FUZZY SYSTEM APPROACHES FOR DATA STREAMS AND FUNCTIONAL DATA REGRESSION

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2019

By
Dongjiao Ge
Department of Computer Science
# Contents

Abstract ............................................................... 15  
Declaration ............................................................... 17  
Copyright ................................................................. 18  
Publications .............................................................. 19  
Acknowledgements ..................................................... 20  

1 Introduction .......................................................... 21  
  1.1 Context and Motivation .............................................. 21  
  1.2 Research Questions .................................................. 25  
  1.3 Research Objectives ............................................... 26  
  1.4 Contributions ....................................................... 27  
  1.5 Thesis Structure ................................................... 29  

2 Background and Related Work ....................................... 32  
  2.1 Chapter Introduction ................................................. 32  
  2.2 Evolving Fuzzy Systems — Streaming Data Regression Approaches ........................................... 32  
    2.2.1 Fuzzy Sets and Membership Functions ...................... 33  
    2.2.2 Fuzzy Systems ................................................... 35  
    2.2.3 Evolving Fuzzy Systems ....................................... 42  
    2.2.4 Critical Analysis ................................................. 58  
  2.3 Functional Data Regression ........................................ 63  
    2.3.1 Functional Analysis Basics .................................... 64  
    2.3.2 Data Representing, Statistics ................................. 65  
    2.3.3 Scalar-on-Function Regression ................................. 68
2.3.4 Function-on-Scalar Regression ........................................ 73
2.3.5 Function-on-Function Regression ..................................... 76
2.3.6 Critical Analysis .......................................................... 82
2.4 Chapter Summary .............................................................. 85

3 Evolving Fuzzy Systems: Optimum Approaches .......................... 86
3.1 Chapter Introduction .......................................................... 86
3.2 Learning Evolving Mamdani Fuzzy Systems Based on A Local Online Optimization Approach ............................................. 87
  3.2.1 Problem Statement ....................................................... 87
  3.2.2 EMFSPO Online Learning Method .................................... 89
  3.2.3 Mechanism of EMFSPO ................................................. 94
  3.2.4 Numerical Examples .................................................... 95
3.3 LEOA: Learning Evolving T-S Fuzzy Systems Based on A Local Error Optimization Approach ................................................... 100
  3.3.1 Problem Statement ....................................................... 100
  3.3.2 LEOA Learning ............................................................ 101
  3.3.3 Mechanism of LEOA ..................................................... 110
  3.3.4 Numerical Examples .................................................... 112
3.4 Chapter Summary .............................................................. 121

4 Evolving Fuzzy Systems: Self-Learning Approaches for Dynamic Thresholds ......................................................... 123
4.1 Chapter Introduction .......................................................... 123
4.2 Problem Statement ............................................................ 124
4.3 A Self-Learning Approach with Dynamic Rule Generation Threshold ............................................................... 126
  4.3.1 Incremental Learning Algorithm of SEFS ............................ 126
  4.3.2 Numerical Results ....................................................... 137
4.4 A Self-Learning Approach for Dynamic Rule Generation & Reduction Thresholds ............................................................... 143
  4.4.1 Our Approach ............................................................. 144
  4.4.2 Learning Approaches of EFS-SLAT ................................. 145
  4.4.3 Numerical Examples .................................................... 152
4.5 Chapter Summary .............................................................. 160
7.1.3 Functional Fuzzy System — An Offline Function-on-Function Regression Approach .................................................. 198
7.1.4 Evolving Functional Fuzzy System — An Online Function-on-Function Regression Approach ............................. 198
7.2 Future Work ................................................................. 199

A Mathematical Inductions, Proofs and Additional Numerical Results 202
A.1 Mathematical Proof in Chapter 3 ....................................... 202
A.1.1 Proof of Theorem 3.1 .................................................... 202
A.1.2 Proof of Theorem 3.2 .................................................... 202
A.1.3 Proof of Theorem 3.3 .................................................... 203
A.1.4 Proof of Theorem 3.4 .................................................... 203
A.1.5 Proof of Theorem 3.5 .................................................... 203
A.1.6 Proof of Theorem 3.6 .................................................... 204
A.1.7 Deduction of Formula (3.68) ........................................... 204
A.2 Mathematical Proof in Chapter 4 ....................................... 205
A.2.1 The Proof of Proposition 4.1 ......................................... 205
A.2.2 Induction of (4.8) and (4.9) in rule updating method .......... 205
A.2.3 Induction of (4.20) and (4.21) in rule merging method ........ 206
A.2.4 The Proof of Theorem 4.1 .............................................. 207
A.2.5 Induction of VFF-WRLS ............................................... 208
A.3 Mathematical Proof in Chapter 6 ....................................... 208
A.3.1 Induction and Numerical Computation of $L^2$ Norms in (6.6) .... 208
A.3.2 Induction of (6.13), (6.14) and (6.15) ............................. 209
A.3.3 Calculate The Integrals in (6.13), (6.14) and (6.15) .......... 210
A.4 Additional Numerical Results in Chapter 5 .......................... 210
A.4.1 Complexity Analysis of LEOA ....................................... 210
A.4.2 Numerical Results for EFFS .......................................... 211

B Benchmark Examples & Datasets 218
B.1 Benchmark Examples & Datasets for Evolving Fuzzy Systems .... 218
B.2 Benchmark Examples & Datasets for Functional Fuzzy Systems .... 221

Bibliography 225
List of Tables

3.1 RMSEs of example 1 ......................................................... 97
3.2 RMSEs of example 2 ......................................................... 98
3.3 Sensitivity of predefined thresholds to online testing accuracy. ...... 113
3.4 Sensitivity of predefined thresholds to global accuracy. ............... 113
3.5 RMSEs of example 1 ......................................................... 114
3.6 RMSEs of example 2 ......................................................... 116
3.7 RMSEs of example 3 ......................................................... 118
3.8 NDEIs of example 4 ....................................................... 121

4.1 Sensitivity analysis (Example 1 in section 4.3.2.2). ....................... 137
4.2 Example 1: Nonlinear dynamic plant identification with time varying characteristic. .................................................. 139
4.3 Example 2: Mackey-Glass Chaotic time series (long term prediction) .. 140
4.4 Example 3: Online prediction of S&P 500 daily closing price. ......... 142
4.5 Example 1: S&P 500 daily closing prices. ................................ 155
4.6 Example 2: Mackey-Glass Chaotic time series. ............................. 156
4.7 Example 2: Online testing results for all the data. ......................... 156
4.8 Example 3: Delta Ailerons data. ........................................ 157
4.9 Example 3: Online testing results for all the data. ......................... 158
4.10 Example 4: Helicopter unmanned aerial vehicle streaming data. ....... 158
4.11 Example 4: Online testing results for all the data. ......................... 159
4.12 Example 5: MPG and Boston Housing data. ............................. 159

5.1 Example 2: DTI data prediction. ........................................ 173
5.2 Example 3: Capital bike share data prediction. ........................... 175
5.3 Example 4: Forecast the electricity consumption of France. .......... 176
5.4 Example 1: Simulation example. ........................................ 178
6.1 Example 1: Result comparison of different methods.  
6.2 Example 1: Online learning MSPEs. 
6.3 Example 2: Result comparison of different methods. 
6.4 Example 2: Online learning MSPEs. 
6.5 Example 3: Online learning MSPEs (Rule Num.) for DIA with Basis Num. = 10. 
6.6 Example 3: Online learning MSPEs (Rule Num.) for DIA with Basis Num. = 20. 
6.7 Example 3: Online learning MSPEs (Rule Num.) for QQQ with Basis Num. = 10. 
6.8 Example 3: Online learning MSPEs (Rule Num.) for QQQ with Basis Num. = 20. 
A.1 Example 1: Testing MSPEs with 50 pairs for training (spline order = 2). 
A.2 Example 1: Online testing MSPEs (spline order = 2). 
A.3 Example 1: Average number of rules (spline order = 2). 
A.4 Example 1: Testing error with 50 pairs for training (spline order = 3). 
A.5 Example 1: Online testing MSPEs (spline order = 3). 
A.6 Example 1: Average number of rules (spline order = 3). 
A.7 Example 1: Testing errors with 50 pairs for training (spline order = 4). 
A.8 Example 1: Online testing MSPEs (spline order = 4). 
A.9 Example 1: Average number of rules (spline order = 4). 
A.10 Example 2: Testing errors with 50 pairs for training (spline order = 2). 
A.11 Example 2: Online testing MSPEs (spline order = 2). 
A.12 Example 2: Average number of rules (spline order = 2). 
A.13 Example 2: Testing errors with 50 pairs for training (spline order = 3). 
A.14 Example 2: Online testing MSPEs (spline order = 3). 
A.15 Example 2: Average number of rules (spline order = 3). 
A.16 Example 2: Testing errors with 50 pairs for training (spline order = 4). 
A.17 Example 2: Online testing MSPEs (spline order = 4). 
A.18 Example 2: Average number of rules (spline order = 4).
List of Figures

1.1 Summary of contributions. ........................................... 29
2.1 The framework of a fuzzy system. ............................... 36
2.2 The architecture of ANFIS. ........................................... 41
2.3 The architecture of SEIT2FNN. .................................... 42
2.4 General framework of an EFS learning approach. .......... 44
2.5 The general framework of an EFS learning approach. ...... 59
3.1 Online identification absolute errors for training data in example 1. .... 97
3.2 Simulation results for example 2. ............................... 99
3.3 Computational steps of LEOA. ..................................... 110
3.4 Example 1: predictions for all historical data using $M_2$ (3.85). ........ 115
3.5 Example 2: prediction results for testing data. ............... 116
3.6 Example 2: prediction errors for all historical data. .......... 117
3.7 Example 3: prediction results for testing and all historical data. ...... 118
3.8 Example 4: error trace. ............................................. 121
4.1 The flowchart of SEFS. ............................................. 136
4.2 Online identification results in example 1. ($t \in [900,2100]$) ........ 139
4.3 Online prediction errors of all 3500 data points in example 2. ......... 141
4.4 Online prediction results for example 3. ....................... 142
4.5 Prediction errors for example 3. .................................. 143
4.6 Underfitting vs overfitting. ........................................ 145
4.7 Online testing errors. ............................................... 153
4.8 Trajectories of the thresholds. .................................... 154
5.1 The framework for identifying an FFS. ......................... 165
5.2 The training and testing procedure of an FFS. ................. 170
5.3 Simulated coefficients of inputs data. ................................. 172
5.4 Centres and consequents of rules. ................................. 174
5.5 Centres and consequents of rules. ................................. 175
5.6 Simulated inputs data. ....................................... 179
5.7 Simulated input-output pairs and outputs. ................. 180

6.1 The framework of EFFS. ....................................... 188
6.2 Centres and consequents of rules. ................................. 191

B.1 DTI data. .............................................. 223
B.2 Casual bike users and hourly temperature. ................. 223
B.3 The daily temperature and electricity consumption of France in 2017. . . . 223
# Abbreviations & Nomenclature

## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFFS</td>
<td>Evolving Functional Fuzzy System</td>
</tr>
<tr>
<td>EFS-SLAT</td>
<td>EFS which has Self-Learning/Adaptive Thresholds</td>
</tr>
<tr>
<td>EFSs</td>
<td>Evolving Fuzzy Systems</td>
</tr>
<tr>
<td>EFS</td>
<td>Evolving Fuzzy System</td>
</tr>
<tr>
<td>EMFSPO</td>
<td>Evolving Mamdani Fuzzy Systems Based on Parameter Optimization</td>
</tr>
<tr>
<td>EMSE</td>
<td>Excess Mean Square Error</td>
</tr>
<tr>
<td>ERLS</td>
<td>Extended Recursive Least Squares</td>
</tr>
<tr>
<td>EWRLS</td>
<td>Extended Weighted Recursive Least Squares</td>
</tr>
<tr>
<td>FDA</td>
<td>Functional Data Analysis</td>
</tr>
<tr>
<td>FFS</td>
<td>Functional Fuzzy System</td>
</tr>
<tr>
<td>FLM</td>
<td>Functional Linear Model</td>
</tr>
<tr>
<td>FPCA</td>
<td>Functional Principal Component Analysis</td>
</tr>
<tr>
<td>LEOA</td>
<td>Local Error Optimization Approach</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
</tbody>
</table>
**MLE**  Maximum Likelihood Estimation

**MLP**  Multi-Layer Perceptron

**MSPE**  Mean Squared Prediction Error

**NDEI**  Non-Dimensional Error Index

**RBFN**  Radial Basis Function Network

**RKHS**  Reproducing Kernel Hilbert Space

**RLS**  Recursive Least Squares

**RMSE**  Root Mean Square Error

**SEFS**  Self-Evolving Fuzzy System

**SGD**  Stochastic Gradient Decent

**SVM**  Support Vector Machine

**VFF-RLS**  Recursive Least Squares algorithm with Variable Forgetting Factor

**VFF-WRLS**  Weighted Recursive Least Squares algorithm with Variable Forgetting Factor

**WRLS**  Weighted Recursive Least Squares

**Nomenclature**

\[ \alpha, \xi_i \]  threshold used in VFF-RLS, a certain threshold

\[ \beta_1, \beta_2, M_0, M \]  parameters used by VFF-RLS

\[ e^{(\alpha)}, e^{(\rho)}, M \]  threshold that controls rule generation, threshold that controls rule pruning, the threshold of the age

\[ \varepsilon_0, k_0 \]  other predefined parameters used in rule generation
\( \varepsilon, \varepsilon_{\sigma}, \varepsilon^s, \varepsilon^*_{\sigma}, \varepsilon^{(m)} \) thresholds that control rule merging

\( \varepsilon_t^{(a)}, \varepsilon_{\max}, \varepsilon_{\min}, \varepsilon_t^{(m)} \) dynamic threshold that controls rule generation, upper bound of \( \varepsilon_t^{(a)} \), lower bound of \( \varepsilon_t^{(a)} \), dynamic threshold that controls rule merging

\( \Gamma_{i,j} \) fuzzy sets

\( \gamma(x), \theta(x) \) firing strength of rule \( R_i \), the normalized firing strength of rule \( R_i \)

\( \hat{\gamma}_{\text{train}} \) estimated training output

\( k, \pi_{\max}, \pi_{\min} \) the number of samples for a particular parameter, upper bound of RMSEs, lower bound of RMSEs

\( x, xe, y, \hat{y}, y_i, n \) input, generalized input \( xe = (1, x) \), output, predicted output, the output of rule \( R_i \), dimension of input \( x \)

\( \mu_{i,j}(x_j) \) the membership degree for \( x_j \) in \( \Gamma_{i,j} \)

\( \tau \) stopping threshold for functional approximation using basis function

\( \theta_i(t), \gamma_i(t), K_i(t) \) firing strength of rule \( R_i \) at time \( t \), the normalized firing strength of rule \( R_i \) at time \( t \), rule number at time \( t \)

\( \rho_f, \rho, \iota, \nu_t, \) noise-free signal, noisy signal, white Gaussian noise signal, Gaussian noise signal

\( A(t), e_k, \lambda, \lambda, U_i \) cumulative online training error, online training error, forgetting factor, time-varying forgetting factor, utility of rule \( R_i \)

\( B_p^Y(s), B_q^Y(t), \alpha_p, \beta_q \) basis functions for approximate input \( x(s) \) and output \( y(t) \), coefficients of these basis when approximating \( x(s) \) and \( y(t) \)

\( c_{i,j}, \sigma_{i,j}, \psi_i \) centres, radii, consequent parameters of EMFSPO, LEOA, SEFS, and EFS-SLAT

\( e_{i}(t), e'_i(t) \) prior error signal, noise-free prior error signal
$h_{kd}$ consequent parameters of the FFS and EFFS

$ind_a, ind_m$ rule generation indicator, rule merging indicator

$N_i(t)$ the number of data points in the central part of rule $R_i$ at time $t$

$R_i, K$ the $i$-th fuzzy rule, rule number

$S(\cdot, \cdot), D(\cdot, \cdot), B(c, \sigma)$ similarity degree, distance, multidimensional open ball with centre $c$ and radius $\sigma$
Abstract

FUZZY SYSTEMS APPROACHES FOR DATA STREAMS AND FUNCTIONAL DATA REGRESSION
Dongjiao Ge
A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy, 2019

With the arrival of the data explosion era, data modelling/prediction has become one of the most important research areas. Many datasets in our real-life are essentially data streams and functional data. This thesis investigates fuzzy system-based regression approaches for these two representative datasets. In line with the data types, this thesis is divided into two scenarios.

The first part focuses on the data stream regression problem with input/output being real vectors. It is widely known that evolving fuzzy systems (EFSs) are effective approaches for solving data stream regression problems, in that EFSs are structurally self-organized, capable of updating structure/parameters in an online manner, and acting as the one-pass approaches without the requirement of storing historical data. However, current state-of-the-art studies indicate that the existing evolving structure/parameter approaches would impose a negative impact on the optimality of EFSs. To our best knowledge, research on proposing optimal EFSs was still rare. Furthermore, selecting predefined thresholds to control the structure/parameters evolution of EFSs is of importance, which has not been systematically investigated thus far. In this part, this thesis focuses on addressing the aforementioned two EFSs-related problems, which might provide implications for the research on the data stream. From an optimality viewpoint, EFS learning approaches for both Takagi-Sugeno and Mamdani fuzzy systems, that is, local error optimization approach (LEOA) and identifying evolving Mamdani fuzzy systems from the parameter optimization aspect (EMFSPO), are proposed. To automatically tuning the thresholds, two approaches, i.e. the self-evolving fuzzy system (SEFS), and an
extended work based on SEFS, that is, EFS with self-learning/adaptive thresholds (EFS-SLAT), are proposed. Finally, through a wide range of benchmark examples, LEOA, EMFPSO, SEFS, and EFS-SLAT are shown to be capable of improving the accuracy compared with many state-of-the-art approaches.

The second part of this thesis focuses on the functional data regression problem, where inputs/outputs are functions. Current techniques for functional data regression are mainly statistical models with certain assumptions and restrictions for a specific model. Since the kernel functions should be used in integral in these approaches, the resulting models were lack of interpretability and difficult to understand especially for the nonlinear case. More importantly, they were mainly offline models, which were not suitable to handle big functional data/functional streaming data considering the computational cost for storing a vast of historical information during model analysis. To improve these weaknesses, a pioneering functional regression approach, i.e. the functional fuzzy system (FFS), is developed. It acts as an operator between two functional spaces. FFS is an offline nonlinear regression model enabling its inputs/outputs to be functions. It has a flexible structure that can explain any nonlinear relationship between functions using an interpretable “If-Then” fuzzy inference. In order to make the model applicable, the corresponding learning approach is proposed for identifying FFS. Since FFS is an offline model, it is not suitable for big functional datasets and functional streaming data. To overcome such bottlenecks, the first functional real-time regression model known as evolving functional fuzzy system (EFFS) is developed. EFFS is endowed with an exceptional advantage of learning from data online in addition to inheriting all the superior characteristics of FFS. EFFS provides a potential approach for handling big functional data by processing functional streaming data with each data point being a function. Finally, it demonstrates that FFS and EFFS outperform many of the state-of-the-art approaches through comparison experiments on various benchmark examples. These numerical results verify the effectiveness of FFS and EFFS.
Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.library.manchester.ac.uk/about/regulations/) and in The University’s policy on Presentation of Theses.
Publications

Journal Papers


Book Chapters


Conference Papers

Acknowledgements

First of all, I would like to give my greatest thanks to my supervisor Prof. Xiao-Jun Zeng for his constant support and supervision during my Ph.D. career. It is my great fortune to be Prof. Zeng’s Ph.D. student. Besides, I would also thank my supervisor for his kind help when I have difficulties. I would sincerely thank my co-supervisor Dr. Ke Chen for his support and guidance.

I would express my deepest thanks to my father Mr. Xiaotian Ge and my mother Ms. Yajun Gao. Thank you for standing by my side, supporting me and encouraging me whenever I meet difficulties. I am grateful to my husband Dr. Pengzhan Sun. He has given me so much wonderful memories.

Many thanks to all the Machine Learning & Optimization (MLO) group members and my friend Dr. Yangmei Li and Ms. Yuan Zhao from the School of Physics. I would like to thank Dr. Fanlin Meng for his helpful suggestions during my PhD.

Last but not least, I would sincerely thank the President’s Doctoral Scholar Award of the University of Manchester for the financial support during my Ph.D. I give my great thanks to my colleagues, Ms. Yuan Chai and Mr. Hashir Kiani, from MLO group, for their great help on professional proofreading. I also want to thank Ms. Rosemary Goodier from The Main Library, University of Manchester, for English language proofreading.
Chapter 1

Introduction

1.1 Context and Motivation

When first reading the title of this thesis, one would naturally have several questions on his (or her) mind. What are data streams and functional data? Why are they worth being studied? Why are fuzzy systems meaningful in regression problems of these two kinds of data?

In the following content of this section, it answers these questions in three aspects.

1. **What are data streams? Why are they important?**

In a formal sense, a data stream is known as a continuous flow of data, which is usually unbounded, coming rapidly and often accompanied with nonstationary (evolving or concept drift) phenomena. Many examples of data streams can be found in our daily life, for example, data generated by web applications, signals from telecommunication systems, records on social softwares, transactions on stock exchanges, records on surveillance systems, image and signals from the satellites, etc. Generally, data streams are usually studied in a finite space $\mathbb{R}^n$. There are two main types of data streams, complex and big data streams, which need to be tackled.

- Complex data streams are usually time varying and nonstationary data. Learning nonstationary data is referred to as learning in evolving or dynamic environment. Unlike the stationary data, the underlying distribution and the
parameters such as mean and variance are no longer fixed and changing over time [1]. Furthermore, seasonal and periodical effects may be blended in the dataset.

- The big data streams need to have the features of both data streams and big data. Big data is swiftly increasing in its amount so as to that traditional data mining and database mechanisms cannot meet the requirement of typical big data analysis, processing and storage [2]. Big data is characterized by five “V”s, volume, variety, velocity, value as well as veracity [3], [4]. To be more specific, the size of the dataset is huge, the data itself has no fixed structure and always comes in a high speed, and analyzing the data requires new value in application.¹

As data streams are closely related to our daily life, it is essential to study their mining techniques. Mining data streams is never an easy job. As the data streams are unbounded and nonstationary, the traditional mining techniques, which require the storage of loads of historical data, lack the ability to face the new challenges in terms of data storage, data management, and processing speed. Therefore, it is important to study effective data stream mining approaches that can: (1) Act as one-pass approaches that discard the data points after they have been processed, or store a part of data for a certain period and discard them later; and (2) React to the nonstationary phenomenon and perform consistently when the underlying distributions of the data are changed.

2. What are functional data and why are they important?

Functional data are a set of data with each data point being a functional (or a function). Functional data should be studied in a functional space, which is usually infinite dimensional with each data point being a functional. As data are observed discretely, the observations of a “data point” of the functional data are formed by the observations on the curve of the corresponding function. To give a better explanation, the dataset of high-frequency stock transactions is taken as an example. Considering there is a daily curve formed by the transaction volumes of every minute in a particular day, the observations on this curve can be treated as a daily data stream. However, when considering all the transaction volumes daily curves within several days, one month, or for a longer time, it is reasonable to regard them as functional data with each “data point” being a daily volume curve.

¹Complex data streams do not need to be big data streams.
Such as data streams, there also exists different types of functional data, e.g., functional streaming data and big functional data. To be more specific, functional streaming data are a flow of data stream with each data point a functional data, and a big functional dataset is the functional dataset that has extremely large sample size. Another example is the analog signal. Analog signal in every minute is a continuous function. When we regard the analog signal in every minute as a functional data point, the every-minute analog signal curves we get in the whole morning form a functional streaming data. Considering the 10-year historical data of the every-minute curves of a certain analog signal, we can get a big functional dataset. Because many data in research areas, such as medicine, biology, economic, and engineering, which have their generating mechanism are intrinsically stochastic functions [5], it is reasonable and important to model them as functional data instead of discrete points. Because most researchers with regard to functional data are from research community of statistics, research related to functional data is currently recognized as a branch of statistics known as functional data analysis (FDA), which includes statistical theory and models for analyzing the functional data. Noticing the characteristics of the functional data, FDA studies would face challenges from the following aspects:

- As stated in [6], one of the biggest challenges is the dimension reduction of the functional data, as many of the functionals should be considered in an infinite dimension that is far beyond the processing ability of the advanced technologies. Furthermore, the statistical tools used in a finite space $\mathbb{R}^n$ are unlikely to be able to work in an infinite functional space, due to the inherent difference between these two types of spaces.

- Most statistical methods are applied under assumptions of models or data. For example, the model assumes the input and output follows a linear relationship, or the residuals follow a Guassian distribution, etc. However, with the dynamics of the data, it is hard to make a judgement as to what types of model the data should follow; and it is also difficult to check whether the data satisfy the conditions required by a certain statistical tool.

