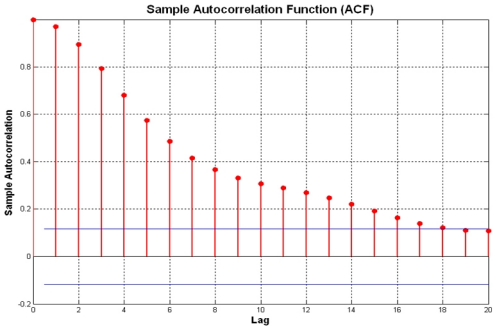


Figure B.17: ACF plot for Box-Jenkins Gas Furnace data

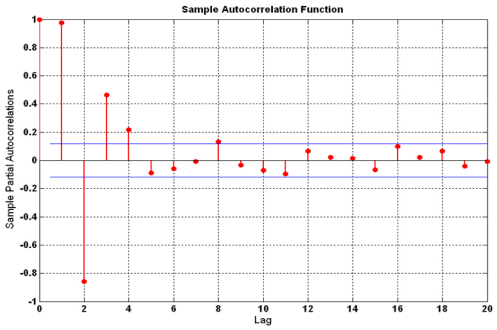


Figure B.18: PACF plot for Box-Jenkins Gas Furnace data

Appendix C

Non-linearity test of the financial data

In this section, we provide the test results for the existence of non-linearities in the financial dataset that we have used in this thesis. We follow the methodologies that have been used in the study by Das and Das [216]. We have used the surrogate data method proposed by Theiler *et al.* [217]. In this approach the data signal is produced from the given data signal by changing its phase randomly, so that the mean and variance of both the given signal and obtained signal remain the same. Being generated by randomizing the phases, a variety of surrogate signals for a given signal can be produced. To generalize, we have generated five surrogate signals for each of the financial time series data denoted as S_1, S_2, S_3, S_4 and S_5.

Lyapunov exponent is another indicator that refers to the chaotic characteristics of a dataset. Das and Das used the Largest Lyapunov Exponent (LLE) to identify the underlying chaotic behaviour as well as non-linearity in time series data. For further details of this approach readers are referred to [216].

Following Das and Das, we have attempted to quantify the nonlinearity using surrogate signals and obtained LLE values corresponding to the signals. The difference in the LLE values between the original signal and the surrogate signals are apparent from the plots (see Fig. C.1, C.2, C.3, C.4, C.5, C.6). According to Das and Das, a large difference in the LLE values implies the degree of non-linearity as well as chaos in the dataset.

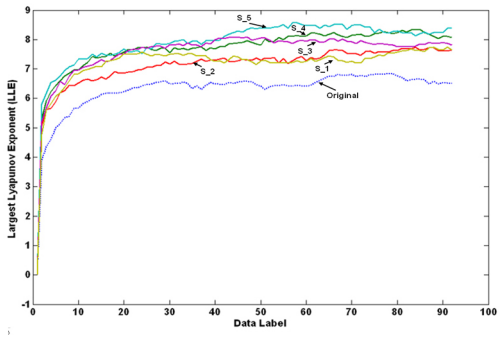


Figure C.1: Prediction error versus length of prediction for British Airlines to estimate LLE for original data (dotted line) and its surrogate sets

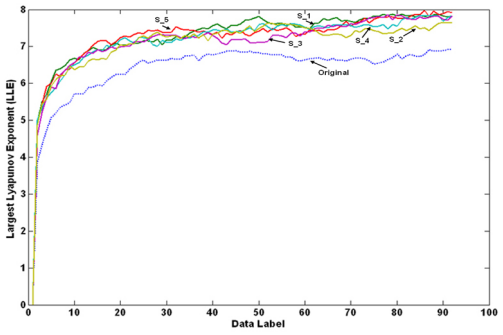


Figure C.2: Prediction error versus length of prediction for Delta Airlines to estimate LLE for original data (dotted line) and its surrogate sets