- Reviewing the existing FDA research works, the study of functional streaming data still remains blank. Existing functional data regression models are offline models, which have slow speed and always require large memory
space. Therefore, instant application of the existing models is almost impossible for functional streaming data and big functional data.

Even though a great number of studies have been made on functional data, there is still a long journey in order to tackle all the possible challenges in FDA research.

3. Why fuzzy systems are meaningful regression tools for data streams and functional data?

Constructed on fuzzy logic proposed by Zadeh [7], fuzzy systems are mathematical models that contain linguistic rules, “If-Then” fuzzy rules, in order to mimic human thinking and reasoning. Fuzzy systems are interpretable models, taking the advantages of the “If-Then” reasoning process. Coming back to the regression problems that this thesis is concentrated on, the reasons for this research to use and propose fuzzy system based approaches can be illustrated by the following points:

- Fuzzy systems are interpretable models. They can model any nonlinear relationships in a flexible and interpretable way. To be more specific, fuzzy systems can represent the complex nonlinear relationships using a collection of “If (reason), Then (result)” fuzzy rules. These fuzzy rules assist to explain the invisible knowledge in a natural reasoning processing. It is understandable that some reasons cause some results.

- Fuzzy system based online approaches have been widely considered by data stream regression. These fuzzy systems are usually evolving fuzzy systems (EFSs). Generally, EFSs have been proposed as one-pass approaches, which have a low requirement for the memory and storage of data, thus to fit the online manner of the data streams. On the one hand, EFSs are equipped with a real-time (i.e. online) evolving structure that can capture the data dynamics and concept drifts on the fly. On the other hand, EFSs can present the knowledge learnt online in an accurate, transparent and interpretable way [8].

- To the best of our knowledge, fuzzy systems have never been studied for solving functional data regression problems. However, the theory and frameworks of fuzzy systems have their own priority and potential to be powerful tools of FDA. The flexible structure of fuzzy systems enables them to be plug-and-play approaches for a broad range of data. Furthermore, fuzzy systems act as universal approximators and have no statistical assumptions
which need to be checked. This enables fuzzy systems to get accurate results for diverse types of data. Moreover, fuzzy systems are nonlinear models represented by a set of simple “If-Then” fuzzy rules, which provides us a great help for modelling the complex nonlinear relationships between data in an interpretable way. Lastly, EFS frameworks can be borrowed and adopted to seek possible approaches for coping with large-scale functional data regression problems.

Regarding the application background of these important data categories (data streams, functional data, and functional streaming data), the following chapters of this thesis focus on: (i) Presenting the proposed EFSs learning/identification methodologies that can properly deal with the unsolved problems of EFSs learning approaches; and (ii) Presenting the proposed functional data regression models, which are flexible, easy to implement, or evolving, for both small and large-scale functional datasets.

1.2 Research Questions

Considering the research background and motivations shown in Section 1.1, the general research idea and a rough direction of this research can be indicated. However, based on the existing literature on EFSs and functional regression, the ideas about the exact topics that would be investigated are inspired by considering the following research questions for the two types of target data — data streams and functional data:

For data streams regression problem, the research questions came out from the regression models — EFSs. Due to EFSs are online evolving models, EFSs need to be learnt by certain online learning approaches. Furthermore, EFSs online learning approaches are the soul of EFSs. Therefore, the first question would be “What are the limitations and unsolved problems for EFSs online learning methods?” This requires us to fully analyze the existing studies and be sensitive to the crucial research weaknesses. Furthermore, a natural question is “How to manage and avoid these limitations?”. Moreover, a further question to be asked is that “Can the proposed approaches achieve better performance compared with the state-of-the-art methods?”. Moving on to the functional data regression: The very first question is “How to develop an interpretable model that can solve nonlinear regression problems on the
general functional spaces?”. Noticing the advantages of the fuzzy systems, this thesis attempts to solve regression problems for functional data under the framework of fuzzy systems. However, the traditional fuzzy systems are defined on $\mathbb{R}^n$; Therefore, a question arises: “Is it possible to develop a new type of fuzzy system that has both input and output being functionals from the general functional spaces? and How to identify the parameters of the systems?”. To answer these questions, the mathematical model of fuzzy system on functional space should be given, and under which the algorithm needs to be proposed in order to learn the model parameters. Furthermore, big data issues are recognized to be tough problems and this inspires us to investigate the research question, “Can fuzzy systems be further extended to have the potential to process big functional datasets and functional streaming datasets?”.

1.3 Research Objectives

Pointing to data streams and functional data regression issues, the overall aims are: (i) To discuss the research gaps of the existing EFSs learning method and to propose advanced learning approaches; (ii) To investigate the possibility for fuzzy systems to be applied for functional data regression, and to discuss whether fuzzy systems are promising approaches for big functional data and functional streaming data. To be more specific, the research objectives of this thesis can be divided into several small sections:

- **Propose EFS online learning methods with local optimality for nonstationary data streams.**
  The structure change of the EFS would influence the optimality of EFS. In order to cope with this problem, EFS learning methods should be inducted to serve the aim of optimality. This procedure should be completed through a rigorous mathematical proof and induction.

- **Propose EFS online learning methods with self-learning control parameters (or thresholds) for nonstationary data streams.**
  The determination of the right thresholds for EFS is a very crucial problem that has not been effectively solved. It is a difficult task to select the uniform thresholds that are suitable for a huge number of datasets. To realize this objective, a self-learning strategy for the thresholds, which controls the structure or parameters evolving of the system, should be proposed.
• **Develop a functional fuzzy system model along with its learning approach for the functional regression problem.**

Model interpretability is left to be a difficult problem for the functional regression models. Taking the advantages of the interpretability and the simple model structure of fuzzy systems, this thesis aims at developing a new type of fuzzy system with inputs and outputs being functionals in functional spaces. Furthermore, formulas and methods for learning the parameters of the new fuzzy system should be induced and studied.

• **Develop an online fuzzy system model and its learning approach to solve the regression problem of functional streaming data and big functional data.**

Functional streaming data requires the functional regression model for processing the data in an online manner. As streaming processing is recognized to be an efficient theory to handle the big functional data, it can be considered the regression problems of these data to be the same one. Inspired by the online learning models, EFS, in $\mathbb{R}^n$, this thesis would like to develop a new type of EFS model for learning the functional data in an online manner. Furthermore, the corresponding learning approach, which includes how to learn the system structure and parameters of the developed functional regression model dynamically, should be proposed.

### 1.4 Contributions

This section summarizes the research contributions that have been made by this thesis. Fuzzy system based regression models have been investigated for solving the regression problems of data streams and functional data. On one hand, two major classes of EFSs online learning approaches have been proposed for data stream regression problems; On the other hand, the fuzzy system based regression models have been developed for solving the functional data regression problems. The following bullet gives a more specific introduction to the research contributions: (i) and (ii) summarize contributions of this thesis upon the data stream regression approaches—EFSs; and (iii) and (iv) give a brief summary for the contributions made by this thesis on the fuzzy system based functional data regression models.

(i) The local optimal EFSs (Mamdani and Takagi-Sugeno type) learning approaches, which are evolving Mamdani fuzzy systems from the parameters optimization
(EMFSPO) and local error optimization approach (LEOA), have been proposed to cope with the shortcomings that most of the existing EFSs learning methods have:

(1) Developed based on a human heuristic manner, rather than an optimal approach, putting the main focus on tracking the most recent local model, thus leading to an “unlearning effect” and often poor global accuracy; (2) Lack of optimality of the consequent parameters when there is a structure update of the fuzzy system. The proposed methods can address these issues by having their antecedent learning method derived from minimizing a bunch of local error functions, and ensuring the optimality of the consequent parameters by proposing an extended weighted recursive least squares (EWRLS) method.

(ii) Noting the importance and difficulties in selecting the thresholds for the EFS, a self-evolving fuzzy system (SEFS) has been proposed as a learning approach for the EFS. Unlike the fixed control parameters that are commonly used in EFSs learning methods, SEFS uses online training errors which can assist in measuring the quality of an identified model in presenting the dynamics of the data stream, in order to set a dynamic threshold automatically for rule generation. This self-tuning parameter, which controls the speed and coverage for fuzzy rule generation, helps SEFS to properly deal with the underfitting/overfitting problems. Furthermore, an $L^2$-distance-based geometric similarity measure has been proposed in the rule merging phase. Based on the SEFS, a fully automatic learning method for both thresholds that controls rule base expansion and shrinkage has been proposed, with the name of EFS-SLAT (EFS which has self-learning/adaptive thresholds). This self-learning strategy enables the thresholds to be completely learnt and updated, relying on the information contained in the data. This procedure has been carried out by causing the thresholds to restrict each other.

(iii) An functional fuzzy system model, which is referred to as functional fuzzy system (FFS), is the first fuzzy system approach to be developed for dealing with the functional regression problem. FFS enables both inputs and outputs of the model to be functionals (or functions). Constructed by a collection of the “If-Then” fuzzy rules, on the one hand, FFS has flexible structures, and can model the functional data without any model assumptions; On the other hand, the fuzzy inference also enables FFS to be an interpretable model. In order make FFS applicable, an offline learning approach has been proposed for FFS. This learning approach learns the antecedent part of FFS with the help of k-mean method, and the consequent part
by the least squares estimation formulas that have been inducted in this thesis.

(iv) Based on FFS, a functional regression online model named “Evolving Functional Fuzzy System (EFFS)” has been proposed. EFFS enables to deal with the online regression problems of the functional streaming data (and big functional data) in an transparency manner. EFFS acts as a one-pass approach, which has no requirement for storing historical data. In order to learn the parameters of EFFS, an online learning approach has been proposed for EFFS. The identification approach enables the EFFS evolve from an empty rule base, expand and shrink the rule base, and update the parameters dynamically. The recursive method for learning the antecedent and consequent parameters have been inducted with mathematical inductions.

In order to better explain the contributions that have made by this thesis, the following figure (see Fig.1.1) gives a brief summary. The contributions of this thesis are shown in the bullets inside the two light yellow rectangles.

![Summary of contributions](image)

Fig. 1.1: Summary of contributions.

1.5 Thesis Structure

The rest of this thesis is organized as follows.

Chapter 2 can be separated into two parts based on the regression approaches for two target data types: data streams and functional data. The first part will focus on
Chapter 1. Introduction

Summarizing the existing works of the data stream regression approaches: EFSs. This part includes: Introduction to basic information about fuzzy sets and systems, and a summary of EFSs and their learning approaches. Furthermore, a critical discussion on the limitations of the existing works is provided. The second part of this chapter gives a systematic review for the models of functional data regression based on the data type (scalar or function) of the input and output. In a similar way to the first part, a critical analysis of the literature on functional data regression is presented.

Chapter 3 is used to present the works that investigate the optimality of the EFSs learning approaches. Two optimum approaches, EMFSPO and LEOA, have been proposed in order to learn the EFSs. EMFSPO and LEOA are two methods inducted from minimizing a bunch of local error functions for Mamdani and Takagi-Sugeno (T-S) fuzzy systems respectively. Additionally, the merits and limitations of these proposed approaches are analyzed within this chapter. Both EMFSPO and LEOA have been verified and compared with existing methods on some selected benchmark examples.

Following on from Chapter 3, but differing in content, Chapter 4 concentrates on the predefined-thresholds automatic tuning problem of the EFSs learning methods. In order to propose approaches that can automatically learn the thresholds from the data without the predefined-thresholds, a two-stage work is presented. SEFS is the initial (first stage) work which enables the threshold that controls rule generation tuning automatically in an interval instead of using a fixed predefined parameter. The extended work of SEFS presented in Chapter 4 is EFS-SLAT. It has both thresholds which control the rule generation and reduction, learnt automatically and dynamically from the data. Comparisons to the state-of-the-art approaches have been made in several frequently-used benchmark examples.

Chapter 5 presents the proposed offline model, FFS, to solve functional data regression problem. Firstly, it presents the interpretable nonlinear functional regression model, FFS, which has both inputs and outputs being functionals. Secondly, the offline learning approach for FFS has been shown in this chapter. A detailed explanation and mathematical inductions has been provided. Comparisons to the state-of-the-art approaches have been presented. The applicability of FFS has been verified by the numerical examples.

Furthermore, in Chapter 6, an online functional streaming data regression model has been proposed. This model is EFFS, and can be regarded as a one-pass approach that has no requirement to store historical data. The corresponding online learning method of
EFFS has been demonstrated in this chapter. EFFS has been compared with the existing state-of-the-art functional regression methods in several benchmark datasets. Additionally, an online learning application for EFFS for learning the intra-daily volumes of two Exchange Traded Funds (ETFs) is also shown in this chapter.

Conclusions and future works are presented in Chapter 7.
Chapter 2

Background and Related Work

2.1 Chapter Introduction

In this chapter, the background and related works are summarized. Section 2.2 introduces EFSs related works, which includes the basic technique background knowledge (i.e., fuzzy sets and systems), architectures/frameworks of EFSs and the learning approaches of EFSs. Additionally, Section 2.2 also gives a critical analysis of the previous EFSs learning approaches. A detailed literature review of functional regression approaches based on the data type of the inputs and outputs is provided in Section 2.3. The research gaps and limitations are also discussed at the end of Section 2.3. Finally, this chapter is briefly summarized in Section 2.4.

2.2 Evolving Fuzzy Systems — Streaming Data Regression Approaches

EFSs are advanced online fuzzy system approaches for modelling data. Date back the history of EFS and fuzzy systems, the concepts of fuzzy systems were introduced by Zadeh [7] in 1960s. In a similar way to artificial neural networks, fuzzy systems are also universal approximators [9–16], which have the ability to approximate any nonlinear continuous functions in any accuracy. In a different way from artificial neural
networks, fuzzy systems use “If-Then” fuzzy rules, which make fuzzy systems more understandable and interpretable. With these characteristics, fuzzy systems are widely used in classification, decision making, pattern recognition, nonlinear and complex systems identification and prediction.

Early research on fuzzy systems usually assume that the systems have fixed structures, causing these systems to have a lack of flexibility for dealing with the situation that data comes as streams and changes over time. Therefore, in order to learn the data streams effectively and to forecast the data streams accurately, the research of fuzzy systems in recent years are focusing more on developing fuzzy systems which can update their structures using continuously learnt knowledge from the streaming data. This makes the development of EFS research.

This section first introduces the basic knowledge required for understanding EFSs, including fuzzy set, membership functions, fuzzy systems. Afterward, evolving fuzzy systems research is summarized.

### 2.2.1 Fuzzy Sets and Membership Functions

This section introduces the terms which are used for building a fuzzy system, including terms such as fuzzy sets, membership functions, $\alpha$-cut and fuzzy rules.

**Definition 2.1. Definition of fuzzy sets** [17] A fuzzy set $\Gamma$ in a given space $X$ ($X \neq \emptyset$) is a set of pairs:

$$\Gamma = \{ (x, \mu_\Gamma(x)) | x \in X \}, \quad (2.1)$$

where $\mu_\Gamma(x) : X \rightarrow [0, 1]$ is the map associate with $\Gamma$ from $X$ to $[0, 1]$; This function $\mu_\Gamma(x)$ is the membership function of a fuzzy set $\Gamma$ used to indicate the membership degree of each element $x \in X$ to the fuzzy set $\Gamma$.

In [17], it was suggested that the membership function $\mu_\Gamma(x)$ can be explained from the following three situations:

1. $\mu_\Gamma(x) = 1$: the full membership of $x$ to the fuzzy set $\Gamma$;
2. $\mu_\Gamma(x) = 0$: the lack of membership of $x$ to the fuzzy set $\Gamma$;
3. $0 < \mu_\Gamma(x) < 1$: a partial membership of $x$ to the fuzzy set $\Gamma$. 
The membership functions have varies forms, the most simple one being the Singleton function. The most widely used functions are the Gaussian membership function and the triangle membership function. Occasionally, trapezoidal membership functions would also be used. As a smooth function, the Gaussian membership function has stronger universality than the triangle and trapezoidal membership functions. These membership functions are shown as follows, and more detailed information can be found in [17, 18].

- **Singleton** membership function $\mu(x)$ can be presented as:

$$
\mu(x) = \begin{cases} 
1, & \text{if } x = c, \\
0, & \text{if } x \neq c,
\end{cases}
$$

in which fuzzy set $\Gamma$ is the set which only contains one point $c$.

- **Gaussian** membership function $\mu(x)$ has the form:

$$
\mu(x) = \exp\left\{-\frac{(x-c)^2}{2\sigma^2}\right\},
$$

where $c$ is the cluster centre (the centre of the Gaussian curve), $\sigma$ stands for the radius (the width of the Gaussian curve).

- **Triangle** membership function has the form:

$$
\mu(x) = \begin{cases} 
0, & \text{if } x \leq a \text{ or } x \geq c \\
\frac{x-a}{b-a}, & a \leq x \leq b, \\
\frac{c-x}{c-b}, & b \leq x \leq c.
\end{cases}
$$

- **Trapezoidal** membership function has the form:

$$
\mu(x) = \begin{cases} 
\frac{x-a}{b-a}, & a \leq x < b \\
1, & b \leq x \leq c, \\
\frac{x-d}{c-d}, & c < x \leq d, \\
0, & \text{others}.
\end{cases}
$$

Membership function can help to compute the membership degree regarding to a single fuzzy set. However, the membership value of intersection can be computed using
“T-norm” [19]. Some examples taken from [20] for T-norm are shown as follows:

\[
\text{Product: } T(x_1, x_2, \ldots, x_n) = x_1 x_2 \cdots x_n, \quad (2.6)
\]

\[
\text{Minimum: } T(x_1, x_2, \ldots, x_n) = \min(x_1, x_2, \ldots, x_n), \quad (2.7)
\]

Łukasiewicz: \[ T(x_1, x_2, \ldots, x_n) = \max\left(\sum_{i=1}^{n} x_i - (n - 1), 0\right). \] (2.8)

The ‘product’ is the most widely used one.

2.2.2 Fuzzy Systems

2.2.2.1 Basic Techniques of Fuzzy Systems

Depending on the definitions of fuzzy sets and membership function, the definitions of fuzzy rules, fuzzy rule base and fuzzy systems are presented. Note that fuzzy rules, and rule base are used to construct fuzzy systems.

The fuzzy rule, also known as “If-Then” fuzzy rule, is a reasoning process constructed by antecedent part \( \Gamma \) and consequent part \( B \). The general form of a fuzzy rule is (2.9):

\[
\text{If } x \text{ is } \Gamma, \text{ Then } y \text{ is } B. \quad (2.9)
\]

Note that a fuzzy rule can have multiple antecedents and consequences, as well as multiple forms. For instance, the multiple antecedent part can be either conjunctive, which uses “AND” to connect each input, or disjunctive, applying “OR” to connect the inputs; Besides, consequent parts may contain conditional statements using “ELSE”, or may be nested with another rule. However, among those fuzzy rules, the most frequently used rules would be single-input-single-output (SISO) shown in (2.9), and multi-input-single-output (MISO) and multi-input-multi-output (MIMO) rules, as presented in (2.10) and (2.11), respectively:

\[
\text{If } x_1 \text{ is } \Gamma_1 \text{ and } x_2 \text{ is } \Gamma_2 \text{ and } \ldots \text{ and } x_n \text{ is } \Gamma_N, \text{ Then } y \text{ is } B, \quad (2.10)
\]

\[
\text{If } x_1 \text{ is } \Gamma_1 \text{ and } x_2 \text{ is } \Gamma_2 \text{ and } \ldots \text{ and } x_n \text{ is } \Gamma_N, \text{ Then } y_1 \text{ is } B_1 \text{ and } \ldots \text{ and } y_m \text{ is } B_m. \quad (2.11)
\]
More detailed introduction about other forms of fuzzy rules can be found in [21].

A fuzzy rule base consists of a collection of “If-Then” fuzzy rules. Usually, a fuzzy system is constructed with input, output and fuzzy rule base. It works with the help of fuzzification, inference system, and defuzzification. The general framework of a fuzzy system can be presented in Fig. 2.1.

![Fig. 2.1: The framework of a fuzzy system.](image)

The following bullet-pointed list explains the role of each of the blocks in Fig. 2.1.

- **Fuzzification**
  The major aim of this block is to construct the fuzzy sets. It can help to convert the quantities to be measured into the fuzzy sets.

- **Inference System**
  As stated by [18], this block has two tasks: (1) To determine the firing strength of each rule; and (2) To construct an implied fuzzy set for each fuzzy rule in accordance with its firing strength, in order to know how much each rule should be weighted for the final decision.

- **Rule base**
  The rule base provides a set of relations, which are conveyed by the “If-Then” fuzzy rules, for reasoning.

- **Defuzzification**
  There exist many different methods for defuzzification, e.g., centre of gravity (COG), centre of area (COA), mean of maxima (MOM), expected value and expected interval (EV and EVI), maximum criterion, etc. (see [22, 23]), in order to...
accomplish the task of this process. Take COG, which is the most commonly used approach, as an example. The COG can be defined as:

\[ y = \frac{\int y \mu_{B'}(y) dy}{\int \mu_B(y) dy}, \]  

(2.12)

where the \( B' \) is the fuzzy set obtained by the fuzzy inference. The discrete form of (2.12) can be written as (2.13):

\[ y = \frac{\sum y_i \mu_{B'}(y_i)}{\sum \mu_{B'}(y_i)}. \]  

(2.13)

### 2.2.2.2 Examples of Fuzzy Systems

Based on the basic knowledge and structure of a fuzzy system, several most widely used architectures (or frameworks) of fuzzy systems will be presented in this section. These systems can be SISO, MISO or MIMO. Due to the MISO type being the one which is concentrated on in this thesis, the MISO type is applied to each architecture in the following examples.

#### 2.2.2.2.1 Rule-Based Fuzzy Systems

Taking T-S fuzzy system, Mamdani fuzzy system, generalized fuzzy system, and interval type-2 fuzzy system as examples to show the basic architecture of the rule-based fuzzy systems. Because the existing works of them may have different membership functions, fuzzification and defuzzification approaches, some examples are selected from the existing work in order to demonstrate how these systems work.

##### 2.2.2.2.1.A T-S Fuzzy System

T-S system\(^{1}\) is known as precise fuzzy modeling approaches aiming to achieve high output accuracy \([25]\). From the existing literatures, e.g., \([26–41]\), the fuzzy systems, which have T-S framework, have the \( i \)-th fuzzy rule \( R_i \) shown as:

\[ R_i : \text{If } \mathbf{x}_1 \text{ is } \Gamma_{i,1} \text{ and } \mathbf{x}_2 \text{ is } \Gamma_{i,2} \text{ and } \ldots \text{ and } \mathbf{x}_n \text{ is } \Gamma_{i,n}, \text{ then } y_i = \psi_{i,0} + \sum_{j=1}^{n} \psi_{i,j} x_j, \]  

(2.14)

---

\(^{1}\)It has been stated by \([24]\) that the T-S system is most frequently applied fuzzy system framework in EFSs studies.
CHAPTER 2. BACKGROUND AND RELATED WORK

where \( i = 1, 2, \ldots, K \), \( K \) is the number of fuzzy rules, \( \mathbf{x} = (x_1, x_2, \ldots, x_n) \), in which \( x_j \) is the \( j \)-th input, \( \mathbf{x} \in \Omega = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n] \subset \mathbb{R}^n \), \( y_i \) is the output of rule \( R_i \), coefficients \( \psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,n} \) form the vector of consequent parameters \( \psi_i = (\psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,n}) \), and \( n \) is the number of the input variables.

The membership function of \( \Gamma_{i,j} \) is \( \mu_{i,j}(x_j) \), which is a Gaussian membership function \([42–44][28–30, 32–37, 39–41]\) with form:

\[
\mu_{i,j}(x_j) = \exp\left\{-\frac{(x_j - c_{i,j})^2}{2\sigma_{i,j}^2}\right\}, \quad (2.15)
\]

in which \( c_{i,j} \) and \( \sigma_{i,j} \) are the cluster centre and radius, respectively. There are also other functions being used, for example triangular \([26]\), \([31]\) uses Cauchy type membership functions, \([38]\) uses a distance based membership functions by solving an optimization problem. Furthermore, for rule \( R_i \), the firing strength \( \gamma_i(x) \) and the normalized firing strength \( \theta_i(x) \) are presented by (2.16) and (2.17), respectively:

\[
\gamma_i(x) = \prod_{j=1}^{n} \mu_{i,j}(x_j), \quad (2.16)
\]

\[
\theta_i(x) = \frac{\gamma_i(x)}{\sum_{j=1}^{K} \gamma_j(x)}, \quad (2.17)
\]

and the final output of the system \( \hat{y} \) can be computed by (2.18):

\[
\hat{y} = \sum_{i=1}^{K} \theta_i(x)y_i. \quad (2.18)
\]

2.2.2.2.1.B Mamdani Fuzzy System  

The Mamdani fuzzy system is recognized to be more interpretable than the T-S types \([45]\). Obviously, Mamdani fuzzy system is constructed with a group of Mamdani fuzzy rules:

\[ R_i : \text{If } x_1 \text{ is } \Gamma_{i,1} \text{ and } x_2 \text{ is } \Gamma_{i,2} \text{ and } \ldots \text{ and } x_n \text{ is } \Gamma_{i,n}, \text{ then } y_i \text{ is } \tilde{\Psi}^i, \quad (2.19) \]

where \( i = 1, 2, \ldots, K \), \( K \) is the number of fuzzy rules; The input is \( \mathbf{x} = (x_1, x_2, \ldots, x_n) \); \( y_i \) is the output; \( \tilde{\Psi}^i \) and \( \Gamma_{i,j}, j = 1, 2, \ldots, n \) are fuzzy sets; The centres of fuzzy sets \( \tilde{\Psi}^i \) are \( \psi_i (i = 1, 2, \ldots, n) \), respectively; \( n \) is the dimension of the input. The membership function \( \mu_{i,j}(x_j) \) of the Mamdani type can also be Gaussian \([46–50]\), triangular \([48, 51]\),
To compute the final output, taking SOFMLS [47] as an example, the firing strength \( \gamma_i(\mathbf{x}) \) of each rule \( R_i \) can be computed by (2.16), and the final output can be computed by:

\[
\hat{y} = \sum_{i=1}^{K} \theta_i(\mathbf{x}) \psi_i,
\]

(2.20)

in which \( \psi_i \) are the centres of fuzzy sets \( \tilde{\Psi}^i \) \( (i = 1, 2, \ldots, n) \), and \( \theta_i(\mathbf{x}) \) is the same as (2.17).

It can be seen that if the same number of rules are used, a Mamdani system has fewer number of consequent parameters to be estimated compared with a T-S system.

2.2.2.2.1.C Generalized Fuzzy System

Conventional fuzzy systems treat the input variables independently, while a generalized fuzzy system considers the relationship between them from a probability point of view. Based on the probability theory, generalized fuzzy system can make up the information which is lost when interactions happen between input variables [52–54]. An example of the generalized fuzzy system is a system built by a generalized form of the T-S rule base with multivariate Gaussian membership function proposed in [54]. With the characteristics that the multivariate Gaussian membership function models the relationship between input variables, generalized fuzzy system becomes increasingly one of the main architectures for fuzzy systems. The fuzzy system built by many recent research studies, e.g., [8, 55–60], are based on the multivariate Gaussian membership functions. To better explain the generalized fuzzy system, the system shown in [54] is applied as an example. It is formed by a set of fuzzy rules. The i-th rule \( R_i \) of the generalized fuzzy system can be presented as follows:

\[
R_i : \text{If } \mathbf{x} \text{ is } \Gamma_i, \text{then } \mathbf{y}_i = \psi_{i,0} + \sum_{j=1}^{n} \psi_{i,j} \mathbf{x}_j,
\]

(2.21)

where \( i = 1, 2, \ldots, K \), and the membership function \( \mu_i(\mathbf{x}) \) is:

\[
\mu_i(\mathbf{x}) = \exp\left\{-\frac{1}{2}(\mathbf{x} - \mathbf{c}_i)\Sigma^{-1}_i(\mathbf{x} - \mathbf{c}_i)^T\right\},
\]

(2.22)

in which \( \mathbf{c}_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,n}) \) is the centre, and the covariance matrix \( \Sigma_i \) is a symmetric, positive-definite matrix. In this section, the firing strength \( \gamma(\mathbf{x}) \) of rule \( R_i \) is eventually the membership degree \( \mu_i \) (i.e. \( \gamma(\mathbf{x}) = \mu_i(\mathbf{x}) \)). The final output of the system
can be calculated by (2.17) and (2.18).

### 2.2.2.2.1.D Interval Type-2 Fuzzy System

Interval type-2 fuzzy system \([61–65]\) is another frequently used fuzzy system. SEIT2FNN \([64]\) is selected as an example to show the general structure of an interval type-2 fuzzy system. The fuzzy rule \(R_i\) of the system can be demonstrated by (2.23):

\[
R_i: \text{If } x_1 \text{ is } \tilde{\Gamma}_{i,1} \text{ and } x_2 \text{ is } \tilde{\Gamma}_{i,2} \text{ and } \ldots \text{ and } x_n \text{ is } \tilde{\Gamma}_{i,n}, \text{ then } y_i \text{ is } \tilde{\psi}_{i,0} + \sum_{j=1}^{n} \tilde{\psi}_{i,j}x_j. \tag{2.23}
\]

where \(\tilde{\Gamma}_{i,j} (j = 1, 2, \ldots, n)\) are interval type-2 fuzzy sets, \(\tilde{\psi}_{i,j} = [p_{i,j} - s_{i,j}, p_{i,j} + s_{i,j}]\) \((j = 0, 1, 2, \ldots, n)\) are interval sets. The Gaussian membership function \(\mu_{i,j}(x)\) should be presented by:

\[
\mu_{i,j} = \exp\left\{-\frac{(x - c_{i,j})^2}{2\sigma_{i,j}^2}\right\}, \tag{2.24}
\]

in which \(c_{i,j} \in [\underline{c}_{i,j}, \overline{c}_{i,j}]\). Therefore, \(\mu_{i,j}\) is located in the \([\underline{\mu}_{i,j}, \overline{\mu}_{i,j}]\), and the firing strength \(\gamma_i\) is within \([\underline{\gamma}_i, \overline{\gamma}_i]\) with \(\underline{\gamma}_i = \prod_{j=1}^{n} \underline{\mu}_{i,j}\) and \(\overline{\gamma}_i = \prod_{j=1}^{n} \overline{\mu}_{i,j}\). Consequents are computed by \([y'_i, y''_i] = \tilde{\psi}_{i,j} = [p_{i,0} - s_{i,0}, p_{i,0} + s_{i,0}] + \sum_{j=1}^{n} [p_{i,j} - s_{i,j}, p_{i,j} + s_{i,j}]\). The final output is computed through using Karnik-Mendel iterative procedure \([66]\) to reorder the \(y'_i, y''_i, \underline{\gamma}_i, \overline{\gamma}_i\) with an ascending order. The output should be \(y = \frac{y'_i + y''_i}{2}\), where

\[
y'_i = \frac{\sum_{i=1}^{L} \overline{\gamma}_i y'_i + \sum_{i=L+1}^{K} \overline{\gamma}_i y'_i}{\sum_{i=1}^{L} \overline{\gamma}_i + \sum_{i=L+1}^{K} \overline{\gamma}_i}, \tag{2.25}
\]

\[
y''_i = \frac{\sum_{i=1}^{R} \overline{\gamma}_i y''_i + \sum_{i=R+1}^{K} \overline{\gamma}_i y''_i}{\sum_{i=1}^{R} \overline{\gamma}_i + \sum_{i=R+1}^{K} \overline{\gamma}_i}. \tag{2.26}
\]

### 2.2.2.2 Neuro-Fuzzy Networks

Unlike the rule-based systems presented in Section 2.2.2.2.1, another frequently used architecture, Neuro-Fuzzy Networks (NFNs), is presented in this section. From the existing work \([67]\), it can be learned that NFNs can be equivalent to fuzzy rule-based
systems. NFNs have neural network structures, whilst having fuzzy inference imbedded. The underlying fuzzy rules can be Mamdani \([46–48]\), T-S \([49, 68–82]\) or interval type-2 \([61–65]\). There are different types of structures for NFNs, due to the differences between fuzzy rules. As the basic architectures of most of them are similar and the biggest differences between them are the learning approaches, some typical examples are selected to show a rough picture of the structures of these NFNs.

NFNs that are based on Mamdani and T-S rules are introduced together because they have very similar structures. One of the most famous and classic NFNs, known as ANFIS \([67]\), is taken as an example. ANFIS is the fuzzy rule based system with conventional T-S fuzzy rules shown in Section 2.2.2.1.A, which can be equivalent to a five-layer NFN presented by Fig. 2.2. It can be seen from \([47]\) and \([46]\), the NFN with Mamdani rules can also be presented in the same way as Fig. 2.2, while the architecture of \([48]\) looks a little bit different with a temporal connection layer paralleled with the layers 1, 2 and 3 in Fig. 2.2. This temporal connection layer has a short memory of the previous behaviour. In addition, SEIT2FNN \([64]\) shown in Section 2.2.2.2.1.D is an example, in order to demonstrate the equivalent NFN structure of an interval type-2 system. The corresponding NFN framework for SEIT2FNN can be depicted in Fig. 2.3.

Note that only the most frequently used NFNs have been introduced in this thesis.
There also exist some rarely studied NFN structures such as the neo-fuzzy neuron network [83], and the modular neural network (MNN) [84]. The first one is modeled by cutting each input variable into different partitions; The second one, meanwhile, is a hierarchical structure which enables us to model the outputs from a radial basis function (RBF) network by additional subnetworks.

### 2.2.3 Evolving Fuzzy Systems

In this section, an overview of the research work concerning EFSs in the field of regression is given. The architectures (models) of EFSs are different types of fuzzy systems, such as the fuzzy systems shown in Section 2.2.2.2. The major difference between an EFS and a fuzzy system is that EFS has the fuzzy rule numbers (neurons) and the parameters changing over time. Therefore, the essential problems that need to be solved are: 1) How to identify the rule numbers online? 2) How to estimate the parameters dynamically? Proposing effective methodologies to deal with these aforementioned problems is the common and crucial problem studied by the EFS researchers.

Early works about EFSs are usually offline learning approaches which are required to save historical data in order to train the system. For example, [85] proposed an NFN learning approach that can learn its structure from the training dataset by K-nearest neighbour heuristic, and can recursively estimate the parameters. This model has a fixed
structure, and it should be learnt using all the training data at the same time. A similar method can be found in [86], in which the ANFIS has been proposed with only the parameters being adaptive online. Unfortunately, the fixed structure models are inconvenient for using in learning the complex and the nonstationary data in an online manner. As a result, some improved approaches have been proposed to enable the system to keep up-to-date. Furthermore, due to the fact that the more data fed into the system, the more complete picture of the data can be reflected. The fixed structure models mean that it is hard to depict the data dynamics, which therefore leads to terrible predictions. The NFN proposed in [87] allows both the parameters and structure to be evolved online, while an initialization phase is needed to help the system to get started. Based on these basic works, incremental approaches, such as dynamic evolving neural-fuzzy inference system (DENFIS) [26], evolving Takagi-Sugeno approach (eTS) [27], and flexible fuzzy inference systems (FLEXFIS) [28] have been proposed to allow the system to evolve from an empty rule base. These incremental approaches can only generate new rules on its learning process, but have no mechanism for simplifying the rule base [88]. Besides, as the EFSs are changing by nature, some of the rules may possibly to grow similar, and this may cause rule conflict. To relieve the computational burden, and to prevent overfitting and rule conflict, a great deal of EFSs approaches have been proposed, such as the most recent works, [8] and [89].

However, the evolving structure learning approaches are far from identifying the EFSs in an online manner. The learning approaches for the parameters have a great influence on the online learning ability of the EFSs. Many existing approaches whose structures can be evolved online may also need to store the training data. This situation is usually caused by the optimization methods applied for estimating the parameters. For example, the stochastic gradient descent (SGD) [90] can realize the online training process. However, better performance can be achieved by allowing the system to repeat training on the training set hundreds of times. In contrast, the recursive least squares (RLS) used in [27], usually perform as one-pass approaches.

In the following content of this section, in Section 2.2.3.1, approaches for rule generation and simplification, which are proposed in the existing literatures, are summarized. Besides, several generally used parameters learning approaches are also presented in Section 2.2.3.1.
2.2.3.1 Approaches to Learn Evolving Fuzzy Systems

As an EFS can automatically vary and adjust its structure and parameters according to the data, in this section, a summary of the frequently used approaches of an EFS for evolving its structure is shown and updating its parameters. Every time when a new data comes, an EFS learning approach learns the EFS relaying on the following theory: If the new knowledge exceeds the explanation ability of the EFS, then the fuzzy rule base should be expanded to cover the new knowledge for avoiding underfitting; With the increasing knowledge that is gained from the data and the growing number of fuzzy rules, there are likely to exist conflict fuzzy rules or redundant rules. Thus, it is reasonable to simplify the rule base in order to increase the accuracy while avoiding overfitting. The evolving of the EFSs is very likely to give rise to a question: How does an EFS evolve its structure and tune its parameters?

This question has been answered in many research works from two major modules: (1) How to evolve the structure? (2) How to update the parameters? To be more specific, structure evolving usually includes rule generation, merging, and pruning; whereas the parameters learning requires us to learn both antecedent and consequent parameters. There is no requirement for all the EFSs learning approaches to contain all these

---

2 The structure learning contains learning the rule number in a rule based structure or neuro number in an NFN structure.
3 Parameters updating includes learning the parameters in the fuzzy rules.
modules, but each EFS method should contain at least one module. The general framework for an EFS (or an NFN) learning approach can be demonstrated as Fig. 2.4. To explain an EFS learning approach in another way, setting proper learning strategies for each module in this figure is the major task of an EFS learning approach. A great deal of research works have been undertaken to identify EFSs. The evolving approaches for each module in the existing state-of-the-art methods will be discussed in the remaining part of this section.

2.2.3.1.1 Structures Evolving of EFSs

The structure evolving of each EFS is aroused by the changing number of fuzzy rules in essence. This procedure includes fuzzy rule generation and rule simplification. Some major techniques proposed in the existing works are demonstrated as follows:

2.2.3.1.1.A Fuzzy Rule Generation

Fuzzy rule generation is also known as “fuzzy rule adding”. Since there is usually no prior knowledge about how many and what fuzzy rules are needed in order to depict the input space, it is a crucial task to learn the fuzzy rules including the cluster centres and radii as well as fuzzy rule number online. In order to make the system able to make a decision for generating a new fuzzy rule, the criterion which triggers this process should be the key point. In the early work of EFSs, a distance-based criterion has been used. The basic idea of this criterion is to judge the distance between the new input and the existing clusters. The criterion can be presented as:

**Criterion 2.1.** If $\forall i, d(x, \tilde{C}_i) > \varepsilon^{(a)}$, then a new rule should be generated. The $\tilde{C}_i$ is the existing clusters decided by their centres and radii, $\varepsilon^{(a)}$ is a threshold that controls the rule generation speed.

The above criterion generates new fuzzy rule based on the following theory: large distances between new data $x$ and any of the existing cluster centres $\tilde{C}_i$ demonstrate that $x$ is beyond the range that the current system can explain, therefore, a new rule should be generated to help the system to cover $x$. To do so, new rule is usually generated with center $x$. Radius and Consequent parameters of the new rule are initialized by the mean of the radii and consequent parameters of the existing rules, respectively. More approaches for initializing the parameters of the new generated rules are presented in Section 2.2.3.1.2.
Existing studies which apply distance-based criterion rule generation approaches are, for example, the dynamic evolving neural fuzzy inference system (DENFIS) as proposed in [26]. This adds new fuzzy rules based on the Euclidean distances between the new input data and the existing cluster centres. Furthermore, flexible fuzzy inference systems (FLEXFIS) [28] and dynamically evolving clustering (DEC) [35] judged whether a new input sample is in the existing clusters by comparing distances between this input and the cluster centres with the corresponding radii. Considering EFSs developed based on generalized fuzzy rules, the Mahalanobis distance has been used to control the fuzzy rule growth (see, e.g., [58] and [38]).

In a similar way to the distance-based methods, there exists another effective criterion built according to the firing strength (or activation degree). As the firing strength takes the distance between the input and existing cluster centres into consideration by nature, the essence of the firing strength is the same as the distance-based criteria. However, the firing strength is more intuitive in assessing whether a data point is close to a cluster centre. The general form of this kind of criterion can be presented as:

**Criterion 2.2.** If \( \max_{i=1,...,K} \gamma_i(x) < \varepsilon^{(a)} \), then a new rule should be generated. The \( \gamma_i \) is the firing strength shown in (2.16).

This criterion helps to add new rules based on the following theory: A firing strength \( \gamma_i(x) \) demonstrates the degree that the data \( x \) located in a certain cluster. The lower the firing strength, the lower degree for \( x \) close to the existing clusters. Therefore, the criterion above illustrates that a new rule should be added when \( x \) is far away from all the existing clusters.

EFSs, such as the self-organizing fuzzy neural network (SOFNN) [68], self-organizing fuzzy modified least-square network (SOFMLS) [47] whose improved approaches can be found from [8] and [42], evolving neural-fuzzy semantic memory model (eFSM) [91] and evolving neo-fuzzy neural network approach (eNFN) [83] have implemented the firing strength to make decisions on fuzzy rule generation. Furthermore, recent research studies that have used firing strength, for example, are the generic self-evolving Takagi-Sugeno-Kang fuzzy framework (GSETSK) [69], and the EFSs identification method proposed in [92].

Besides, the datum significance (DS) criterion has also been used as a rule generation criterion. As a generalized version of the significant criterion [77], [43,44,93] and the influence [94], the DS criterion measures whether the new input can contribute more
to the prediction results than existing clusters. For example, the parsimonious network based on the fuzzy inference system (PANFIS) [55] and the Generic Evolving Neuro-Fuzzy Inference System (GENEFIS) [56] have used DS criterion in rule generation. In addition to these approaches for rule generation, there also exist many other criteria, for instance, rule potential [27, 29, 31] and arousal index, [30, 40, 54, 82]. More specifically, these criteria can be presented mathematically in the following bullet:

- **Significant Criterion:** This criterion is based on both the distance between the input and the existing clusters, and the possible statistical contribution of a new generated rule. If (2.27) holds, then a new rule should be generated.

\[
\begin{align*}
\|x_t - c^*\| > e^{(a)}, \\
E_{inf} > \text{thr}_{E},
\end{align*}
\] (2.27)

where \(c^*\) is the \(x_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,n})\)’s nearest cluster centre, \(E_{inf}\) is known as the significance in [93] and Influence in [94], \(e^{(a)}\) and \(\text{thr}_{E}\) are thresholds that control rule generation. In [94], \(E_{inf}\) is computed by the following formula (2.28):

\[
E_{inf} = \|e(t)\| \left(\frac{1.8\kappa\|x_t - c^*\|}{\sum_{i=1}^{K+1} (1.8\sigma_i)}\right)^n,
\] (2.28)

where \(\|e(t)\| = \|y_t - \hat{y}_t\|\) is the estimation error of output \(y_t\); \(\kappa\) is a predefined overlapping factor, \(K + 1\) is the total number of fuzzy rules when there is a new rule \(R_{K+1}\) just added. Whereas \(E_{inf}\) in [93] is slightly different from (2.28), and it can be computed by (2.29):

\[
E_{inf} = \sum_{k=t-M+1}^{t} \frac{\|e(t)\|}{M} \frac{\gamma_{K+1}(x_k)}{\sum_{i=1}^{K+1} \gamma_i(x_k)},
\] (2.29)

in which \(M\) is a predefined number that used to set the number of data that should be considered to evaluated the influence of the new added rule; \(\gamma(x)\) are the firing strengths. Criterions in (2.27) consider both whether data \(x\) is far away from the existing clusters and whether the data has significant influence to the system.
• **Datum Significance (DS) Criterion**: If $D_t \geq \epsilon(a)$, then a new rule should be generated. The $D_t$ is the DS computed by:

$$D_t = \|e(t)\|\left\{\frac{\text{det}(\Sigma_{K+1})}{\sum_{i=1}^{K+1} \text{det}(\Sigma_i)}\right\}^n,$$  \hspace{1cm} (2.30)

where $e(t) = y_t - \hat{y}_t$, $K$ is the rule number, $\Sigma_i$ are the covariance matrix of a multivariate Gaussian membership function, and $n$ is the dimension of the input space. Similar to the significant criterion, DS criterion recruit new rules relying on the statistical contribution.

• **Rule Potential**: The potential measures the spatial proximity of a data point to all the other data. The bigger difference the data point has, the higher potential it has to become a new focal point. The criterion can be presented as: If $\forall i \in [1, 2, \ldots, K]$ $P_t(x_i) > P_t(c_i)$ holds, then a new fuzzy rule should be generated. The potential $P_t$ can be calculated by:

$$P_t(x_t) = \frac{1}{1 + \frac{1}{t+1} \sum_{i=1}^{t-1} \sum_{j=n+1}^{n+1} (d_{i,t}^j)^2},$$  \hspace{1cm} (2.31)

where $x_t$ is the new coming input data, $c_i$ are the cluster centres, $d_{i,t}^j = x_{t,j} - x_{l,j}$, $(j = 1, 2, \ldots, n)$, and $d_{i,t}^j = y_t - y_j$, $(j = n + 1)$.

• **Arousal Index**: If $\forall i = 1, 2, \ldots, K$ $a^i_t > \epsilon(a)$, then a new rule should be generated. The arousal index $a^i_t$ is computed recursively by:

$$a^i_t = a^i_{t-1} + \beta(1 - \rho^i_t - a^i_{t-1}),$$  \hspace{1cm} (2.32)

$$\rho^i_t = 1 - \frac{\|x_t - c_i\|}{n},$$  \hspace{1cm} (2.33)

where $\rho^i_t$ is the compatibility index that indicates the average distance between each component of vectors $x_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,n})$ and $c_i$; $n$ is the dimension of the input space. The arousal index detects the compatibility of the current state of the system to the observations, and generates new rules to fit the new environment, once a big gap has been observed.
2.2.3.1.1.B Fuzzy Rule Simplification

The rule simplification process helps an EFS to lower down the computational burden, keep a simple system, in order to make the system more understandable, avoid rule conflict, remove redundant rules, and avoid overfitting. There are two main ways of helping EFS to eliminate redundant rules: one is fuzzy rule pruning and the other is fuzzy rule merging. Rule pruning and merging serve different aims, but both of these two procedures can decrease the rule numbers.

The rule pruning module removes those redundant rules in order to assist the system to simplify the rule base. Redundant rules stand for those rules that are unimportant to the system. Removing these rules enables the system to use a simpler system to achieve similar accuracy without worrying about overfitting. Therefore, the method to judge a redundant rule is a crucial task for rule pruning. The frequently used criteria to judge redundant rules are, for example: Population/density [47, 95], which regards those clusters with a small number of data points as negligible clusters; Rule influence [55, 74, 94] and rule importance [68, 73] recognize negligible rules by computing the influence of the rules to the systems; Utility [29, 30, 37, 38, 96] and age [31, 32, 83] judge those seldom activated rules, or rules which have not been used for a long time, as unimportant rules. To be more specific, the mathematical form and the theory of these frequently used criteria are listed below:

- **Population/Density**: The population/density helps to remove those clusters that contain a small number of data points, whilst keeping those clusters with the majority of the data points inside. Assume that \(d_i\) is the population/density that counts the number of data points located within the corresponding cluster of rule \(R_i\) (i.e. \(d_i = d_i + 1\) when \(x\) belongs to this cluster). The rule pruning criterion can be presented as “If \(d_i < \epsilon^{(p)}\), then the rule \(R_i\) should be discarded.” In this criterion, \(\epsilon^{(p)}\) is the threshold and \(i^* = \arg\min_{i=1,...,K} d_i\).

- **Rule Influence**: The rule influence measure is defined as being the statistical contribution of a certain rule to the final output. A rule with very little contribution is naturally to be regarded as a redundant rule. The criterion can be presented as “If \(\exists i\) s.t. the influence \(E_{inf}(i) < \epsilon^{(p)}\), then remove the rule \(R_i\)” \(E_{inf}(i)\) is

\[
E_{inf}(i) = \frac{||\psi_i||}{\sum_{i=1}^{K} (1.8\sigma_i)^n}, \quad (2.34)
\]

where \(\psi_i\) is the consequent parameter, \(\sigma_i\) is the radius. Formula (2.34) shown
in [94] is based on the conventional rule base. The criteria in [55] and [74] are the extended versions.

- **Rule Importance**: This criterion judges the importance of the rules by computing how much the root mean square error (RMSE) would be changed by removing the rules. The smaller the change of RMSE, the higher redundant degree the rule has. Compute the RMSE when removing the rule that has the highest degree of redundancy; and then, compare this RMSE to the threshold. The rule would be permanently removed if the aforementioned RMSE was smaller than the threshold.