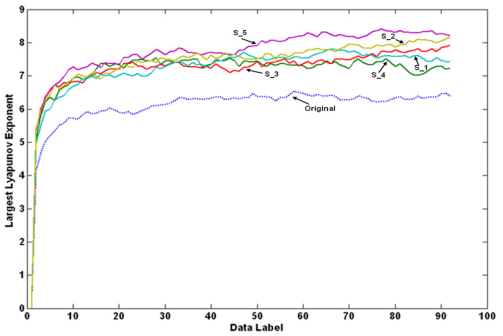


Figure C.3: Prediction error versus length of prediction for Ryanair Airlines to estimate LLE for original data (dotted line) and its surrogate sets

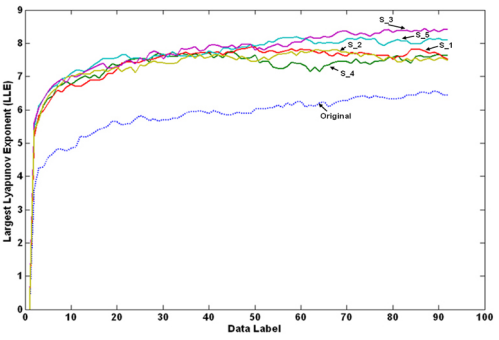


Figure C.4: Prediction error versus length of prediction for Apple Computer Inc. to estimate LLE for original data (dotted line) and its surrogate sets

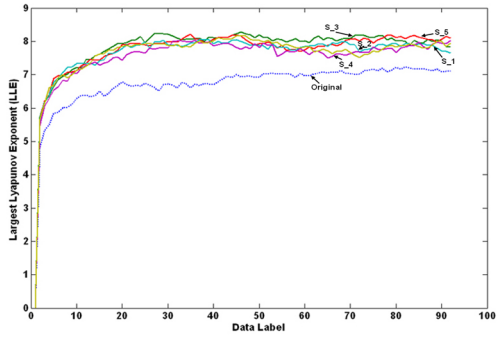


Figure C.5: Prediction error versus length of prediction for IBM Corporation to estimate LLE for original data (dotted line) and its surrogate sets

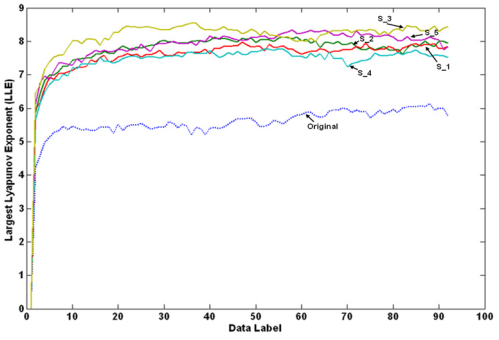


Figure C.6: Prediction error versus length of prediction for Dell Inc. to estimate LLE for original data (dotted line) and its surrogate sets

Appendix D

The parameter values of the evolutionary algorithm

Table D.1: The GA parameters in the fusion model

Parameter for optimization in the HMM	GA parameter name	GA parameter values
Emission probability distribution function	Chromosome size	16
	Polpulation type	Double
	Population size	20
	Elite parent selection	2
	Crossover fraction	0.8
	Migration fraction	0.2
	Generations	100
	Fitness Limit	-infintity
	Initial population	Random
	Fitness scaling	Rank scaling
	Selection	Stochastic uniform
Prior probability matrix	Chromosome size	4
	Polpulation type	Double
	Population size	20
	Elite parent selection	2
	Crossover fraction	0.8
	Migration fraction	0.2
	Generations	100
	Fitness Limit	-infintity
	Initial population	Random
	Fitness scaling	Rank scaling
	Selection	Stochastic uniform
Transition probability matrix	Chromosome size	16
	Polpulation type	Double
	Population size	20
	Elite parent selection	2
	Crossover fraction	0.8
	Migration fraction	0.2
	Generations	100
	Fitness Limit	-infintity
	Initial population	Random
	Fitness scaling	Rank scaling
	Selection	Stochastic uniform