- **Utility**: This criterion measures the usefulness of a fuzzy rule, and removes those which are seldom activated rules. Utility stands for the average firing strength (activation degree) of a rule since it was added. Low utility demonstrates the rule is unimportant with low contribution, then, should be removed. Assume the utility of rule $R_i$ is $U_i$. If $U_i < \epsilon^{(p)}$, then eliminate rule $R_i$. $U_i$ can be computed by (2.35):

$$U_i = \frac{\sum_{t=1}^{T} \theta_i(x_t)}{(T - T_i)}, \quad (2.35)$$

where $t = 1, 2, \ldots, T$ are the time stamps, $T_i$ is the time stamp when rule $R_i$ is generated, and $\theta_i(x_t)$ is the normalized firing strength shown in (2.17).

- **Age**: “Age” is defined slightly differently in the existing literatures; while, these “age” measurements define the redundant rules according to how long the rules have been used. Those rules that have been generated for a long time are defined as “old” rules and treated to be out-of-date; thus, they should be removed.

Apart from pruning redundant rules, another desirable approach to simplifying the rule base is known as fuzzy rule merging. However, rule merging serves for a dual goal of avoiding overfitting and rule conflict at the same time. In essence, rule merging is different from rule pruning and acts on those similar fuzzy rules. Rule merging is needed because of the adaptation mechanism that leads to some fuzzy rules growing to be similar, along with the information gained. These two (or several) similar clusters would be merged to one cluster, in order to not only decrease the computational burden, but also to avoid the rule conflict. As a result, the question of how to judge the similarity of two fuzzy rules becomes a crucial problem. This is discussed by research studies addressing the merging process. There are two major branches of rule similarity
measure: Set-theoretic similarity measures and Geometric similarity measures \[97\]. Set-theoretic measures measure the proportion of the intersection of two clusters compared with the union of these two clusters, while geometric measures compare the distance between the membership functions of two clusters. Set-theoretic measures are valid measures which can determine the similarity between two overlapping clusters. Examples of previous research studies that have used set-theoretic similarity measures are GSETSK \[69\], eFSM \[91\] and eT2FIS \[61\]. Unlike set-theoretic measures, geometric measures are distance-based alternatives. Geometric similarity measures are usually fast and comparatively easy to calculate. Furthermore, they have also been widely used in many existing EFS identification algorithms (see, e.g., \[8, 30, 54–56, 58\] and \[89\]). In the following bullet-pointed list, more specific forms of both set-theoretic and geometric measures used in the existing literature are given as examples.

- **Set-Theoretic**: The common aim for the set-theoretic measures is to measure the overlapping degree between the fuzzy sets. Generally, the set-theoretic measure is based on the intersection and the union of the fuzzy sets, which should be calculated depending on the shape of the membership functions. Three forms of set-theoretic similarity measure are presented as examples.

  - The intersection between two fuzzy sets has been used as the similarity measure e.g., \[98\]. Assume \(S(R_1, R_2)\) and \(S(Γ_1, Γ_2)\) are the similarity between rules and the corresponding fuzzy sets, respectively. Thus, \(S(R_1, R_2)\) can be computed by:

    \[
    S(R_1, R_2) = S(Γ_1, Γ_2) = |Γ_1 \land Γ_2|. \quad (2.36)
    \]

  - Alternatively, the similarity measure can be defined as a measure of the degree that of a fuzzy set \(Γ_1\) to be a subset of \(Γ_2\) \[61, 69, 91\]. In this sense, the similarity measure \(S(R_1, R_2)\) can be presented as:

    \[
    S(R_1, R_2) = S(Γ_1, Γ_2) = \frac{|Γ_1 \land Γ_2|}{|Γ_2|}. \quad (2.37)
    \]

  - Another similarity measure considers the proportion of the intersection between \(Γ_1\) and \(Γ_2\) to their union \[99\]. The formula can be displayed as:

    \[
    S(R_1, R_2) = S(Γ_1, Γ_2) = \frac{|Γ_1 \land Γ_2|}{|Γ_1 \lor Γ_2|}. \quad (2.38)
    \]
• **Geometric**: The key point of the geometric measures is the computation of the distance between two fuzzy sets. No matter what form the similarity has, the distance \( \text{dist}(\Gamma_1, \Gamma_2) \) should be computed. Whereas there exist many alternative forms of \( \text{dist} \) in the existing works.

  - The basic form of the geometric measure discussed by [97] is:
    \[
    S(R_1, R_2) = S(\Gamma_1, \Gamma_2) = \frac{1}{1 + \text{dist}(\Gamma_1, \Gamma_2)},
    \tag{2.39}
    \]
    where \( \text{dist} \) can be the Hausdorff distance, or the Minkowski distance etc.

  - If the distance between center satisfy \( \|c_1 - c_2\| \rightarrow 0 \), and the distance between radii satisfies \( \|\sigma_1 - \sigma_2\| \rightarrow 0 \) at the same time, then it is natural to induce \( \text{dist}(\Gamma_1, \Gamma_2) \rightarrow 0 \). Therefore, the similarity measure can be built using the distances \( \|c_1 - c_2\| \) and \( \|\sigma_1 - \sigma_2\| \) (see [55, 56, 89] for example). Noticing that the fuzzy rules with close centres would have a very high level of overlapping, the similarity measure (e.g., [30]) can also be built based on this theory.

  - Bhattacharyya distance [8, 58] and Mahalanobis distances [54] are two similarity measures which are used to evaluate the similarity between two fuzzy sets with (multivariate) Gaussian membership functions.

To conclude, no matter which similarity measure (either set-theoretic or geometric) is used, high value of similarity \( S(R_1, R_2) \) indicates rule \( R_1 \) and \( R_2 \) are similar. These similar rules should be merged to help the system to avoid rule conflict and overfitting. The merging process of the similarity rules should be triggered by:

\[
\text{If } S(R_1, R_2) > \epsilon^{(m)} \text{, then } R_1 \text{ and } R_2 \text{ should be merged to be one rule.} \tag{2.40}
\]

Assume that the two rules \( R_i \) and \( R_j \) are about to merge. Furthermore, assume the merged rule is \( R_{i0} \), then the original two rule \( R_i \) and \( R_j \) are removed from the original system, and the merged rule \( R_{i0} \) is added to the system. Compared with the system before rule merging executed, the new system contains less number of rules. Furthermore, the merged rule \( R_{i0} \) should have its antecedent parameters and consequent parameters initialized. This process is specifically introduced in Section 2.2.3.1.2.
2.2.3.1.2 Learn The Parameters of EFSs

The formulas applied in order to update the antecedent and consequent parameters are discussed in this section.

2.2.3.1.2.A Antecedent Parameters Learning

In this section, firstly, an introduction of the approaches to update the antecedent parameters when data \( x \) is located close to the centre of a certain rule is given. This procedure happens when there is no new rule added. Secondly, this section would like to explain how to learn the antecedent parameters of a new added rule or a merged rule.

Antecedent parameters updating happens when the rule generation criterion shown in Section 2.2.3.1.1.A is not satisfied. Summarizing the existing works, the two most frequently used antecedent parameters updating methods count for: stochastic gradient decent (SGD) based approaches [90], and the Kohonen rule [100] based approaches [58]. The SGD based approaches usually update centre \( c \) and radius \( \sigma \) as shown as follows:

\[
c_i(t + 1) = c_i(t) - \eta \frac{\partial E}{\partial c_i}, \tag{2.41}
\]
\[
\sigma_i(t + 1) = \sigma_i(t) - \eta \frac{\partial E}{\partial \sigma_i}, \tag{2.42}
\]

where \( E = \frac{1}{2} [y_{t+1} - \hat{y}_{t+1}]^2 \) is the error function, and \( \eta \) is the learning rate. Using the Kohonen rule, the centre and radius can usually be updated by utilizing the difference between the new data and the centre in order to tune the old parameters. The updating formulas can be presented in (2.43) and (2.44):

\[
c_i(t + 1) = c_i(t) + \beta (x_{t+1} - c_i(t)), \tag{2.43}
\]
\[
\Sigma_i(t + 1) = \Sigma_i(t) + \beta ((x_{t+1} - c_i(t))^T (x_{t+1} - c_i(t)) - \Sigma_i(t)), \tag{2.44}
\]

where \( \beta \) is the learning rate and \( \Sigma_i \) is the covariance matrix. For the conventional fuzzy systems, \( \Sigma_i \) is a diagonal matrix. When \( \beta = 1 / N_i \), where \( N_i \) is the size of the cluster, then (2.43) presents the recursive formula of the sample mean. Despite (2.44), the recursive sample variance formula [28, 32, 39], which is presented by (2.45), is also a widely used approach:

\[
\Sigma_i(t + 1) = \Sigma_i(t) + \frac{1}{N_i} \{(c_i(t + 1) - x_{t+1})(c_i(t + 1) - x_{t+1})^T - \Sigma_i(t)\}
\]
+ (c_i(t) - c_i(t + 1))T (c_i(t) - c_i(t + 1)). \tag{2.45}

In addition, other revised version of the covariance updating formulas can be found in [38, 60, 96].

Because the structure change of the system may have an influence to the parameters of the system. Before giving the exact approaches for antecedent parameters learning, it needs to make some assumptions. Take the most widely used T-S system as an example, it needs to assume the fuzzy system \( F \) is formed by \( K \) fuzzy rules \( \{R_1, R_2, \ldots, R_K\} \). The structure change bring an influence to the number of parameters of the system. Therefore, every time when a structure change happens, there is a need to initialize the parameters of the system. The antecedent parameters initializing process regarding to rule generation, merging, and pruning are explain as follows:

1. **Rule Generation:**
   Assume the new fuzzy rule generated as \( R_{K+1} \), then the center \( c_{K+1} \) is set as the new data \( x_{t+1} \), i.e. \( c_{K+1} = x_{t+1} \). Since there is no information about the radius of this new cluster, the mean of the existing radii can be used as the initial value of \( \sigma_{K+1} \), i.e. \( \sigma_{K+1} = \left\{ \sum_{k=1}^{K} \sigma_k \right\} / K \) (see [58]). There also exist other approaches for initializing \( \sigma_{K+1} \). For example, [8] used the 1 divided by a small fraction (e.g., 1/100) to initialize the radius, [27, 29] used 1 to initialize the radius, [55] has used a \( \varepsilon \)-completeness based approach that computed the new radius based on the distance between the new center and the existing centers.

2. **Rule Pruning:**
   Assume rule \( R_i^* \) is regarded as a redundant rule, then \( R_i^* \) should be removed from the system \( F \) directly. In this case, there is no need to learn the antecedent parameters of this rule.

3. **Rule Merging:**
   Assume rule \( R_i \) and \( R_j \) are merged to \( R_{i0} \). The center \( c_{i0} \) and radius \( \sigma_{i0} \) are computed as the sample mean and the standard deviation of the data points located inside of the merged two clusters. The formulas of \( c_{i0} \) and \( \sigma_{i0} \) are given as:

\[
c_{i0} = \left\{ N_i c_i + N_j c_j \right\} / \left\{ N_i + N_j \right\}, \tag{2.46}
\]
\[(\sigma_{i_0,l})^2 = \frac{1}{N_i + N_j} \{N_i(\sigma_{i,l})^2 + N_j(\sigma_{j,l})^2 + N_i(c_i - c_{i_0,l})^2 + N_j(c_j - c_{i_0,l})^2 \} \].

This approach has been used in research works, e.g., \[39, 58\]. In \[55\], the formulas for computing \(c_{i_0}\) and \((\sigma_{i_0,l})^2\) are:

\[c_{i_0} = \frac{(\text{max}(U) + \text{min}(U))}{2}, \quad (2.48)\]
\[\sigma_{i_0,l} = \frac{(\text{max}(U) - \text{min}(U))}{2}, \quad (2.49)\]

where \(U\) is a set formed by \(U = \{c_i - \sigma_i, c_i + \sigma_i, c_j - \sigma_j, c_j + \sigma_j\}\). As two rules are merged, then, it is reasonable to think all the data points in the previous two clusters should be located inside of the merged cluster. Therefore, the number of data points in the merged cluster \(N_{i_0}\) should be computed by \(N_{i_0} = N_i + N_j\).

### 2.2.3.1.2.B Consequent Parameters Learning

The same as antecedent learning, the consequent learning should be separately discussed based on whether there is structure evolving of the system. The reason behind is that the structure evolving changes the dimension of the matrix formed by consequent parameters.

The following discussion is based on the assumption that there is no structure change of the system. SGD based and least squares (LS) based approaches are two most common optimization methods used in the existing works. The SGD based approaches \[90\] update the consequent parameters \(\psi_i\) by minimizing the error function \(E = \frac{1}{2}(y_{t+1} - \hat{y}_{t+1})^2\) at each online training step. The adaptation formula of SGD is similar to \(2.41\). SGD requires to compute the mathematical formulas for those derivatives, and this can be a difficult task when the membership functions have complex forms. Whereas the RLS based approaches (which have no requirement to compute the derivatives and can provide simple updating formulas) update the consequent parameters by solving the following global or local optimization problem:

\[\min E = \min \sum_{k=1}^{t+1} (y_k - \hat{y}_k)^2, \quad (2.50)\]
\[\text{or } \min E_i = \min \sum_{k=1}^{t+1} \theta_i(x_k)(y_k - \hat{y}_{k,i})^2, \quad (2.51)\]

where \(\hat{y}_{k,i} = xe_k^T \psi_i(k - 1)\) is the output estimated by rule \(R_i\), \(xe_i = [1, x_i^T]^T\). Regarding
the global and local optimization, RLS and weighted recursive least squares (WRLS) [27] (or recursive fuzzily weighted least squares (RFWLS) [28, 60]) are frequently applied in the above two cases, respectively. Taking the RLS used in [27] as an example, the recursive formula can be shown as:

\[
\begin{align*}
\hat{\psi}(t+1) &= \hat{\psi}(t) + P(t+1)\phi(t+1)(y_{t+1} - \phi(t+1)^T \hat{\psi}(t)), \\
P(t+1) &= P(t) - \frac{P(t)\phi(t+1)\phi(t+1)^T P(t)}{1 + \phi(t+1)^T P(t) \phi(t+1)},
\end{align*}
\]

(2.52)

(2.53)

in which \(\phi(t) = [\theta_1(x_t)xe_t^T, \theta_2(x_t)xe_t^T, \ldots, \theta_K(x_t)xe_t^T]^T, \psi = [\psi_1^T, \psi_2^T, \ldots, \psi_K^T]^T, y_t = [\psi_{t,0}, \psi_{t,1}, \ldots, \psi_{t,n}]\). The WRLS can be presented by:

\[
\begin{align*}
\hat{\psi}_i(t+1) &= \hat{\psi}_i(t) + \theta_i(x_t)P_i(t+1)xe_{t+1} (y_{t+1} - xe_{t+1}^T \hat{\psi}_i(t)), \\
P_i(t+1) &= P_i(t) - \frac{\theta_i(x_t)P_i(t)xe_{t+1}xe_{t+1}^T P_i(t)}{1 + \theta_i(x_t)xe_{t+1}P_i(t)xe_{t+1}}.
\end{align*}
\]

(2.54)

(2.55)

The optimality of the global learning approach, RLS, would be severely disturbed by the rule number and parameters evolving, and there is a need to develop an appropriate approach in order to update the covariance matrix \(P(t)\) used in the computation, when there is a structure or parameter evolving [27, 101]. Estimating the consequent parameters locally and individually is regarded to be a much more accurate approach than the global learning, due to the fact that rule evolving can only influence the optimality through the fuzzy weights, in which way the negative influence brought by the rule evolving can be decreased with a large degree.

Consider there is a structure change, e.g., rule generation, merging, pruning, the approaches mentioned above are not suitable to learn the consequent parameters directly. The reason behind is that the structure evolving changes the dimension of the parameters. The following bullet lists the most frequently used approaches to initialize the consequent parameters when these structure changes happen.

1. **Rule Generation:**

   The introduction about how to initialize the consequent parameters of the new rule are bare. Herein, a brief introduction of the most classic approach used in [27] and [29] is given. Assume \(R_{K+1}\) is the new added rule. Before updating the consequent parameters, the consequent parameters of this rule should be initialized first. As there is no prior knowledge of the consequent parameters of the new rule,
so the consequent parameters can be initialized by the the mean or the weighted average of the consequent parameters of the existing rules. To be more specific, the initial value of consequent parameters of rule $R_{K+1}$ can be initialized using the following formulas:

$$
\psi_{K+1} = \frac{1}{K} \sum_{k=1}^{K} \psi_k,
$$

or

$$
\psi_{K+1} = \frac{1}{K} \sum_{k=1}^{K} \theta_k(x) \psi_k.
$$

Consider RLS is used to update the consequent parameters, then $\phi(t)$, $\hat{\psi}(t)$ and $P_{K+1}(0)$ used in (2.52) and (2.53) should be computed by:

$$
\phi(t) = [\theta_1(x_t)xe_t^T, \theta_2(x_t)xe_t^T, \ldots, \theta_K(x_t)xe_t^T, \theta_{K+1}(x_t)xe_t^T],
$$

$$
\hat{\psi}(t) = [\psi_1^T, \psi_2^T, \ldots, \psi_K^T, \psi_{K+1}^T]^T
$$

$$
P(t) = \begin{bmatrix} P(t)^{(0)} \\ \Omega_0 I_{(n+1)\times(n+1)} \end{bmatrix},
$$

where $P(t)^{(0)}$ is the one before adding a new rule, $\Omega_0$ can be set as a large number, e.g., 1000 or 10000, and $I_{(n+1)\times(n+1)}$ is an identity matrix. Thereafter, the consequent parameters can be updated using (2.52) and (2.53).

Unlike RLS, only $P_{K+1}(0)$ should be initialized when using WRLS. $P_{K+1}(0)$ is initialized as a diagonal matrix with $P_{K+1} = \Omega_0 I_{(n+1)\times(n+1)}$, where $\Omega_0$ and $I_{(n+1)\times(n+1)}$ are the same as above. Then, the consequent parameters of the added rule can be updated using WRLS shown in (2.54) and (2.55).

2. Rule Pruning:

Relaying on [27], assume rule $R_i$ is discarded. There is no influence for WRLS but removing $\hat{\psi}_i(t)$, $\phi_i(t)$, and $P_i(t)$. However, the dimension of matrices used in RLS should be changed. These changes are shown in the formulas below:

$$
\phi(t) = [\theta_1(x_t)xe_t^T, \ldots, \theta_{i-1}(x_t)xe_t^T, \theta_{i+1}(x_t)xe_t^T, \ldots,
$$

$$
\theta_K(x_t)xe_t^T],
$$

$$
\psi(t) = [\psi_1^T, \ldots, \psi_{i-1}^T, \psi_{i+1}^T, \ldots, \psi_K^T]^T.
$$
3. **Rule Merging**:

If two rules merged, the number of rules will reduce from $K$ to $K - 1$. Assume $R_i$ and $R_j$ are two rules, which they about to merge. With the information of these two rules, the consequent parameters of the merged rule $R_{i0}$ can be initialized as

$$
\psi_{i0} = \left( N_i \psi_i + N_j \psi_j \right) / (N_i + N_j).
$$

For RLS, $\psi(t)$ and $\phi(t)$ should be initialized by:

$$
\phi(t) = [\theta_1(x_t) xe_t^T, \ldots, \theta_{i-1}(x_t) xe_t^T, \theta_{i0}(x_t) xe_t^T, \theta_{i+1}(x_t) xe_t^T, \ldots, \theta_{j-1}(x_t) xe_t^T, \theta_{j1}(x_t) xe_t^T, \ldots, \theta_{K}(x_t) xe_t^T],
$$

$$
\psi(t) = [\psi_1(t), \ldots, \psi_{i-1}(t), \psi_{i0}(t), \psi_{i+1}(t), \ldots, \psi_{j-1}(t), \psi_{j1}(t), \ldots, \psi_K(t)]^T.
$$

$P(t)$ is initialized by $\Omega_0 I_{(K-1)(n+1) \times (K-1)(n+1)}$.

For WRLS, $P_{i0}(t)$ and $\phi_{i0}$ should be initialized. Based on the information of the two clusters, $P_{i0}$ is calculated as $P_{i0} = (N_i P_i + N_j P_j) / (N_i + N_j)$. Parameter $\phi_{i0}$ can be computed directly as $\phi_{i0} = \theta_{i0}(x_t) xe_t^T$.

### 2.2.3.1.3 Frameworks of EFSs

Methods of how structure and parameters should be learnt for an EFS have been specifically introduced in Sections 2.2.3.1.1 and 2.2.3.1.2. However, it is still unclear about how an EFS learning approach works. In order to give a more visible explanation, most of the existing EFSs online learning approaches are summarized using a general framework shown in Fig. 2.5. In Fig. 2.5, it assumes the EFS that learnt and updated after using the input-output pair $(x_t, y_t)$ is $F_t$. This figure presents how to update to obtain the updated EFS $F_{t+1}$ using $(x_{t+1}, y_{t+1})$.

### 2.2.4 Critical Analysis

A number of different architecture and learning approaches for EFSs have been summarized in Sections 2.2.2 and 2.2.3. The previous works that implement the frameworks introduced in Section 2.2.2 and evolving criteria shown in Section 2.2.3.1 have been numerically tested as effective methods. However, all these approaches still suffer from technical limitations. These are summarized as follows, in order to highlight the research motivation of this thesis on EFSs.
The learnt EFS: $F_t$

Read $(x_{t+1}, y_{t+1})$

Check a rule generation criterion following from Section 2.2.3.1.1.A.

- **yes**
  - Generate a new rule following from Section 2.2.3.1.1.A.
  - A new fuzzy rule is generated.
  - Check a criterion for rule merging following from Section 2.2.3.1.1.B.
  - Merge similar rules following from Section 2.2.3.1.1.B.
  - Check a criterion for rule pruning following from Section 2.2.3.1.1.B.
  - Prune redundant rules following from Section 2.2.3.1.1.B.
  - Compute online training result $\hat{y}_{t+1}^{\text{train}}$.

- **no**
  - Update antecedent parameters based on Section 2.2.3.1.2.A.

Compute online training result $\hat{y}_{t+1}^{\text{train}}$.

Update the consequent parameters following from Section 2.2.3.1.2.B, and set $t = t + 1$.

Compute the prediction of $y_{t+1}$: $\hat{y}_{t+1}^{\text{test}} = F_t(x_{t+1})$.

The learnt EFS: $F_{t+1}$.

update the EFS by $F_{t+1}$

Fig. 2.5: The general framework of an EFS learning approach.
(1) Most existing EFSs and their online learning algorithms focus on adapting the structure and parameters of a fuzzy system by fitting more recent data. These structure evolving strategies are often designed from a heuristic manner similar to online clustering approaches which are usually one-pass and unsupervised (e.g. [35,102]). This is certainly correct, if learning an evolving system is the aim to achieve. Furthermore, the ignorance of the connection between consequent and antecedent learning of most EFSs online identification methods may lead to an “unlearning effect”. To be more specific, in many applications, the streaming data arrived in each given period of time are often local data (that is, with a given time period, all data arrived are located at a sub-region of the input space). In such cases, the obtained fuzzy system by these existing learning algorithms end up as a good local model which fits well with the latest arrival data accurately but a poor global model which fits the historical data very inaccurately. As the accuracy measurement of EFSs is usually the one-step ahead or testing error, therefore, such a global accuracy issue has been largely ignored in many of the existing approaches. To illustrate this point in more detail, a simulation example is given in Section 3.3.4.2. An existing method which effectively deals with this “unlearning effect” is proposed in [57], which has developed a consequent learning method using adaptive forgetting factors that updated by the recent contribution of a cluster to enable a fuzzy system keeps its global accuracy. In some sense, this is an indirect approach to addressing this issue by using different adaptive forgetting factors for different rules. Furthermore, [103] considers to develop connections between antecedent and recursive consequent learning by the aim of ensuring optimality of WRLS, but the exact calculation and proofs have not been given. Hence, this ignored influence of the rule base evolving strategies to the optimality of recursive consequent learning method needs to be further investigated.

(2) When the structure of the fuzzy system is unchanged and summarizing these previous state-of-the-art EFSs identification methods, it can be seen that the most widely used consequent learning method is a local optimum method known as WRLS (e.g. [30, 58, 69, 73]) rather than the global version (i.e. RLS). The reason behind this is that WRLS minimizes a bunch of local error functions, which hence enables more flexible structure evolving. It can be seen from these existing approaches, that many different kinds of generalized versions of WRLS have been plugged-in learning consequent parameters. [55] has used an extended recursive least squares (ERLS) to minimize local weighted error functions. This ERLS has been proven to
have a small finite system error. In [92], WRLS has been used to update the consequent parameters, but the samples that are used in this updating procedure should be selected in order to avoid underfitting in a high noisy region through taking the uncertainty in model outputs into consideration. Besides, [56] has applied a fuzzily weighted generalized recursive least squares (FWGRLS) that extends the generalized recursive least squares (GRLS) [104] to a local learning version and adds a weight decay term in the error function. This decay term conveys the prior distribution of the local model, so as to make sure of the convergency and to enhance the generalization ability. Furthermore, this FWRLS has been further generalized to learn the consequent parameters of type-2 fuzzy systems in [89]. However, using both RLS and WRLS directly to learn the consequent parameters are approximation ways, because there is a need to assume that the rule base is unchanged in order to ensure they are accurate methods [27]. If the structure of the rule base is unchanged (i.e. if there is no rule generation, deleting or merging) or the antecedent parameters are unchanged (i.e. the centres and radii of membership functions remain the same), then such a parameter updating algorithm is an optimal approach. Nevertheless, with the structure or antecedent parameters of the rule base continuously changing during the learning process of an EFS, such a direct application of WRLS means that it is hard to ensure the local optimality of the estimation of the consequent parameters. Some existing results, such as [28] and [55] firstly addressed the issue that any structural change of the antecedent part will affect the optimality and convergency of the consequent part, and a sub-optimal solution was proposed. However, there is no existing work which gives the explicit calculation of the optimal consequent parameters and assures their optimality.

(3) As discussed above, the rule generation and simplification of the EFSs should be determined by some criteria. However, a fundamental problem that has not been properly solved is that of how to determine the right thresholds for such criteria. The thresholds are very crucial in controlling the speed for the rule growth and accuracy of the systems. The thresholds can be set to make sure the rule number grows slowly and obtains a simple and coarse system with very few big clusters; and, on the other hand, the thresholds can be set to allow a high rule growth speed, thus obtaining an overcomplicated model with many tiny clusters. It can be seen from this phenomenon that the criteria controlled by inappropriate thresholds are likely to cause underfitting or overfitting issues. Unfortunately, the current practice of setting such control thresholds as fixed values based on the experienced values
or trial-and-error offline experiment. The experienced values do not work as the different threshold values. These are needed when learning different systems, and there are no uniform values, which fit all the systems. Strictly, setting the individual value for each system to be learnt based on the offline experiment is inappropriate, as data streams always have a dynamic nonstationary and nonlinear phenomenon. Fixed thresholds are hard to generate a fuzzy system with appropriate complexity to approximate the data stream. Furthermore, a fixed threshold is hard to guarantee that new generated rules can ensure the reduction of prediction errors. Therefore, in many applications, it is difficult to find a fixed-value threshold, no matter whether it is based on experience or experiment, to make an EFS evolve effectively and accurately according to the state and the need of a data stream.

(4) Both set-theoretic and geometric similarity measures face two common challenges: a) The direct usage of Gaussian membership functions makes it hard to meet the requirement of online learning with regard to computation speed [87]; b) Approximations of Gaussian membership functions and the heuristic similarity measures mean that it is difficult to accurately measure the rule similarities. To be more specific, on one aspect, set-theoretic similarity measures are usually computationally expensive when using Gaussian or bell-shaped membership functions [99], because of the difficulty in computing the intersection of fuzzy sets even for offline learning. Many alternative methods have been proposed in previous works, such as using triangle [61] or trapezoidal [99], to approximate the Gaussian membership function based on the $\alpha$-cut of the fuzzy set. These approximated measures are inaccurate due to the different shapes between Gaussian and the approximated membership functions. On the other aspect, geometric measures are distance-based measures and easier to compute, as only the distance between the membership functions needs to be calculated, and so they are widely used for online learning. Considering the computation speed, inaccurate and intuitive approaches (e.g., distances between cluster centres or radii) have been frequently applied. The assumption behind these measures is that, if the parameters or the statistical samples present extremely similar behaviour, then the firing strengths would have high similarity. Unfortunately, the existing approaches are approximate, or heuristic, and so are inaccurate; as a result, they can lead to the wrong merge. Therefore, there is a need to propose a similarity measure in order to tackle these problems.


2.3 Functional Data Regression

Functional data have attracted increasingly attention in recent years, since the improvement of modern technology to record and restore a vast number of data. Functional data are infinite dimensional. For example, assume $t$ presents time, $x(t)$ presents the temperature of a day. The temperature $x(t)$ is a continuous function of $t$, thus, an infinite dimensional data. They can be sampled dense, sparse, and even irregular. Huge information contained in the samples of functions provide opportunities for the development of a statistical research area — FDA, which is the term firstly proposed in [105]. The major characteristic for FDA is to treat each function as a single component instead of the collection of data samples [106]. This thinking promotes the development of possible approaches to dealing with the rich information behind the huge data samples and the irregularly spaced data. FDA also includes a vast number of research topics, for example: Clustering, e.g. k-means [107], functional principal component analysis (FPCA) scores based method [108,109], Bayesian based approaches [110,111]; Classification, e.g. support vector machine [112], functional discriminant analysis [113,114]; Regression, e.g. functional linear model (FLM) [115]. Early systematic discussion can be found from [115]. Besides, there are some classic text books about FDA counting for [116–118], which cover the basic knowledge and techniques for dealing with the functional data. The website: “http://www.psych.mcgill.ca/misc/fda/downloads/FDAfuns/” provides the R, SPLUS, MATLAB toolboxes for the approaches and datasets discussed in these books. Recent excellent reviews for FDA and functional regression can be found in [119] and [106], respectively.

This thesis is particularly interested in the functional regression problems. Based on [106], functional data regression problem can be divided into three categories: (i) functional input with scalar output (scalar-on-function), (ii) functional output with scalar input (function-on-scalar), (iii) functional input and output (function-on-function). Although this thesis only concentrates on proposing the function-on-function regression approaches, important literatures and models of all these three categories will be summarized in this section.

Before reviewing the existing literatures about functional regression in Sections 2.3.3, 2.3.4 and 2.3.5, some basic knowledge of functional analysis are presented in Section 2.3.1, and important data representing methods are summarized in the Section 2.3.2. At last, a critical analysis for functional regression approaches are presented in Section
2.3.6.

2.3.1 Functional Analysis Basics

In order to give a better understanding of the research about FDA, some basic definitions and theories regarding to functional analysis will be briefly introduced in this section.

**Definition 2.2. Banach Space:** A Banach space is a complete norm linear space.

**Definition 2.3. Hilbert Space:** A Hilbert space is a complete inner product space.

Based on these basic definitions in functional analysis, the following theorem presents the well-known Riesz representation theorem. This representation theorem is very important to some functional data regression approaches, which will be summarized in the following subsections.

**Theorem 2.1. Riesz Representation Theorem:** Assume $H$ is a Hilbert Space, and its dual space is $H^*$. Assume $f \in H^*$, then there exists $g \in H$ such that $\forall \phi \in H$ (2.65) holds:

$$f(\phi) = \langle \phi, g \rangle.$$  

Furthermore, $\|g\|_H = \|f\|_{H^*}$.

**Definition 2.4. $L^p$ (1 ≤ p < ∞) Space:** Assume $(X,S,\mu)$ a measurable space, and $f : X \to \mathbb{R}$ (or $\mathbb{C}$) is a real (or complex) valued measurable function. $L^p$ is a space formed by all the $p$-integrable measurable functions $f$, i.e.,

$$\left(\int_X |f|^p d\mu\right)^{\frac{1}{p}} < \infty.$$  

(2.66)

An example of $L^p$ space is $L^p[a,b]$ (1 ≤ $p$ < ∞, and $-\infty < a < b < \infty$). $L^p[a,b]$ stands for the set of all the functions that satisfies $(\int_a^b |f(t)|^p dt)^{\frac{1}{p}} < \infty$.

**Definition 2.5. Measure $\| \cdot \|_p$ on $L^p$ (1 ≤ p < ∞) Space:** Measure $\| \cdot \|_p$ on an $L^p$ space is defined as:

$$\|f\|_p = \left(\int_X |f|^p d\mu\right)^{\frac{1}{p}}.$$  

(2.67)
Proposition 2.1. An $L^p$ space is a Banach space.

Proposition 2.2. When $p = 2$, then $L^p$ space is a Hilbert space (i.e., $L^2$ space is a Hilbert space).

Definition 2.6. Inner Product on $L^2$ space: Taking $L^{a,b}$ as an example, the inner product on $L^2[a,b]$ is $<f, g> := \int_a^b f(x)\overline{g(x)} \, dx$, $f, g \in L^2[a,b]$,

where $\overline{g(x)}$ is the conjugate of $g(x)$.

Definition 2.7. Operator: An operator $F : X \to Y$ is a mapping that maps $X$ to $Y$.

Definition 2.8. Infinite and Finite Dimensional Vector Space: A vector space $X$ is said to be finite dimensional when it has a basis with finite number of elements. Otherwise, $X$ is an infinite dimensional space.

In order to better explain the definition of finite and infinite dimensional spaces, some typical examples are given below:

1. $\mathbb{R}^n$: It is obvious that $\{e_1, \ldots, e_i, \ldots, e_n\}$ (Each $e_i$ is the unite vector with the $i$-th element equals to 1 and others equal to 0, for example, $e_1 = (1,0,\ldots,0)$, $e_2 = (0,1,0,\ldots,0)$, and $e_n = (0,\ldots,0,1)$.) forms the standard basis of $\mathbb{R}^n$. Basis $\{e_1, \ldots, e_i, \ldots, e_n\}$ contains $n$ elements with $n < \infty$, therefore, $\mathbb{R}^n$ is a finite dimensional space.

2. $L^2[0,1]$: A classic basis of $L^2[0,1]$ can be presented as $\{\exp\{2\pi imx\} : m \in \mathbb{Z}\}$, where $i$ is the imaginary unit. This basis has infinite many elements, therefore, $L^2[0,1]$ is an infinite dimensional space.

2.3.2 Data Representing, Statistics

Since the data are actually observed in a discrete sense, when getting the observations of the functional data, the very first thing that needs to be done is to represent the data as functions. Depending on different characteristics, different kinds of basis functions are used to do this job. The most frequently-used bases are: (1) The spline (B-spline)
basis is suitable for non-periodical data; (2) The Fourier Basis is particular powerful to the periodical data; (3) The wavelet basis can deal with the time-frequency from the data instead of only the frequency extracted by the Fourier basis. Additionally, FPCA can be regarded as both the dimension reduction tool and the basis to represent the functional data. As well as these basis functions, there are also many other basis such as exponential basis, polynomial basis, power basis, etc.

In the following parts of this section, some basic statistics, e.g., mean, variance, covariance, and frequently used basis functions are introduced. For the convenience of expression, some assumptions should be made as: \( \{ x_i(t) \}_{i=1}^{N} \) are a series of functions, and \( \{ x_i(t_1), x_i(t_2), \ldots, x_i(t_{M_i}) \}_{i=1}^{N} \) are the observations. Note that \( M_i (i = 1, 2, \ldots, N) \) do not have to be equal.

### 2.3.2.1 Basic Statistics

Based on the textbook [116], the mean, variance, covariance and cross-covariance can be computed as follows (Note that \( x(t) \) in this section is a function instead of a real number):

- **Mean:** \( \bar{x}(t) = \frac{1}{N} \sum_{i=1}^{N} x_i(t) \).

- **Variance:** \( \text{Var}_X(t) = \frac{1}{N} \sum_{i=1}^{N} (x_i(t) - \bar{x}(t))^2 \).

- **Covariance:** \( \text{Cov}_X(s,t) = \frac{1}{N} \sum_{i=1}^{N} (x_i(s) - \bar{x}(s))(x_i(t) - \bar{x}(t)) \).

### 2.3.2.2 Basis Functions

Let \( x(t) \) be a function in a linear functional space \( X \). (The same as Section 2.3.2.1, \( x(t) \) is a function rather than a real number.) In many real applications, it is almost impossible to know the full information of \( X \). Therefore, a safe way is to assume \( X \) is an infinite dimensional space. Generally, some function values of a certain function \( x(t) \) are the only things that can be observed, instead of knowing the analytic form of \( x(t) \). For example, \( \{x(t_1), x(t_2), \ldots, x(t_m)\} \), which are the function values of \( x(t) \) on discrete points \( \{t_1, t_2, \ldots, t_m\} \), can be observed. However, no clues can show whether \( x(t) \) has an analytic form such as \( x(t) = at + b, x(t) = at^2 + bt + c \), or etc. Therefore, before any
analysis of \( x(t) \) can be made, the very first step should be representing \( x(t) \) using the information of the observations.

It is naturally to represent \( x(t) \) using the basis of \( X \). Assuming \( \{b_1(t), \ldots, b_n(t), \ldots \} \) is a group of bases of \( X \), then, any \( x(t) \in X \) can be presented by \( x(t) = \sum_{i=1}^{\infty} \alpha_i b_i(t) \) (\( \alpha_i \) are the coefficients). There is a computational difficulty when using the infinite basis to represent \( x(t) \). Therefore, basis are used to represent the function \( x(t) \) in a infinite dimensional space with finite numbers of coefficients and functions. The basis functions can help to realize this, thus, to represent function \( x(t) \) as a linear combination of the basis functions:

\[
x(t) = \sum_{p=1}^{P} \alpha_p B_p(t).
\]  

Note that (2.69) is a \( P \)-truncation of \( x(t) = \sum_{p=1}^{\infty} \alpha_p B_p(t) \). The reason behind is that \( P \) basis functions are used to present \( x(t) \) rather than infinite basis functions. The well-known spline basis, Fourier basis, and wavelet basis, as well as FPCA and PCs, will be demonstrated.

- **B-spline Basis:**

  From the text book [120], it can be seen that the B-spline can be presented by the recursive formulas. Assuming \( t_1, t_2, \ldots, t_{P+n+1} \) are the knots, then the B-spline basis with \( n \)-th degree, \( B_{p,n} \), can be recursively calculated by:

  \[
  B_{p,n}(t) = \frac{t-t_p}{t_{p+n}-t_p} B_{p,n-1}(t) + \frac{t_{p+1+n}-t}{t_{p+1+n}-t_{p+1}} B_{p+1,n-1}(t), \]

  and the \( p \)-th basis spline with degree zero can be presented as:

  \[
  B_{p,0}(t) = \begin{cases} 
  1 & t_p \leq t < t_{p+1} \\
  0 & \text{otherwise}.
  \end{cases}
  \]  

- **Fourier Basis:**

  Fourier basis is the orthogonal basis system. The periodical Fourier basis has the form: \( B_0(t) = 1, B_{2r-1}(t) = \sin(r \omega t) \) and \( B_{2r}(t) = \cos(r \omega t) \). In detail, a function
The function \( x(t) \) approximated by the Fourier basis can be presented as:

\[
x(t) = \alpha_0 + \alpha_1 \sin(\omega t) + \alpha_2 \cos(\omega t) + \alpha_3 \sin(2\omega t) + \alpha_4 \cos(2\omega t) + \ldots \quad (2.72)
\]

- **Wavelet Basis:**
  As an orthogonal basis system, the wavelet basis is constructed on the *mother wavelet* \( \psi(t) \). The wavelet basis can approximate the function \( x(t) \) through:

\[
x(t) = \sum_{p,k} \alpha_{p,k} \psi_{p,k}(t), \quad (2.73)
\]

in which \( \psi_{p,k}(t) = 2^{p/2} \psi(2^p t - k) \).

- **FPCA & PCs:**
  FPCA was first proposed in [121]. It is not only a data representation approach, but also an important dimension reduction approach. Based on Mercer’s theorem, the spectral decomposition of the covariance \( \text{Cov}_X(s,t) \) presented in the Section 2.3.2.1 can be expended as

\[
\text{Cov}_X(s,t) = \sum_{j=1}^{\infty} \lambda_j \phi_j(s) \phi_j(t), \quad (2.74)
\]

where \( \lambda_j \) are the eigenvalues that are arranged in descending order, and \( \phi_j(\cdot) \) are the corresponding orthogonal eigenfunctions. The Karhunen-Loève (K-L) expansion gives that \( x(t) \) is

\[
x(t) = \bar{x}(t) + \sum_{j=1}^{\infty} C_j \phi_j(t), \quad (2.75)
\]

in which \( \bar{x}(t) \) is the mean, \( C_j \) are the PCs with the form \( C_j = \int (x(t) - \bar{x}(t)) \phi_j(t) dt \).

In practice, the number \( J \) is selected to present \( x(t) \) as \( x(t) = \bar{x}(t) + \sum_{j=1}^{J} C_j \phi_j(t) \).

### 2.3.3 Scalar-on-Function Regression

Scalar-on-function models have functional inputs and scalar outputs. The final aim for scalar-on-function regression problem is learning the operator \( F \) that maps a functional space \( X \) to a real space \( Y \subset \mathbb{R} \). When considering the regression problems, the
simplest model should be the linear model. In this section and the following two sections, the existing works about the functional regression starting from the linear model are summarized. As presented in [115], the linear model with function input has the form:

$$y = \alpha + \int \beta(t)x(t)dt + \varepsilon,$$  \hspace{1cm} \text{(2.76)}

where $y, \alpha, \varepsilon \in \mathbb{R}$ are real numbers, $\beta(t)$ is the function coefficient, $x(t)$ is the function input. Identifying $\beta(t)$ is the foremost matter that needs to be considered, when using (2.76). Despite proposing various models, many research works concentrate on estimating the coefficient $\beta(t)$. The approach in [115] has suggested to identify (2.76) from representing the infinite functions $\beta(t)$ and $x(t)$ using basis functions (e.g., Fourier basis): $\beta(t) = \sum_{i=1}^{n_1} b_i \psi_{i,j}(t)$ and $x(t) = \sum_{i=1}^{n_2} c_i \psi_{2,i}(t)$, respectively. Thereafter, $\beta(t)$ estimation problem is converted into estimating the $\{b_i\}_{i=1}^{n_1}$ (which are used to represent $\beta(t)$) in a real space. Using the least squares to minimize the error function $E$:

$$E = \sum_{j=1}^{N} \{y_j - \alpha - \int (\sum_{i=1}^{n_1} b_i \psi_{1,i}(t)) (\sum_{i=1}^{n_2} c_i \psi_{2,i}(t))dt\}^2 + \lambda \int \{L \sum_{i=1}^{n_1} b_i \psi_{1,i}(t)\}^2 dt$$  \hspace{1cm} \text{(2.77)}

where $L$ is a linear differential operator, the estimated $\{b_i\}_{i=1}^{n_1}$ and $\beta(t)$ can be obtained. In [122], model (2.76) has been transformed to multivariate problems in $\mathbb{R}^N$ in order to estimate the discrete values of $\{\beta(t_i)\}_{i=1}^{N}$. Then the function form of $\beta(t)$, then, can be obtained from applying the spline interpolation. The method proposed in [123] uses spline basis to represent $\beta(t)$ and to estimate the coefficients though least squares with a penalty being responsible for the smoothness. A wavelet-based approach has been proposed in [124]. The model (2.76) has been rewritten using the wavelet decomposition of $x(t)$; Then, functional coefficient has been estimated based upon the maximum likelihood estimation (MLE) with $l-1$ penalty, when Gaussian distribution assumption is set on the conditional distribution of $y$. The approach proposed in [125] treats model (2.76) as $y = \Psi(x) + \varepsilon$ with $\Psi(\cdot)$ is the real-valued continuous linear operator on the Hilbert space; then, $\Psi(\cdot)$ is approximated by the FPCA and Hilbertian auto-regression processes [126]. In [127], the above linear model has been investigated in the reproducing kernel Hilbert space (RKHS). To be more specific, $\beta(t)$ was represented by the reproducing kernel $K(s,t)$ from $\beta(t) = \sum_{i=1}^{N} b_i \int K(s,t)x_i(s)dt$, in which $K(s,t) = \sum_{k=1}^{\infty} \theta_k \psi_k(s) \psi_k(t)$ and $x_i(t) = \sum_{k=1}^{\infty} c_{ik} \psi_k(t)$ with $\psi_k(t)$ are orthogonal bases obtained from FPCA. Furthermore, $\beta(t)$ can be estimated by $\min\{\sum_{i=1}^{N} (y_i - \alpha - \int \beta(t)x_i(t)dt)^2 + \lambda \int (\beta''(t))^2 dt\}$. 
Some extensions of the functional linear model have been proposed to enable a wider range of data to be handled. A mixed model known as the partially functional linear model (which has the inputs mixed by the function and vector) was studied in [128]. The model has the following form:

$$ y = \sum_{j=1}^{d} \int \beta_j(t)x_j(t)dt + Z^T \gamma + \epsilon, \quad (2.78) $$

in which $y, \epsilon \in \mathbb{R}$ are real numbers, $d \in \mathbb{N}_+$ is a positive natural number, $\beta_j(t)$ ($j = 1,2,\ldots,d$) are functional coefficient, $x_j(t)$ ($j = 1,2,\ldots,d$) are function inputs in functional spaces $X_j$ ($j = 1,2,\ldots,d$), respectively, $Z$ is the scalar covariate, and $\gamma$ is the coefficients that need to be estimated. The input $x_j(t)$ and coefficient $\beta_j(t)$ were represented using the K-L expansion and FPCA basis of $X_j$. Finally, the coefficients in (2.78) were estimated though least squares with penalties on both $\beta_j(t)$ and $\gamma$. The number of bases and the tuning parameters were suggested to be selected by BIC and AIC criteria.

Since the linear model is lack of ability to explain the complex phenomenon contained in the data, there also exist many research works in constructing models to explore the nonlinear and complex relationship between the data. Several kinds of functional nonlinear models have been investigated. For example, a generalized functional linear model has been proposed in [129] with form (2.79):

$$ y = g(\alpha + \int \beta(t)x(t)dt) + \epsilon, \quad (2.79) $$

where $y, \epsilon \in \mathbb{R}$ are real numbers, $g(\cdot)$ is a monotone and twice continuously differentiable function, $\beta(t)$ is the function coefficient, $x(t)$ is the function input in function space $X$. The same set of orthogonal bases were used to represent $\beta(t)$ and $x(t)$, and the parameters of these basis were estimated by a quasi-likelihood approach.

Research works such as [130] and [131] studied functional generalized additive model:

$$ g(E[y_i|x_i(t)]) = \alpha + \int F(x_i(t),t)dt, \quad (2.80) $$

where $y_i, \alpha \in \mathbb{R}$ are real numbers, $g(\cdot)$ is a known function, $x_i(t)$ are function inputs, and $F(x_i(t),t)$ is a smooth function to be estimated. The linear model (2.76) is the
special case of (2.80) when setting \(g(x) = x\) and \(F(x, t) = \beta(t)x(t)\). In \([130]\), the function \(F(x_i(t), t)\) was rewritten using tensor product: \(F(x_i(t), t) = \sum_j \sum_k \theta_{j,k} B_j^x(x) B_k^t(t)\).

In order to avoid zero columns of the design matrix caused by no observations appear on certain grids of the support covered by the splines, \(x_i(t)\) was transformed to \(G_t(x_i(t)) = P(x_i(t) < x_i)\) and estimated by the empirical cumulative distribution function (CDF) of \(G_t(x_i(t))\). The coefficients \(\theta_{j,k}\) were estimated from maximize the penalized log-likelihood with the density of \(y\) being assumed as exponential family. The roughness penalties were chosen along with two directions \(x\) and \(t\). In \([131]\), a different approach — a Bayesian Markov Chain Monte Carlo (MCMC) approach was proposed for searching the estimation of the \(\theta_{j,k}\) of the tensor product form of \(F(x_i(t), t)\).

Another nonlinear model named as the functional additive regression model:

\[
y_i = \sum_{j=1}^{p} f_j(x_{ij}(t)) + \varepsilon_i, \tag{2.81}
\]

where \(y_i, \varepsilon_i \in \mathbb{R}\) are real numbers, \(p \in \mathbb{N}_+\) is a positive natural number, \(x_{ij}(t)\) are function inputs, and \(f_j(\cdot)\) are general linear functions of \(x_{ij}(t)\). It has been studied by \([132]\). This model is a generalized model, which can deal with both linear and nonlinear problem and the high-dimensional problem. In order to estimate the model, \(f_j(x_{ij}(t)) (j = 1, \ldots, p)\) were represented by a selected nonlinear function \(g_j(\cdot)\) from: \(f_j(x_{ij}(t)) = g_j(\int \beta_j(t)x_{ij}(t)dt)\). Furthermore, model (2.81) can be represented as:

\[
y_i = \sum_{j=1}^{p} g_j(\int \beta_j(t)x_{i,j}(t)dt) + \varepsilon_i, \tag{2.82}
\]

where \(g_j(\cdot)\) and \(\beta_j(\cdot)\) are both nonparametric smooth functions. \([132]\) estimated the model (linear or nonlinear) from proposing an iterative approach to minimize the error function: \(\frac{1}{N} \sum (y_i - \sum_{j=1}^{p} f_j)^2 + \sum_{j=1}^{p} \lambda(\frac{1}{\sqrt{N}} \sqrt{\int f_j^2 dt})\). The tuning parameter \(\lambda\) is suggested to be identified using cross-validation.

Without setting the model to be linear (or a certain nonlinear) form, existing works have made achievements by studying the generalized model:

\[
y = f(x(t)) + \varepsilon, \tag{2.83}
\]

where \(y, \varepsilon \in \mathbb{R}\) are real numbers, \(x(t)\) is the function input, \(f(x(t))\) is an unknown smooth regression function. In order to estimate (2.83), nonparametric approaches have
been proposed and used in [133–136]. Compared with parametric models, the nonparametric approaches can essentially be used in any situations, and are particularly suitable for the general model (2.83), taking the advantage of that nonparametric approaches have no requirement to know the prior knowledge about the model form. The method proposed in [134, 135] gives a nonparametric kernel estimator of $f(\cdot)$:

$$
\hat{f}(x(t)) = \frac{\sum_{i=1}^{N} y_i K(d(x_i(t), x(t))/h)}{\sum_{i=1}^{N} K(d(x_i(t), x(t))/h)},
$$

(2.84)

where $K(\cdot)$ is the kernel, $d(\cdot, \cdot)$ is the semi-metric, $h$ is the bandwidth, and $\{x_i(t)\}_{i=1}^{N}$ are the samples of input functions. Asymptotic theory of $\hat{f}$ has been derived in these two papers. Based on [134] and the kernel estimator (2.84), a bootstrap approach with the stochastic disturbance $\varepsilon$ has been proposed in [133]. The research in [136] has demonstrated that the curse of dimensionality of the nonparametric approach can be solved by setting $d(\cdot, \cdot)$ as a semi-metric. Despite of the kernel methods, the method proposed in [137] approximates $f(x(t))$ using a polynomial of degree 1, then for every $z(t)$ in the neighbourhood of $x(t) \in L^2[0,1]$ there exists an $f(z(t)) \approx f(x(t)) + (b(x(t)), z(t) - x(t))$, where $b(x(t)) \in L^2[0,1]$, and $(\cdot, \cdot)$ is the $L^2[0,1]$ inner product. The problem can be further transformed to estimating the coefficients in a real space by ordinary least squares method, when representing $b(x(t)) = \sum_{j=1}^{J} b_j \psi_j(t)$ and $x_i(t) - x(t) = \sum_{j=1}^{J} c_{ij} \psi_j(t)$ using the K-L expansion and orthonormal bases $\{\psi_j(t)\}_{j=1}^{\infty}$. The error function for least squares method can be presented as follows:

$$
E = \sum_{i=1}^{N} K(\|x_i(t) - x(t)\|/h) \{y_i - (a + \sum_{j=1}^{J} b_j c_{ij})\}^2,
$$

(2.85)

where $a \approx f(x_i(t))$.

More nonparametric FDA approaches can be found in a detailed summary [138]. Recent work on nonparametric approaches can also be found in [139], whose corresponding R functions have been imbedded in the R package poridge.
Function-on-Scalar Regression

Function-on-scalar models have scalar inputs and functional outputs. For function-on-scalar regression problems, the operator \( F \) that maps a real space \( X \subset \mathbb{R}^n \) to a functional space \( Y \) should be learned. A basic model for function-on-scalar regression investigated by [140] has the form:

\[
y(t) = x^T \beta(t) + \epsilon(t),
\]

where \( \beta(t) = (\beta_1(t), \ldots, \beta_K(t))^T \) is a vector of functions, \( x^T \in \mathbb{R}^K \) is a vector, \( y(t) \) is the function output, \( \epsilon(t) \) is the function residuals. In [140], \( \beta(t) \) in the (2.86) was estimated using discrete observations of the ordinary least squares method with

\[
\hat{\beta}' = (x^T x)^{-1} x^T y',
\]

in which \( y' \) present the discrete observations on the function \( y(t) \), and \( \beta' \) can be regarded as the pointwise estimation of \( \beta(t) \). Then, \( \beta(t) \) was constructed using interpolation on its pointwise estimation \( \hat{\beta}' \). Also, two general approaches to tackling model (2.86) have been suggested in [116]: The first one is the pointwise approach which is the same as [140]; The second one is almost the same as that used in [141], the first step of which is representing \( \beta(t) \) and \( y(t) \) using the same group of bases, i.e. \( y = \sum_{j=1}^{J} c_j \psi_j(t) \) and \( \beta_k(t) = \sum_{j=1}^{J} b_{kj} \psi_j(t) \), afterwards, estimating the coefficients \( b_{kj} \) using penalized least squares through minimizing (2.87):

\[
\sum_{i=1}^{N} \sum_{p=1}^{P} [y_i(t_p) - (XB\Psi)_{ip}]^2 + \sum_{k=1}^{K} \lambda_k \int [L(b_k^T \psi)(t)]^2 dt,
\]

where \( X \) is a \( N \times K \) matrix that contains all the observations of \( x \), \( B = (b_1, \ldots, b_K)^T \) with \( b_i = (b_{i1}, \ldots, b_{iJ})^T \) \( (i = 1, 2, \ldots, K) \), \( \Psi = (\psi_j(t_p))_{j=1, \ldots, J, p=1, \ldots, P} \) is a \( J \times P \) matrix, \( L(\cdot) \) is a linear differential operator, \( \lambda_k \) can be selected by leave-one-out cross-validation. Model (2.86) has been investigated in some recent literature and estimated by new approaches. For example, [142] has estimated (2.86) in the RKHS though presenting \( \beta(t) \) utilizing the inner product with the reproducing kernel, and [143] has estimated the (2.86) with the extended LASSO.

Some modifications have been made on (2.86). A typical example can be found in [144], which has investigated the model:

\[
y(t) = x^T \beta(t) + v(t) + \epsilon(t),
\]

2.3.4 Function-on-Scalar Regression
where function outputs \( y(t) = (y_1(t), \ldots, y_N(t))^T \) have \( y_i(t) \) \((i = 1, \ldots, N)\) to be independent stochastic processes, \( x = (x_1^T, \ldots, x_N^T) \) is the input matrix, the variation \( v(t) = (v_1(t), \ldots, v_N(t))^T \) has each \( v_i(t) \) to be the \( i \)-th individual variation, \( \epsilon(t) = (\epsilon_1(t), \ldots, \epsilon_N(t))^T \) has \( \epsilon_i(t) \) to be the \( i \)-th measurement error process. Model (2.88) was regarded as a more general model \( y(t) = f(t) + \epsilon(t) \) with \( f(t) = (f_1(t), \ldots, f_N(t))^T \). Firstly, \( f(t) \) has been rewritten as a p-order polynomial, \( \hat{f}(t) \), using Taylor expansion. Secondly, the parameters of these polynomial were estimated using a weighted least squares, which has the weights provided by a kernel function with bandwidth selected by a generalized cross-validation. Lastly, \( \beta(t) \) was computed by \((x^T x)^{-1} x^T \hat{f}(t)\) and \( \nu(t) \) was calculated by \( \hat{f}(t) - \beta(t) \).

Another modified model of (2.86) was proposed in [145]. It is known as a functional mixed effect model/functional mixed model (FMM). As a blend of fixed and stochastic effect model, this model can be presented as:

\[
y(t) = x\beta(t) + z\alpha(t) + \nu(t),
\]

(2.89)

where \( y(t) = (y_1(t), \ldots, y_N(t))^T \) is a vector of the observed function outputs, \( x = (x_1^T, \ldots, x_N^T) \) and \( z = (z_1^T, \ldots, z_N^T) \) are \( N \times P \) and \( N \times Q \) design matrices, respectively, \( \beta(t) = (\beta_1(t), \ldots, \beta_p(t))^T \) presents the fixed effect, \( \alpha(t) = (\alpha_1(t), \ldots, \alpha_Q(t))^T \) is a vector of random functions presenting the stochastic effect, and \( \nu(t) = (\nu_1(t), \ldots, \nu_N(t))^T \) is a vector of residual functions. The stochastic effect functions \( \alpha(t) \) are assumed to be generated by Gaussian process, and \( \nu(t) \) also follows a Gaussian process. After smoothing \( \beta(t) \) and \( \alpha_i(t) \) using the same group of bases, these two parameters were suggested to be estimated by two methods: one is realized by transforming the model to the traditional multivariate model utilizing the observations, and the other one is the Kalman filter based approach. Model (2.89) was also studied by [146], and the random effect \( \alpha_i(t) \) in (2.89) was assumed to be a mean-zero multivariate Gaussian process. In [146], the model was estimated by a proposed wavelet-based MCMC approach. This approach projects the original model to a wavelet space and applied the MCMC approach to give posterior estimations, and projects it back to the original space afterwards.\(^4\)

Noticing that the approach in [146] may be sensitive to the outliers, a further work, which contains a hierarchical framework and can provide a robust Bayesian approach for (2.89), has therefore been proposed in [147].

\(^4\)The public available code for the method in [146] can be found on http://biostatistics.mdanderson.org/Morris/paper.html.
Multivariate models have been investigated in function-on-scalar regression. Take the multivariate linear model, which has been considered in [143, 148, 149], as an example. The model can be presented as:

\[
y_i(t) = \sum_{k=1}^{K} x_{ik} \beta_k(t) + \epsilon_i(t),
\]

where \(y_i(t)\) \((i = 1, \ldots, N)\) are function outputs, \(x_{ik}\) are scalar inputs, \(\beta_k(t)\) \((k = 1, \ldots, K)\) are functional coefficient parameters, error functions \(\epsilon_i(t)\) \((i = 1, \ldots, N)\) are i.i.d Gaussian processes with zero means. Both sparse and dense (small and large size of the time stamp \(t\)) situations were studied in [143]. Each coefficient \(\beta_k(t)\) was estimated using the penalized least squares, whereas the pointwise estimation has been made for the sparse case and the same estimation procedure has been used on the dense case after the dimensional reduction by FPCA. In [148], B-spline bases were used to represent the outputs and the coefficients, and an adaptive method was proposed to compute the penalized least squares estimator through the ordinary least squares estimator. The research in [149] has extended the result in [148] to the case that the penalty consisting of two parts with one is responsible for variable selection (The same as [143, 148] with the other one controlling the smoothness.).

Another example for multivariate function-on-scalar regression model is the Bayesian model proposed in [150], which can be presented as:

\[
y_i(t) = \sum_{k=1}^{K} f_k(t) \beta_{ik} + \epsilon_i(t),
\]

\[
\beta_{ik} = \mu_k + \sum_{j=1}^{J} \alpha_{jk} x_{ij} + \gamma_{ki},
\]

where \(y_i(t)\) \((i = 1, \ldots, N)\) are \(N\) observed function outputs, \(x_{ij}\) \((i = 1, \ldots, N, j = 1, \ldots, J)\) are scalar inputs, \(\epsilon_i(t)\) \((i = 1, \ldots, N)\) are independent Gaussian processes with zero mean, \(\mu_k, \alpha_{jk}\) and \(\gamma_{ki}\) submit to independent Gaussian distribution with zero means. The model presented in (2.91) can be regarded as the expansion of \(y_i(t)\) using basis \(\{f_k(t)\}_{k=1}^{K}\) and coefficients \(\beta_{ik}\). A corresponding MCMC approach has been proposed to simulate the model, which has the advantage of having no requirement to select any tuning and truncation parameters.

Apart from the aforementioned models, the multilevel regression model is another
kind of model which has been investigated in the previous literature. For instance, the functional multilevel regression model studied by [151] has the form:

$$E[y_{ij}(t)|b_i(t), v_{ij}(t), x_{ij}] = \mu_{ij}(t), \quad (2.93)$$

$$g(\mu_{ij}(t)) = \beta_0(t) + \sum_{k=1}^{p} x_{ijk} \beta_k(t) + b_i(t) + v_{ij}(t), \quad (2.94)$$

where (2.93) presents the conditional expectation of each function output $y_{ij}(t)$ is $\mu_{ij}(t)$ ($i = 1, \ldots, N$, $j = 1, \ldots, J$), $g(\cdot)$ is a known function, $\beta_k(t)$ ($k = 0, 1, \ldots, K$) are coefficient functions, $b_i(t)$ and $v_{ij}(t)$ are two different levels of effects (i.e., subject-level and subject-visit-level, respectively) that can be presented by $b_i(t) = \sum_{k=1}^{K_1} c_{ik}^{(1)} \psi_k^{(1)}(t)$ and $v_{ij}(t) = \sum_{k=1}^{K_2} c_{ijk}^{(2)} \psi_k^{(2)}(t)$, respectively, using FPCA. This model was estimated by a Bayesian approach.

Despite of these models and estimation approaches, more complex function-on-scalar regression models can be found in the modern research, e.g., [152] and [150].

### 2.3.5 Function-on-Function Regression

Function-on-function models have functional inputs and functional outputs. Consider function-on-function regression problems, it needs to learn the operator $F$ that maps $X$ to $Y$, with both $X$ and $Y$ are functional spaces. In this case, both $X$ and $Y$ are usually infinite dimensional spaces. In a similar way to the previous two categories, the simplest and the most frequently used approach is the linear model. Prior to discussing the linear model, the most basic form for function-on-function regression model is in a pointwise (or concurrent) model [116] with the form:

$$y(t) = \alpha(t) + x(t) \beta(t) + \varepsilon(t), \quad (2.95)$$

where $y(t)$ is the function output, $x(t)$ is the function input, $\alpha(t)$ and $\beta(t)$ are the function coefficients, and $\varepsilon(t)$ is residual functions. In [116], the penalized least squares has been used to estimate the coefficients after representing the coefficient functions by cubic splines. A simple estimation approach for (2.95) based on the discrete observations has been proposed in [153], the idea behind which is to estimate the values of the coefficient function on each time stamp. In recent research [154], (2.95) has been extended to $y(s,t) = \alpha(s,t) + \beta(s,t)x(s,t) + \varepsilon(s,t)$, in which function input $x(s,t)$, output $y(s,t)$,
function coefficients \( \alpha(s,t) \) and \( \beta(s,t) \), and function residuals \( \varepsilon(s,t) \) are bivariate functions.

In a different way from (2.95), the most widely recognized and investigated functional linear model has a more complex form than (2.95). The functional linear model can be presented as:

\[
y(t) = \alpha(t) + \int x(s) \beta(s,t) ds + \varepsilon(t),
\]

in which the coefficient function \( \beta(s,t) \) is a bivariate function. Besides, \( y(t) \) is the function output; \( x(s) \) is the function input; \( \alpha(t) \) and \( \varepsilon(t) \) are function coefficient and residuals, respectively. In a similar way to the auto-regressive (AR) time-series model, a special case of the model (2.96), with the input to be the \( n \)-th observed function \( x_n(t) \) and output to be the \( (n+1) \)-th observed function \( x_{n+1}(t) \), is known as a functional auto-regressive (FAR) model, whose properties have been investigated by [126, 155, 156].

The key to identifying the model (2.96) is the determination of the coefficient \( \beta(s,t) \), which can be decomposed into two steps: i) Represent \( \beta(s,t) \) using certain numerical approaches to transfer the infinite dimensional estimation problem to the coefficients estimation problem in \( \mathbb{R}^n \); ii) Seek approaches to identify the coefficients in i).

There are two major approaches (or frameworks), which help to represent \( \beta(s,t) \). These have been summarized in an excellent summary of FDA [116]. The first one is achieved through representing \( \beta(s,t) \) by a tensor product form:

\[
\hat{\beta}(s,t) = \sum_{i=1}^{K_1} \sum_{j=1}^{K_2} b_{i,j} \beta_{1,i}(s) \beta_{2,j}(t) = \beta_1^T(s) B \beta_2(t),
\]

where \( \{\beta_{1,i}(s)\}_{i=1}^{K_1} \) and \( \{\beta_{2,j}(t)\}_{j=1}^{K_2} \) are two groups of bases (e.g. FPCs, splines, Fourier basis) for \( x(s) \) and \( y(t) \), respectively, \( K_1 \) and \( K_2 \) are determined by the K-L expansions of \( x(s) = \sum_{i=1}^{K_1} c_{x,i} \beta_{1,i}(s) \) and \( y(t) = \sum_{j=1}^{K_2} c_{y,j} \beta_{2,j}(t) \). The second one depends on approximating \( \beta(s,t) \) directly using a group of bivariate basis, which can be presented as:

\[
\hat{\beta}(s,t) = \sum_{i=1}^{K} b_{i} \theta_i(s,t),
\]

in which \( \theta_i(s,t) \) are the bivariate basis, e.g. thin-plate basis.
Once $\beta(s,t)$ has been represented, the coefficients applied to represent $\beta(s,t)$ should be estimated. Generally speaking, the most frequently used method for estimating the coefficients in (2.97) and (2.98) is the least squares method, the error function of which to be minimized can be presented as follows:

$$\text{Err} = \sum_{i=1}^{N} \int \|y_i(t) - \hat{y}_i(t)\|^2 + \text{PEN},$$

(2.99)

in which $y_i(t)$ is the $i$-th observed functional output, $\hat{y}_i(t)$ is the estimation of $y_i(t)$ with form $\hat{y}_i(t) = \sum_{j=1}^{K_2} \alpha_j \beta_2, j(t) + \int \hat{\beta}(s,t)x(s)ds$, $\text{PEN}$ is the penalty. Therefore, coefficients $\alpha_j$ and the coefficients for representing $\beta(s,t)$ can be obtained from solving $\min\text{Err}$.

Several examples are provided in order to further explain the above evidence. A classical approach has been provided in [116], which identifies model (2.96) applying the ordinary least squares and the tensor product presentation of $\beta(s,t)$. In [157–159], FPCA basis of $x(s)$ and $y(t)$ have been obtained and used for the identification of $\beta(s,t)$. To be more specific, when $x(s)$ and $y(t)$ satisfy certain conditions, the coefficient $\beta(s,t)$, which minimizes (2.99) with $\text{PEN} = 0$, can be estimated by a tensor product form using the covariance operators of the input and the output. Furthermore, [160] uses a tensor product approach, which is similar to that used in [157–159], to represent $\beta(s,t)$, and to estimate the model using a least squares with a penalty on both directions $s$ and $t$. In [160], the identifiability of the coefficient surface $\beta(s,t)$ has also been discussed. Another literature, [161], has used a bivariate basis spline to approximate $\beta(s,t)$ and estimated the coefficients by least squares with $L_1$ or $L_2$-norm penalty.

Despite those approaches summarized above, some examples, which estimate the coefficient $\beta(s,t)$ of the functional linear model, are presented. A direct estimation of $\beta(s,t)$ has been provided in [162], the estimation requires only the K-L expansion of the input $x(s)$, while using the discrete observations of the output $y(t)$. This approach is derived from ordinary least squares when minimizing (2.99) with $\text{PEN} = 0$. From the functional analysis aspect, the operator $(Tx)(t) = \int x(s)\beta(s,t)ds$ is a linear operator $T : L^2[a,b] \to L^2[c,d]$ and the problem of estimating the kernel $\beta(s,t)$ is actually to estimate the linear operator $T$. In order to propose a general approach to estimating this linear operator, reference [163] has utilized the functional analysis knowledge (e.g. inner product and representation theorem) to give a general bivariate estimator for discrete values of $\beta(s,t)$ computed by $y(t)$ and the conjugate of $x(s)$. A recent approach to
approximate the $\beta(s,t)$ under the RKHS framework has been proposed in [164]. This method relays on the representation theorem and the least squares with PEN is chosen as the integral of the squared curvature of $\beta(s,t)$. The optimality of the finally derived approximator has been proved. Although [165] has investigated the model (2.96) and the coefficient is approximated by a tensor product expansion, the inputs contain random signals and the model was finally estimated by an EM method.

It can be found from the existing references, that there exist some extended linear models based on (2.96). For instance, a function-on-function mixed model (2.100) has been investigated by [166]. This model has the form:

$$y(t) = \alpha(t) + \int x(s)\beta(s,t)ds + u(t) + \varepsilon(t), \quad (2.100)$$

where $u(t)$ and $\varepsilon(t)$ are subject-specific and observation-specific Gaussian process errors, respectively. The coefficient $\beta(s,t)$ in (2.100) was represented by the tensor product form (2.97). Besides, the model was estimated though transforming the model into a discrete form. Another extended model is the multiple functional linear regression (MFLR) model:

$$y_k(t) = \alpha(t) + \sum_{i=1}^{n} \int x_i(s)\beta_{k,i}(s,t)ds + \varepsilon(t), \quad (2.101)$$

in which $y_k(t)$ is the $k$-th element of the multivariate function output $y(t) = (y_1(t), \ldots, y_K(t))^T$, $x_i(s)$ is the $i$-th element of the multivariate function input $x(t) = (x_1(t), \ldots, x_n(t))^T$, $\beta = \{\beta_{k,i}(s,t)\}_{k,i}$ is the function coefficients matrix, $\alpha(t)$ is the intercept function, and $\varepsilon(t)$ is the noise function. Furthermore, (2.101) allows multivariate functional data for both input and output. MFLR with $\alpha(t) = 0$ has been proposed in [167]. This literature has approximated the model through expanding the matrix of the coefficients $\beta = \{\beta_{k,i}(s,t)\}_{k,i}$ using the tensor product shown in (2.97).

A special case of model (2.101) with $K = 1$ has been studied in [168] and [169]. Developing the appropriate basis to approximate $\beta_i(s,t)$ is essential to this model, because complex basis are very likely to bring about a huge computational tasks. In order to relieve the computation burden, a wavelet based signal compression approach\(^5\) has been proposed in [168] and [169]. This signal compression approach represents the model

\(^5\)The R code of the approach discussed in these two papers can be found from http://sites.gsu.edu/rluo/software/.
(2.101) in two steps: 1) Represent the inputs using wavelet basis and rewrite the regression function as \( Y(t) = \alpha(t)I + X\beta(t) + \epsilon(t) \), where \( Y(t) = (y_1(t), y_2(t), \ldots, y_N(t)) \) is the collection of \( N \) elements. 2) Represent the signal part with a best \( K \)-dimensional expansion \( X\beta(t) = \sum_{k=1}^{K} t_k w_k(t) \). In addition, the case when \( \beta(s,t) = (\beta_1(s,t), \ldots, \beta_n(s,t)) \) has sparse structure was considered in [170], and the model was estimated utilizing a sparsity-smooth penalty.

Based on MFLR, a model that studied in [171] allows two different function inputs. The only difference between the model in [171] and (2.101) is that more sources of influence to each observed \( y_i(t) \) e.g., the scalar covariance \( W_i \) and a nonparametric function \( \beta_0(t) \), have been considered in [171]. Then, the model can be presented as:

\[
y_i(t) = W_i + \beta_0(t) + \int x_{i,1}(s)\beta_1(s,t)ds + \int x_{i,2}(s)\beta_2(s,t)ds + \epsilon(t), \tag{2.102}
\]

where \( x_{i,1}(s) \) and \( x_{i,2}(s) \) are the \( i \)-th observed function inputs of \( x_1(s) \) and \( x_2(s) \), respectively. The function coefficients \( \beta_1(s,t) \) and \( \beta_2(s,t) \) have been approximated by the bivariate spline basis, e.g., thin-plate spline basis. The plug-and-play function has been implemented in the \( R \) function \( pfffr(\cdot) \).

Despite these linear models, nonlinear models have also been investigated, even though the research works are minimal when compared with those of linear models. A generalized model:

\[
y(t) = f(x(s)) + \epsilon(t), \tag{2.103}
\]

where \( y(t) \) and \( x(s) \) are function output and function input, respectively; \( f(\cdot) \) is an operator that maps one function space to another; \( \epsilon(t) \) is the function residual. Model (2.103) has been proposed in [172], and it can be either a linear or nonlinear model. The operator \( f(\cdot) \) has been demonstrated can be estimated through taking the advantage of RKHS along with the representation theorem: \( f(x(s)) = \sum_{i=1}^{n} K(\mathcal{F}, x_i)\beta_i(t) \), where \( \mathcal{F} \) is the RKHS, \( K(\cdot,\cdot) \) is the kernel function, and \( \{\beta_i(t)\}_{i} \) forms a group of bases of the domain of \( y(t) \). Gaussian kernel has been used in [172] and the penalty for least squares has been set as \( \lambda \|f(\cdot)\|_{\mathcal{F}}^2 \). A similar model, which is known as the functional single-index model, has been proposed in [173] with \( f(\cdot, \cdot) = f(t, \beta x(t)) \) being a bivariate function. The model in [173] is obtained from extending the single-index model in multivariate case to the function form, and estimated using the weighted least squares (kernel weight) on the Taylor expansion based represented model. A kernel type nonparametric approximator
for \( f(\cdot) \) has been proposed in [174], which enables \( f(\cdot) \) to be estimated by replacing the scalar output \( y_i \) using the function output in (2.84). Furthermore, a bootstrap approach was suggested and proved to be valid by [174].

In a similar way to the development of the linear models, model (2.103) has been extended to a functional additive mixed model [175]:

\[
y_i(t) = \sum_{r=1}^{R} f_r(\mathbf{x}_r, t) + \varepsilon_i(t),
\]

(2.104)
in which \( y_i(t) \) is the \( i \)-th element of the function output vector \( y(t) = (y_1(t), \ldots, y_n(t)) \); \( \mathbf{x}_r \) are the subsets of the complete covariate set \( \mathbf{x} \); each \( f_r(\mathbf{x}_r, t) \) is an operator. [175] has tried to cover a broad region of existing functional regression models (linear or non-linear) with either scalar or function input and output. Furthermore, it also has developed an R function known as `prrf()` which has been embedded in the R package `refund`. In [175], model (2.104) has been represented in a discrete form with the help of the observations:

\[
y_i(t_l) = \sum_{r=1}^{R} f_r(\mathbf{x}_r, t_l) + \varepsilon_i(t_l),
\]

(2.105)

where \( i = 1, \ldots, n \), \( l = 1, \ldots, N \), \( \varepsilon_i(t_l) \) are i.i.d Gaussian residuals. In calculation process, the output observations are listed as an \( nN \times 1 \) vector \( y = (y_1, \ldots, y_n)^T \) with \( y_i = (y_i(t_1), \ldots, y_i(t_N)) \). Afterwards, the model was estimated by a penalized least squares based on the tensor product approximation of \( f_r(\mathbf{x}_r, t) \). To be more specific, each \( f_r(\mathbf{x}_r, t) \) has been represented by:

\[
f_r(\mathbf{x}_r, t) = (\phi_{\mathbf{x}_r} \odot \phi_{t_r}) \theta_r,
\]

(2.106)

where \( \phi_{\mathbf{x}_r} \) (an \( nN \times K_x \) matrix) presents the evaluations of a certain group of bases of the covariate in \( \mathbf{x}_r \), \( \phi_{t_r} \) (an \( nN \times K_t \) matrix) demonstrates evaluations of a certain basis in \( t \), and \( \theta_r \) presents the coefficients; \( \phi_{\mathbf{x}_r} \odot \phi_{t_r} = (\phi_{\mathbf{x}_r} \times 1_{K_x}) \cdot (1_{K_t} \times \phi_{t_r}) \). The penalty has been suggested to be on both margins.

An additive function-on-function model, which is shown in (2.107), has been investigated in [176] and [177]:

\[
y(t) = \int f(x(s), s, t) ds + \varepsilon(t).
\]

(2.107)
Model (2.107) has a trivariate coefficient function \( f(x(s), s, t) \), and it is computationally expensive to estimate using a trivariate spline basis. An computationally efficient approach has been proposed in [176] with a two-step decomposition of \( f(x(s), s, t) \). The first step is to represent \( f(x(s), s, t) \) by \( f(x(s), s, t) = \sum_{i=1}^{I} g_i(x, s) \phi_i(t) \) (the I-truncate of \( f(x(s), s, t) = \sum_{i=1}^{\infty} g_i(x, s) \phi_i(t) \)), which treats the \( x(s) \) and \( s \) as a whole, and to decompose \( f(x(s), s, t) \) in the \( L^2(\Gamma_t) \). The \( \Gamma_t \) is the domain of \( t \) and the \( \phi_i(t) \) are the bases of \( L^2(\Gamma_t) \). The second step comes to represent \( g_i(x, s) \) with a tensor product. The corresponding scalar coefficients are finally estimated through the least squares with penalty \( \lambda \int \int [\partial^2 f(x(s, t))/\partial^2 s]^2 dx ds dt. \) Whereas [177] has proposed a RKHS based method to approximate the same model as [176]. The trivariate function \( f(x, s, t) \in K (K \) is a RKHS) can be presented by its inner product with the RKHS kernel \( k_{x,s,t}(x', s', t') \): 
\[
\langle f(x, s, t), (k_{x,s,t}) \rangle_k.
\]
The penalty of the least squares was taken as \( \lambda \| f \|^2_k \).

Another nonlinear function-on-function model with quadratic and interaction effects has been proposed in [178], with the aim of modelling the unspecified information contained in the data. This model has the following form:

\[
y(t) = \mu(t) + S(t) + \varepsilon(t), \tag{2.108}
\]
\[
S(t) = \sum_i \int x_i(s) \beta_i(s, t) ds + \sum_{i,j} \int \{x_i(u)x_j(v) - E[x_i(u)x_j(v)]\} \gamma_{ij}(u, v, t) dudv, \tag{2.109}
\]

where \( x_1(s), \ldots, x_P(s) \) are \( P \) function inputs; \( \beta_i(s, t) \) is the function coefficient of the main effect; \( \gamma_{ij}(u, v, t) \) is the function coefficient of interaction effect; \( \varepsilon(t) \) is the noise. Both of the linear and quadratic information are contained in this model. [178] has suggested to approximate (2.108) relying on K-L expansion of \( S(t) \), \( \beta(s, t) \) and \( \gamma_{ij}(u, v, t) \).

### 2.3.6 Critical Analysis

Based on the research works for functional regression, the research gaps are analyzed in this section for all those three categories of models and especially for the function-on-function models.

---

6The corresponding R package can be found in http://www4.stat.ncsu.edu/staicu/Code/affpcocode.zip.
2.3.6.1 Research Gaps

Although the existing studies have made a great contribution to the development of functional regression research, there still exist some limitations for these models or regression approaches. These research gaps are summarized in the following three ways:

(1) Most of the early works concentrate on studying the linear models. Given that people can hardly know the real relationship between data, different forms of models, e.g., polynomial [179], quadratic models [178] and the general model \( y = f(x) + \epsilon \) [132–136, 172–175], have therefore been investigated in all three categories (i.e. function-on-scalar, scalar-on-function and function-on-function) of functional regression models. The fixed form models, such as, the linear models, have an assumption — this being that the relationship between the data should submit to a certain relationship. Furthermore, some of their estimation methods even have additional assumptions based on the distribution of the data, evidence of which can be found from the probability based methods, e.g., [124, 129–131, 165]. Therefore, compared with the fixed form models, the model that has a general form \( y = f(x) \) seems to be the most reasonable model for an arbitrary dataset. If \( f(x) \) is a function working on finite dimensional space \( \mathbb{R}^N \), then, many existing universal approximators, e.g., neural networks, fuzzy systems, etc., can be found. However, given \( f(\cdot) \) is a nonlinear operator, which works on functional spaces, then, it is lack of the representation theorem or formulas for learning \( f(\cdot) \). Even though some approaches such as nonparametric methods, RKHS, and basis approximation theory for learning operator \( f(\cdot) \), have been developed to learn the nonlinear operator \( f(\cdot) \), these approaches still suffer from the lack of interpretability.

(2) Existing approaches for functional regression topics have been studied mainly based on classic theory of numerical approximation (e.g., basis functions, FPCA, etc.), statistical approaches (e.g., MLE, MCMC, etc.) and functional analysis (e.g., RKHS). Whereas machine learning or computational intelligence approaches have seldom been applied in order to cope with functional regression problems. Reviewing the existing research, in both areas of classification and clustering, some typical approaches in \( \mathbb{R}^n \) have been generalized to the function space. For instance, the support vector machine (SVM) has been extended to functional data classification in [112]; In [180], the K-means clustering approach has been proposed for functional data; A self-organized map with spline and Gaussian bases has been proposed
for clustering the functional data in [181] and [182], respectively. More examples regarding advanced approaches for functional data clustering can be found in the survey paper [183]. Towards the regression problems, radial basis function networks (RBFN) and multi-layer perceptron (MLP) have been considered in the existing works see [184, 185] for example. To be more specific, the approach proposed in [184] and [185] are actually two new expressions of the RBFN and MLP, but undertake the same job as the classic forms in learning functions from \( \mathbb{R}^n \) to \( \mathbb{R} \), which is completely different from the functional regression topic discussed in this thesis. So far, to the best of our knowledge, the functional data regression approaches, especially the function-on-function type, have seldom been studied using machine intelligence techniques. In real applications, many data are actually functions by nature. When only some data points are used discretely, the whole picture of data is difficult to reflected, and this might lead to a waste of information. Therefore, it is worthwhile to extend the traditional machine learning/computational intelligence approaches in order for them to be able to learn from the functional data.

(3) Almost all the existing models and their identification algorithms are offline approaches. This means that historical data should be provided when using these methods. Once the dataset is huge, it would be difficult to store a large number of historical data. This brings a heavy computational burden to these functional regression approaches, especially to the function-on-function approaches which have more complex forms than the other two categories. As a result, processing big dataset using the existing function-on-function models seems extremely difficult. Furthermore, existing models lack the ability to manage the data streams. On the one hand, the models have a lack of ability to capture the dynamic and complex structures of the data. On the other hand, a vast number of historical data points need to be stored and used in order to train/identify the model, and this may severely influence the computation speed. Even though a time-varying coefficient function-on-function linear regression model has been studied in [186], the model has limited ability to handle the big dataset because it is assumed and estimated relaying on an additional variable and lack of ability to learn the model structure by itself from the data. Therefore, it is desirable to explore and propose functional regression models, which have self-organized structures and can evolve along with the newcoming data, as well as tracking the underlying nonstationary and dynamics of the data.
2.4 Chapter Summary

In this chapter, the existing EFSs and their learning approaches, and functional data regression models and their learning approaches have been separately summarized. We have also given a systematic analysis of the research gaps of the existing works for both topics: EFSs and functional regression. The contributions that have been made in this thesis in order to cope with the discussed research gaps of the existing approaches will be presented in Chapter 3 - Chapter 6.
Chapter 3

Evolving Fuzzy Systems: Optimum Approaches

3.1 Chapter Introduction

This chapter concentrates on solving the local optimization problems of EFSs with either Mamdani or T-S fuzzy systems. Learning approaches, EMFSPO and LEOA for Mamdani and T-S systems, respectively, are proposed. This is in order to seek effective EFSs learning methods that can avoid (or at least reduce) the following shortcomings of the existing approaches: (1) Developed based on a heuristic way, rather than an optimal approach; (2) Lack of optimality of the consequent parameters when there is a structure update of the fuzzy system. (See the weaknesses summarized in (1) and (2) in Section 2.2.4.)

From an optimization point of view, the ultimate purpose is to develop the approaches, which can minimize a series of local error functions considering any changes brought on by the change of parameters and structure of a Mamdani or a T-S fuzzy system. We derive five learning modules for either EMFSPO or LEOA. These learning modules are the structure learning (i.e., rule generation, rule pruning and rule merging) and the parameters learning (i.e., antecedent and consequent parameters updating) which we have presented in Section 2.2.3.1, in order to answer the question about how the structure and parameters of the EFSs can be identified online. To be more specific, the learning method for each module can be summarised as follows: (1) The rule base is expanded relying on
the firing strength of new data in every existing cluster; (2) The rule merging process is triggered when two clusters have similar centres and radii; (3) Redundant fuzzy rules are removed according to the cumulative firing strength and age; (4) The antecedent parameters are learned using the recursive formulas induced from the sample mean and radius; (5) Consequent parameters are updated through EWRLS. Although some of the evolving approaches considering (1)–(4) have been used in the existing works, we give further support to the optimality of these methods. Furthermore, mathematical proofs and calculations are provided in order to verify the local optimality of the proposed EMFSPO and LEOA. They are tested on several benchmark examples and real-world datasets and compared with existing state-of-the-art methods. Additionally, LEOA is found to have an additional property which is the ability to remember the global behavior of the data streams, the evidence can be found from the numerical examples in Section 3.3.4.

The rest of this chapter is arranged as follows: Section 3.2 presents the main approach and theory for EMFSPO. Section 3.3 presents the main theory and framework of LEOA. The conclusions of the proposed two approaches are provided in Section 3.4.

### 3.2 Learning Evolving Mamdani Fuzzy Systems Based on A Local Online Optimization Approach

In this section, the local optimal approach for learning the EFS with Mamdani rules is proposed. This section is organized as follows: Section 3.2.1 introduces Mamdani fuzzy systems and their learning problem; Section 3.2.2 provides EMFSPO structure and parameters learning methods, and a detailed explanation of how to get the optimal solution of the local error functions based on these updating methods. The detailed learning steps for EMFSPO are given in Section 3.2.3. Thereafter, Section 3.2.4 evaluates EMFSPO by two benchmark examples.

#### 3.2.1 Problem Statement

A Mamdani fuzzy system has its $i$-th fuzzy rule $R_i$ in the following form:

$$R_i: \text{If } x_1 \text{ is } \Gamma_{i,1} \text{ and } x_2 \text{ is } \Gamma_{i,2} \text{ and } \ldots \text{ and } x_n \text{ is } \Gamma_{i,n}, \text{ then } y_i \text{ is } \tilde{\Psi}_i,$$

(3.1)

---

1Sections 3.2 and 3.3 are based on our published work [187] and [188], respectively.
where \( i = 1, 2, \ldots, K \), \( K \) is the number of fuzzy rules, \( \mathbf{x} = (x_1, x_2, \ldots, x_n) \) is the input, \( y_i \) is the output of rule \( R_i \), \( \tilde{\Psi}_i, \Gamma_{i,j}, j = 0, 2, \ldots, n \), are fuzzy sets, and \( n \) is the dimension of the input.

The membership functions are of the Gaussian type. With the cluster centre of \( \Gamma_{i,j} \) is \( c_{i,j} \) and the radius is \( \sigma_{i,j} \), the membership degree of \( x_j \) in \( \Gamma_{i,j} \) is \( \mu_{i,j}(x_j) \) as presented by:

\[
\mu_{i,j}(x_j) = \exp\left\{ -\frac{(x_j - c_{i,j})^2}{2(\sigma_{i,j})^2} \right\}.
\]

The firing strength \( \gamma_i(x) \) of the rule \( R_i \) and the final output \( y \) of the model can be calculated by (3.3) and (3.4), separately:

\[
\gamma_i(x) = \prod_{j=1}^{n} \mu_{i,j}(x_j),
\]

\[
y = \sum_{i=1}^{K} \theta_i(x) \psi_i,
\]

where \( \theta_i(x) = \frac{\gamma_i(x)}{\sum_{j=1}^{K} \gamma_j(x)} \) is the normalized firing strength of rule \( R_i \), \( \psi_i \) is the centre of the fuzzy set \( \tilde{\Psi}_i \). \( \theta_i(x) \), \( \gamma_i(x) \) and \( \mu_{i,j}(x_j) \) \( (i = 1, 2, \ldots, K \) and \( j = 1, 2, \ldots, n \) \) are functions of \( x \).

The problem which needs to be solved by EMFSPO is to dynamically identify the structure of the Mamdani-type EFS (i.e. the fuzzy rule number \( K \)) and the parameters (i.e. \( c_{i,j}, \sigma_{i,j} \) and \( \psi_i \)) in order to minimize the local error functions shown in (3.5):

\[
Err_i = \sum_{k=1}^{t+1} \theta_i^{(t+1)}(x_k)(y_k - \psi_i)^2,
\]

in which \( \theta_i^{(t+1)}(x_k) \) \( (i = 1, 2, \ldots, K^{(t+1)} \) are the normalized firing strengths at time \( t + 1 \), \( (x_k, y_k) \) is the \( k \)-th input-output data pair, \( x_k = (x_{k,1}, x_{k,2}, \ldots, x_{k,n}) \), \( k = 1, 2, \ldots, t + 1 \). The reasons for using local error function (3.5) to learn the EFSs are as follows: Firstly, it is a widely used learning criteria [27, 28, 30, 58] that enables flexible rule base changing [189]; Secondly, such a criterion can lead to more robust fuzzy systems being identified compared with the global version, because highly noisy data in one region will not have much impact on the quality and accuracy of the identified model in another region. The detailed EMFSPO structure and parameters learning steps which are induced
from minimizing local error functions given in (3.5), are shown in section 3.2.2.

### 3.2.2 EMFSPO Online Learning Method

EMFSPO is developed from an optimization point of view. To be more specific, structure changing criteria and parameters updating methods are built in order to minimize (3.5). Based on [190], by computing the derivatives of the local error functions $Err_i$ (3.5) with regard to $\psi_i$, the estimation of $\psi_i$ can be calculated by WRLS induced from (3.6):

$$\psi_i = \{\overline{R}(t+1)\}^{-1}f^i(t+1),$$

(3.6)

where $\overline{R}(t+1) = \sum_{k=1}^{t+1} \theta_i^{(t+1)}(x_k)$ and $f^i(t+1) = \sum_{k=1}^{t+1} \theta_i^{(t+1)}(x_k)y_k$. This formula can lead to the local optimality when a fuzzy system has a fixed structure. However, the structure change of the system and the parameters of each fuzzy rule would influence how to get the WRLS solution of $\psi_i$. In the following part of this section, we will propose the new least squares estimation of $\psi_i$ based on (3.6) considering the structure and parameters changes of the EFS. In the remaining parts of this section, we utilize the point of view of minimizing (3.6) in finding out the appropriate approaches for structure learning and parameter learning. The technique details are summarized in the Theorems and Conditions in the following subsections. Note that in the following parts of this section especially every Theorem, we assume $\{x_k\}_{k=1}^\infty$ and $\{y_k\}_{k=1}^\infty$ are bounded.

#### 3.2.2.1 Structure Learning

Following from the general framework of those EFSs learning approaches introduced in Section 2.2.3.1, EMFSPO contains three structure evolving modules: rule generation, pruning and merging. All of these are induced and proposed from an optimization viewpoint.

##### 3.2.2.1.1 Fuzzy Rule Generation

Assume that there is a fuzzy rule generated at time $t+1$ with cluster centre $c_{K(t+1)}^{(t+1)} = x_{t+1}$, and from time $k = 1$ to $k = t$, the fuzzy system kept unchanged with respect to both the rule number and parameters. Furthermore, suppose that the number of fuzzy rules at time $k = 1, \ldots, t$ are $K^{(1)} = K^{(2)} = \ldots = K^{(t)}$, and $K^{(t+1)} = K^{(t)} + 1$. Therefore, at time
CHAPTER 3. EVOLVING FUZZY SYSTEMS: OPTIMUM APPROACHES

\[ t + 1, \text{ for } i = 1, 2, \ldots, K^{(t)}, \text{ Err}_t \text{ can be represented by (3.7):} \]

\[ \text{Err}_t = \sum_{k=1}^{t} \theta_i^{(t+1)}(x_k)(y_k - \psi_i)^2 + \theta_i^{(t+1)}(x_{t+1})(y_{t+1} - \psi_i)^2, \tag{3.7} \]

where

\[ \theta_i^{(t+1)}(x_k) = \frac{\gamma_i^{(t+1)}(x_k)}{\sum_{l=1}^{K^{(t+1)}} \gamma_l^{(t+1)}(x_k)} = \frac{\gamma_i^{(t+1)}(x_k)}{\sum_{l=1}^{K^{(t+1)-1}} \gamma_l^{(t+1)}(x_k) + \gamma_i^{(t+1)}(x_k)}. \tag{3.8} \]

So \( \overline{R}(t + 1) \) and \( f^i(t + 1) \), which used to estimate \( \psi_i(t + 1) \), can be written as:

\[ \overline{R}(t + 1) = \overline{R}(t) + \sum_{k=1}^{t} A_{r,k} + \theta_i^{(t+1)}(x_{t+1}), \tag{3.9} \]

\[ f^i(t + 1) = f^i(t) + \sum_{k=1}^{t} A_{r,k} + \theta_i^{(t+1)}(x_{t+1})y_{t+1}, \tag{3.10} \]

where

\[ A_{r,k} = \frac{\gamma_i^{(t+1)}(x_k)(-\gamma_i^{(t+1)}(x_k))}{(\sum_{l=1}^{K^{(t+1)}} \gamma_l^{(t+1)}(x_k))(\sum_{l=1}^{K^{(t)}} \gamma_l^{(t+1)}(x_k))}. \tag{3.11} \]

Applying (3.6), (3.9), (3.10) and (3.11), we can obtain the fuzzy rule generation method in Condition 3.1 and the consequent parameters estimation approach in 3.1.

**Condition 3.1.** If \( \gamma_i^{(t)}(x_{t+1}) < \varepsilon^{(a)} (i = 1, 2, \ldots, K^{(t)}, \varepsilon^{(a)} = \exp\{-k_0^2/2\} > 0 \) meets, and a new fuzzy rule (the \( R_{t+1} \)-th rule) is generated with cluster centre \( c_{k^{(t+1)}} = x_{t+1} \) and radius \( \sigma_{k^{(t+1)}} \):

\[ \sigma_{k^{(t+1)}, j}^2 = \min\left\{ \frac{n \min\{ (c_{i,j} - k_0 \sigma_{i,j} - c_{k^{(t+1)}, j})^2, (c_{i,j} + k_0 \sigma_{i,j} - c_{k^{(t+1)}, j})^2 \}}{-2 \log \frac{\varepsilon^a}{t}} \right\}, \tag{3.12} \]

where \( j = 1, 2, \ldots, n \), and \( \varepsilon_0 > 0 \) is the tolerance degree.

**Theorem 3.1.** If Condition 3.1 holds, then consequent parameters \( \psi_i^{(t+1)} (i = 1, 2, \ldots, K^{(t+1)} - 1) \) which minimize (3.7) can be obtained recursively by (3.13):

\[ \psi_{t}(t + 1) = \psi_{t}(t) + \overline{R}(t + 1)^{-1} \theta_i^{(t+1)}(x_{t+1}) \{y_{t+1} - \psi_{t}(t)\}. \tag{3.13} \]

**Proof.** The influence of the structure change shown in \( A_{r,k} \) (3.11) can be neglected, when
Condition 3.1 holds. The WRLS recursive formula can be used to compute $\psi_i$. Detailed proof can be found in Appendix A.1.1.

On the one hand, Theorem 3.1 induced (3.13) utilizing the fact that the firing strength $\gamma_i(t) (x_{t+1}) < \varepsilon(a)$ can lead to the $A_{t,k}$ in (3.11) converging to 0; On the other hand, adding fuzzy rules which are reliant on the firing strength can help to obtain the most simple updating formulas of the consequent parameters.

### 3.2.2.1.2 Fuzzy Rule Pruning

Assume that the $i^{th}$ fuzzy rule $R_i^{(t)}$ is pruned at time $t+1$, and that $K^{(1)} = K^{(2)} = \ldots = K^{(t)}$ and $K^{(t+1)} = K^{(t)} - 1$ are satisfied. Furthermore, we assume $k = 1, 2, \ldots, t$. Then, $R_i^{(t+1)}$ and $f_i^{(t+1)}$ are given as follows:

$$R_i^{(t+1)} = R_i^{(t)} + \sum_{k=1}^{t} B_{t,k} + \theta_i^{(t+1)} (x_{t+1}), \quad (3.14)$$

$$f_i^{(t+1)} = f_i^{(t)} + \sum_{k=1}^{t} B_{t,k} y_k + \theta_i^{(t+1)} (x_{t+1}) y_{t+1}, \quad (3.15)$$

where

$$B_{t,k} = \frac{\gamma_i^{(t+1)} (x_k) \gamma_i^{(t)} (x_k)}{\left( \sum_{l=1}^{K^{(t)}} \gamma_i^{(t)} (x_l) \right) \left( \sum_{l=1, l \neq i}^{K^{(t)}} \gamma_i^{(t+1)} (x_l) \right)}.$$  \hspace{1cm} (3.16)

With these formulas, the fuzzy rule pruning method is shown in Condition 3.2, and the consequent parameters updating approach when rule pruning happens is summarized in Theorem 3.2.

**Condition 3.2.** If the age satisfies $t + 1 - t^* > M$ and $\mathcal{N}_i^{(t+1)} = \sum_{l=t^*+1}^{t+1} \gamma_i^{(l)} (x_l) < \varepsilon_p$ hold, rule $R_i^{(t)}$ should be pruned. The parameter $M > 0$, $\varepsilon_p > 0$, $t^*$ is the time when the $i^{th}$ fuzzy rule is built, and $\mathcal{N}_i^{(t+1)}$ can be calculated recursively by $\mathcal{N}_i^{(t+1)} = \mathcal{N}_i^{(t)} + \gamma_i^{(t+1)} (x_{t+1})$, $\mathcal{N}_i^{(t^*)} = 0$, $i^* = 1, 2, \ldots, K^{(t)}$.

**Theorem 3.2.** If Condition 3.2 holds, then consequent parameters, $\psi_i(t+1)$, ($i = 1, 2, \ldots, K^{(t+1)}$), which minimize the local error functions (3.7), can be obtained recursively by (3.13).

**Proof.** See Appendix A.1.2.
Similar to Theorem 3.1, the fuzzy rule pruning criterion has been selected such that \( B_{l,k} \) in (3.16) converges to 0.

### 3.2.2.1.3 Fuzzy Rule Merging

In a similar way to Section 3.2.2.1.1 and 3.2.2.1.2, we assume that rule \( R_{l_1} \) and \( R_{l_2} \) are merged at time \( t + 1 \), and \( K^{(0)} = K^{(2)} = \ldots = K^{(n)} \). The final fuzzy rule merged by \( R_{l_1} \) and \( R_{l_2} \) is \( R_{l_0} \), where \( l_0 = l_1 \) numerically. \( R^{(t)}(t + 1) \) and \( f^{(t)}(t + 1) \) are as follows:

\[
R^{(t)}(t + 1) = R^{(t)}(t) + \sum_{k=1}^{t} C_{t,k} + \theta^{(t+1)}_{j}(x_{t+1}),
\]

\[
f^{(t)}(t + 1) = f^{(t)}(t) + \sum_{k=1}^{t} C_{t,k}y_k + \theta^{(t+1)}_{j}(x_{t+1})y_{t+1}.
\]

where \( C_{t,k} \) is shown in (3.19),

\[
C_{t,k} = \frac{\gamma^{(t)}_l(x_k)[\gamma^{(t)}_{l_1}(x_k) + \gamma^{(t)}_{l_2}(x_k) - \gamma^{(t)}_0(x_k)]}{[\sum_{i=1}^{K^{(t+1)}} \gamma^{(t+1)}_i(x_k)][\sum_{i=1}^{K^{(t)}} \gamma^{(t)}_i(x_k)]}.
\]

From (3.17), (3.18) and (3.19), the fuzzy rule merging method can be shown in Condition 3.3, and the consequent parameters updating formula when rule merging happens is presented in Theorem 3.3.

**Condition 3.3.** If for rule \( R_{l_1} \) and \( R_{l_2} \), and two predefined “large enough” real number \( \varepsilon_c, \varepsilon_\sigma \) > 0 \(^\text{2}\) such that \( \| c_{l_1} - c_{l_2} \| < \varepsilon_c \) and \( \| \sigma_{l_1}^2 - \sigma_{l_2}^2 \| < \varepsilon_\sigma \) at time \( t + 1 \), then \( R_{l_1} \) and \( R_{l_2} \) are merged to \( R_{l_0} \), and the centre and radius of the \( R_{l_0} \) are calculated by (3.20) and (3.21), respectively:

\[
c_{l_0} = \frac{N_{l_1}c_{l_1} + N_{l_2}c_{l_2}}{N_{l_1} + N_{l_2}},
\]

\[
(\sigma_{l_0,j})^2 = \frac{(N_{l_1} - 1)(\sigma_{l_1,j})^2 + (N_{l_2} - 1)(\sigma_{l_2,j})^2 + \frac{N_{l_1}N_{l_2}(c_{l_1,j} - c_{l_2,j})^2}{N_{l_1} + N_{l_2} - 1}}{N_{l_1} + N_{l_2} - 1}.
\]

**Theorem 3.3.** If Condition 3.3 holds, consequent parameters, \( \psi_i(t+1) \) \( (i = 1, 2, \ldots, K^{(t+1)}) \) which minimize the local error functions (3.7) can be updated by (3.22):

\[
\psi_i(t + 1) = \bar{\psi}_i^{(l_2)}(t) + R_i^{(t)}(t)^{-1} \theta^{(t+1)}_{j}(x_{t+1}) \{ y_{t+1} - \bar{\psi}_i^{(l_2)}(t) \},
\]

\(^\text{2}\)The thresholds \( \varepsilon_c \) and \( \varepsilon_\sigma \) present the judgement about similarity between two fuzzy rules. They should be “large enough” small numbers, and predefined based on the data.
where \( \widetilde{\psi}_i^{(l_2)}(t) \) and \( \widetilde{p}_i^{(l_2)}(t) = \widetilde{R}_i^{l_2}(t)^{-1} \) are shown in (3.23),

\[
\widetilde{\psi}_i^{(l_2)}(t) = \psi_i^{(l_2)}(t) + \frac{\widetilde{p}_i^{(l_2)}(t)}{(1 + \theta_i^{(t+1)}(x_t+1)p_i^{(l_2)}(t-1))^2}, \quad \text{with} \quad \widetilde{R}_i^{l_2}(t)^{-1} = \sum_{k=1}^{l_2} \{ \theta_i^{(t)}(x_k) + \frac{\gamma_i^{(t)}(x_k)}{[\sum_{i=1}^{l_2} \gamma_i^{(t)}(x_k)]} \}, \quad l_2 = l_2.
\]

\[\text{Proof.} \quad \text{See Appendix A.1.3.} \]

One can observe from (3.19) that \( C_{t,k} \) can not converge to 0 through setting the similarity measure. Therefore, the least square formulas for \( \psi_i \) would have a more complicated form than those cases which we have discussed in the Section 3.2.2.1.1 and 3.2.2.1.2.

### 3.2.2.2 Parameters Learning

This section presents the methods for EMFSPO in order to learn its antecedent, and summarizes the consequent parameters online updating method.

#### 3.2.2.2.1 Antecedent Parameters Learning

It is easy to check that WRRLS induced from (3.13) can be used to calculate \( \psi_i \) when updating existing cluster centres and radii based on \textit{Condition 3.4}.

**Condition 3.4.** Assume that \( \gamma_i^{(t)} \geq \exp\{-k_0^2/2\} \) and \( i^* = \arg \max_i \gamma_i^{(t)}(x_{t+1}) \), the cluster centre \( c_{i^*} \) and radius \( \sigma_{i^*} \) \( \{c_{i^*,1}, c_{i^*,2}, \ldots, c_{i^*,n}\} \) and \( \sigma_{i^*} = \{\sigma_{i^*,1}, \sigma_{i^*,2}, \ldots, \sigma_{i^*,n}\} \) are updated by (3.25) and (3.26):

\[
c_{i^*, j}(t + 1) = c_{i^*, j}(t) + \frac{x_{t+1,j} - c_{i^*, j}(t)}{N_{i^*}(t)+1}, \tag{3.25}
\]

\[
(\sigma_{i^*, j}(t + 1))^2 = (\sigma_{i^*, j}(t))^2 + \frac{(x_{t+1,j} - c_{i^*, j}(t))^2 - (\sigma_{i^*, j}(t))^2}{N_{i^*}(t)+1}, \tag{3.26}
\]

where \( N_{i^*}(t) \) is the total number of data points \( x \) which satisfies \( \gamma_i^{(t)}(x) \geq \exp\{-k_0^2/2\} \) by time \( t \).
The centres and radii updating formulas shown in (3.25) and (3.26) are the formulas which follow from the Kohonen rule (see (2.43) and (2.44)).

### 3.2.2.2 Consequent Parameters Learning

Based on the consequent parameters learning formulas that have induced in Theorem 3.1, Theorem 3.2, Theorem 3.3 and Section 3.2.2.2.1, we summarize these formulas using a uniform format known as EWRLS. EWRLS works as follows:

(I) When no fuzzy rules are merged, (3.27) and (3.28) should be applied to update the consequent parameters. These formulas can be induced following the WRLS derivation procedure in [190].

\[
\psi_i(t + 1) = \psi_i(t) + \theta_i^{(t+1)}(x_{t+1})P_i(t + 1)(y_{t+1} - \psi_i(t)), \tag{3.27}
\]

\[
P_i(t + 1) = \frac{P_i(t)}{1 + \theta_i^{(t+1)}(x_{t+1})P_i(t)}, \tag{3.28}
\]

(II) Otherwise, assuming that fuzzy rules \(R_i^q\) and \(R_i^p\) are merged to \(R_i^0\), the consequent parameters should be updated by (3.29) and (3.30):

\[
\psi_i(t + 1) = \tilde{\psi}_i^{(lp)}(t) + \theta_i^{(t+1)}(x_{t+1})P_i(t + 1)(y_{t+1} - \tilde{\psi}_i^{(lp)}(t)), \tag{3.29}
\]

\[
P_i(t + 1) = \frac{\tilde{P}_i^{(lp)}(t)}{1 + \theta_i^{(t+1)}(x_{t+1})\tilde{P}_i^{(lp)}(t)}, \tag{3.30}
\]

in which \(\tilde{\psi}_i^{(lp)}(t)\) and \(\tilde{P}_i^{(lp)}(t)\) are updated by (3.23) and (3.24). In each step of the consequent parameters learning, extra information, i.e., \(\tilde{\psi}_i^{(l)}(t)\) and \(\tilde{P}_i^{(l)}(t)\), where \(i, l = 1, 2, \ldots, K, l \neq i\), should be recorded.

### 3.2.3 Mechanism of EMFSPO

Detailed steps of the EMFSPO learning algorithm are presented in this section, by compositing the structure and parameters learning method of EMFSPO proposed in Section 3.2.2. Based on the existing research and common sense in EFS online learning approaches, we suggest setting the thresholds (control parameters) \(\varepsilon_0, \varepsilon_c, \varepsilon_\sigma\) and \(\varepsilon^{(p)}\) to small values, set \(k_0\) with \(k_0 > 1\), and set \(M\) to be a relatively large number.

step 1 Read new input \(x_{t+1}\), real output \(y_{t+1}\), and initialize rule merging indicator \(ind_m =\)
0 and generation indicator \( \text{ind}_a = 0 \).

step 2 Rule generation and updating (see Theorem 3.1)

If \( \gamma_t^{(l)}(x_{t+1}) < \exp\{-k_0^2/2\} \) holds, then a new rule generated with \( K^{(t+1)} = K^{(t)} + 1, \text{ind}_a = 1, c_{K^{(t+1)}} = x_{t+1}, \sigma_{K^{(t+1)}} \) (see Theorem 3.1). Otherwise, update \( c_i^*, \sigma_i \) by (3.25) and (3.26).

step 3 Rule pruning (see Theorem 3.2)

If \( t_{\text{max}} = \max(t + 1 - t_i^* ) > M, i^* = \arg \max_i (t + 1 - t_i^*) \) and \( \sum_{k=t_i^*}^{t+1} \gamma_t^{(k)}(x_k) < \epsilon^{(p)} \) hold, where \( t_i^* \) is the time when rule \( R_i \) is built, then remove the rule \( R_i \).

step 4 Rule merging (see theorem 3.3)

If \( \text{ind}_a = 0, \| c_{l_q}(t + 1) - c_{l_p}(t + 1) \| < \epsilon_c \) and \( \| \sigma_{l_q}^2(t + 1) - \sigma_{l_p}^2(t + 1) \| < \epsilon_{\sigma} \) hold, then merge rule \( R_{l_q} \) and \( R_{l_p} \) to \( R_{l_0} \) with \( c_{l_0} \) and \( \sigma_{l_0} \) are computed by (3.20) and (3.21), and set \( \text{ind}_m = 1 \).

step 5 Output

Compute \( \hat{y}_{t+1}: \hat{y}_{t+1} = \sum_{i=1}^{K^{(t+1)}} \theta_i^{(t+1)}(x_{t+1}) \psi_i^{(t)} \).

step 6 EWRLS

If \( \text{ind}_m = 0 \) holds, then update \( \psi_i^{(l)}(i = 1, 2, \ldots, K^{(t+1)}) \) by (3.27) and (3.28), otherwise apply (3.29) and (3.30). For each \( i = 1, 2, \ldots, K^{(t+1)} \) and \( l = 1, 2, \ldots, K^{(t+1)} \) with \( l \neq i \), update \( \tilde{\psi}_i^{(l)} \) and \( \tilde{P}_i^{(l)} \) using (3.23) and (3.24).

The pseudo code of EMFSPO is shown in Algorithm 1.

### 3.2.4 Numerical Examples

Two widely used benchmark examples of nonlinear and high-dimensional system identification are applied in this section in order to demonstrate that EMFSPO has the ability to make accurate predictions. The accuracy is judged by root mean square error (RMSE) (see (B.1) in Appendix B.1).\(^3\)

\(^3\)Results of EMFSPO are calculated in the environment of intel(R) core(TM) i7-4790 CPU with a 3.6 GHz processor and a 16.0 GB memory.
CHAPTER 3. EVOLVING FUZZY SYSTEMS: OPTIMUM APPROACHES

Algorithm 1: EMFSPO

\begin{verbatim}
Input: \( x_{t+1} = \{x_{t+1,1}, x_{t+1,2}, \ldots, x_{t+1,n} \} \)
Output: \( \hat{y}_{t+1} \)
initialize: \( ind_m = 0, ind_a = 0 \);
if \( \gamma_i(t)(x_{t+1}) < \exp\{-k_0^2/2\} \) then
    \( K^{(t+1)} = K^{(t)} + 1, ind_a = 1, c_{K^{(t+1)}} = x_{t+1}, \sigma_{K^{(t+1)}} \) is shown in Theorem 3.1;
else
    \( K^{(t+1)} = K^{(t)} \), get \( c_i(t+1), \sigma_i(t+1) \) by (3.25);
end
for \( i = 1, 2, \ldots, K^{(t+1)} \) do
    \( \mathcal{M}_i^{(t+1)} = \mathcal{M}_i^{(t)} + \gamma_i^{(t+1)}(x_{t+1}) \);
end
if \( t_{\text{max}} = \max(t + 1 - t_i^*) > N \) and \( \mathcal{M}_i^{(t+1)} < \epsilon^{(p)} \) then
    remove rule \( R_{t_i} \), \( i^* = \arg \max_i (t + 1 - t_i^*) \);
end
if \( ind_a = 0, \|c_{q_i}(t+1) - c_{q}(t+1)\| < \epsilon_q, \) and \( \|\sigma_{c_i}^2(t+1) - \sigma_{c_i}^2(t+1)\| < \epsilon_{\sigma} \) then
    merge \( R_{t_q} \) and \( R_{t_p} \) with \( c_{q_i}, \sigma_{q_i} \) computed by (3.20) and (3.21), and set \( ind_m = 1 \);
end
calculate the model output, \( \hat{y}_{t+1} \), by (3.4);
if \( ind_m = 0 \) then
    get \( \psi_i^{(l)}(t+1) \) and \( P_i^{(l)}(t+1) \) by (3.27) and (3.28);
else
    get \( \psi_i^{(l)}(t+1) \) and \( P_i^{(l)}(t+1) \) by (3.29) and (3.30);
end
update \( \bar{\psi}_i^{(l)}(t+1) \) and \( \bar{P}_i^{(l)}(t+1) \) by (3.23) and (3.24).
\end{verbatim}

3.2.4.1 Example 1: Nonlinear Dynamic Plant (term 4 in the Appendix B.1)

The mathematical model of the nonlinear dynamic system to be identified in this example can be found from (B.9) in Appendix B.1. The experiment setting is exactly the same as that stated in Appendix B.1. The thresholds of EMFSPO have been set as: \( \epsilon_0 = 10^{-5}, k_0 = 1.2, M = 30, \epsilon_{(p)} = \epsilon^{-2}, \epsilon_{c} = 10^{-5}, \epsilon_{\sigma} = 10^{-3} \). The result of EMFSPO has been compared with eTS [27], simulTeTS [95], SAFIS [94], SOFMLS [47], ZDMEKF based FIS [70], as well as OSAMNN [84]. Table 3.1 summaries the numerical results, and indicates that EMFSPO has the ability to produce better prediction results compared with previous works in terms of accuracy. Furthermore, it can be seen from Table 3.1, the rule number that used by EMFSPO is not the minimum. One important reason is the simple structure of Mamdani rule usually need more fuzzy rules to achieve high accuracy. Figure 3.1 shows the absolute online prediction errors of EMFSPO in
the online learning phase. It illustrates that the absolute prediction error has decreased rapidly in the first 1000 learning steps and has reached a level below 0.1, which further demonstrates that EMFSPO is an effective online EFS identification algorithm.

EMFSPO is an interpretable model, the updated model on each learning step can be presented by Mamdani fuzzy rules. The final model learned by EMFSPO contains 26 rules, which is a large number; Therefore, first 5 of these rules are picked to show the interpretability of EMFSPO. These rules are listed below:

\[ R_1 : \text{If } x_1 \text{ is } \Gamma_{1,1}, \text{ and } x_2 \text{ is } \Gamma_{1,2}, \text{ and } x_3 \text{ is } \Gamma_{1,3}, \text{ Then } y_1 = -0.3504; \]
\[ R_2 : \text{If } x_1 \text{ is } \Gamma_{2,1}, \text{ and } x_2 \text{ is } \Gamma_{2,2}, \text{ and } x_3 \text{ is } \Gamma_{2,3}, \text{ Then } y_2 = 1.0391; \]
\[ R_3 : \text{If } x_1 \text{ is } \Gamma_{3,1}, \text{ and } x_2 \text{ is } \Gamma_{3,2}, \text{ and } x_3 \text{ is } \Gamma_{3,3}, \text{ Then } y_3 = 1.1733; \]
CHAPTER 3. EVOLVING FUZZY SYSTEMS: OPTIMUM APPROACHES

\[ R_4 : \text{If } x_1 \text{ is } \Gamma_{4,1}, \text{ and } x_2 \text{ is } \Gamma_{4,2}, \text{ and } x_3 \text{ is } \Gamma_{4,3}, \text{ Then } y_4 = 1.2195; \quad (3.34) \]

\[ R_5 : \text{If } x_1 \text{ is } \Gamma_{5,1}, \text{ and } x_2 \text{ is } \Gamma_{5,2}, \text{ and } x_3 \text{ is } \Gamma_{5,3}, \text{ Then } y_5 = 0.4502. \quad (3.35) \]

The centres and radii of \( R_1, R_2, \ldots, \) and \( R_5 \) are:

\[ c_1 = [-0.1386, 0.1165, -0.3629] \text{ and } \sigma_1 = [0.0020, 0.0015, 0.0038]; \quad (3.36) \]
\[ c_2 = [0.8797, 0.6829, 0.9511] \text{ and } \sigma_2 = [0.0016, 0.0005, 0.0001]; \quad (3.37) \]
\[ c_3 = [1.0531, 0.8800, 0.9980] \text{ and } \sigma_3 = 10^{-5}[0.1615, 0.9252, 0.0364]; \quad (3.38) \]
\[ c_4 = [1.1758, 1.0531, 0.9823] \text{ and } \sigma_4 = 10^{-5}[0.4925, 0.6246, 0.0004]; \quad (3.39) \]
\[ c_5 = [0.7697, 1.0196, 0.3681] \text{ and } \sigma_5 = 10^{-6}[0.1358, 0.2250, 0.4294]. \quad (3.40) \]

3.2.4.2 Example 2: High-dimensional System Identification Problem (term 3 in the Appendix B.1)

This example is used to test EMFSPO on identifying a high-dimensional system, which has the dataset generated from model (B.7) in Appendix B.1. In the same way as Example 1, this example is set to follow the instructions in term 3 in Appendix B.1. Fig. 3.2 shows the prediction results of EMFSPO on the last 300 data points. The obtained RMSEs are displayed in Table 3.2 (all other results can be found from [83]). It is obvious that EMFSPO has obtained smaller RMSE. Based on Table 3.2, compared with the second best method, eNFN [83], it seems that EMFSPO has improved significantly in both accuracy and rule numbers. However, eNFN has used triangle membership functions. Moreover, at each calculation step, there have only at the most, only two fuzzy rules have been activated and used. A tiny number of rules have been activated compared with EMFSPO. Therefore, in this sense, EMFSPO has not achieved a great deal.

<table>
<thead>
<tr>
<th>Method</th>
<th>Rule Type</th>
<th>RMSEs</th>
<th>Rule Num.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DENFIS [26]</td>
<td>T-S</td>
<td>0.2080</td>
<td>13</td>
</tr>
<tr>
<td>eMG [54]</td>
<td>T-S</td>
<td>0.1244</td>
<td>10</td>
</tr>
<tr>
<td>eNFN [83]</td>
<td>T-S</td>
<td>0.1210</td>
<td>107</td>
</tr>
<tr>
<td>eTS [27]</td>
<td>T-S</td>
<td>0.8303</td>
<td>9</td>
</tr>
<tr>
<td>EMFSPO</td>
<td>Mamdani</td>
<td>0.0139</td>
<td>60</td>
</tr>
</tbody>
</table>

In a similar way as Section 3.2.4.1, we list two fuzzy rules form the 60 rules learned...
Example 2: high-dimensional nonlinear system identification

Fig. 3.2: Simulation results for example 2.

by EMFSPO to show the interpretability:

\[
R_1: \text{If } x_1 \text{ is } \Gamma_{1,1}, \text{ and } x_2 \text{ is } \Gamma_{1,2}, \ldots, \text{ and } x_{11} \text{ is } \Gamma_{1,11}, \text{ Then } y_1 = -0.7226; \quad (3.41)
\]

\[
R_2: \text{If } x_1 \text{ is } \Gamma_{2,1}, \text{ and } x_2 \text{ is } \Gamma_{2,2}, \ldots, \text{ and } x_{11} \text{ is } \Gamma_{2,11}, \text{ Then } y_2 = -1.2300. \quad (3.42)
\]

The centres and radii of \( R_1 \) and \( R_2 \) are:

\[
c_1 = [-0.6225, -0.3231, -0.1654, -0.1780, -0.2592, -0.3264, -0.3352, -0.2592, -0.3264, -0.3352, -0.2676, -0.1220, 0.0917, 0.5137] \text{ and } \sigma_1 = [0.0303, 0.0156, 0.0087, 0.0095, 0.0108, 0.0119, 0.0126, 0.0126, 0.0129, 0.0150, 0.0444]; \quad (3.43)
\]

\[
c_2 = [-1.4661, -1.6317, -1.7187, -1.7287, -1.6725, -1.5576, -1.3972, -1.2166, -1.0005, -0.7631, -0.5878] \text{ and } \sigma_2 = 10^{-4}[0.0020, 0.0001, 0.0002, 0.0098, 0.1650, 0.0023, 0.0045, 0.3844, 0.6545, 0.0726, 0.0755]. \quad (3.44)
\]
3.3 LEOA: Learning Evolving T-S Fuzzy Systems Based on A Local Error Optimization Approach

In Section 3.2, we proposed the local optimal approach EMFSPO with regards to Mamdani fuzzy systems. Whereas in this section, we propose a generalized local optimal approach on T-S fuzzy systems. This new approach is called LEOA.

LEOA is introduced in the following arrangement: Section 3.3.1 gives a brief description of the T-S fuzzy system which needs to be identified. The detailed description of structure learning strategies (i.e. fuzzy rule generation, pruning and merging strategies), and parameters (i.e. antecedent and consequent parameters) updating techniques includes EWRLS are deliberated in Section 3.3.2. In Section 3.3.3, the detailed steps and flowchart of the LEOA are presented. Section A.4.1 and 3.3.4.1 analyze the complexity and sensitivity of LEOA. Numerical results of LEOA across four well known benchmark examples are presented in Section 3.3.4.

3.3.1 Problem Statement

The T-S fuzzy system considered in this section is constructed as a collection of T-S rules. The $i$-th fuzzy rule $R_i$ can be presented as follows:

$$R_i : \text{If } x_1 \text{ is } \Gamma_{i,1} \text{ and } x_2 \text{ is } \Gamma_{i,2} \text{ and } \ldots \text{ and } x_n \text{ is } \Gamma_{i,n}, \text{ then } y_i = \psi_{i,0} + \sum_{j=1}^{n} \psi_{i,j} x_j, \quad (3.45)$$

where $i = 1, 2, \ldots, K$, $K$ is the number of fuzzy rules, $x_j$ is the $j$-th input, $y_i$ is the output of rule $R_i$, $\psi_{i,j}$, $j = 0, 2, \ldots, n$, are consequent parameters, $n$ is the dimension of the inputs.

In LEOA, Gaussian membership functions are chosen so that the centre of fuzzy set $\Gamma_{i,j}$ is $c_{i,j}$ and the radius is $\sigma_{i,j}$. The membership function of $\Gamma_{i,j}$ is $\mu_{i,j}(x_j)$ shown in (3.46):

$$\mu_{i,j}(x_j) = \exp\left\{ -\frac{(x_j - c_{i,j})^2}{2(\sigma_{i,j})^2} \right\}. \quad (3.46)$$

Furthermore, the form of the firing strength $\gamma_i(x)$ of rule $R_i$ and the model output $y$ are
displayed in (3.47) and (3.48), respectively:

$$
\gamma_i(x) = \prod_{j=1}^{n} \mu_{i,j}(x_j), \tag{3.47}
$$

$$
y = \sum_{i=1}^{K} \theta_i(x)y_i, \tag{3.48}
$$

in which

$$
\theta_i(x) = \gamma_i(x) / \sum_{j=1}^{K} \gamma_j(x) \tag{3.49}
$$

is the normalized firing strength of rule $R_i$.

The problem which needs to be solved is that of identifying the number of fuzzy rules $K$, antecedent parameters $c_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,n})$, $\sigma_i = (\sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,n})$ and consequent parameters $\psi_i = (\psi_{i,0}, \psi_{i,1}, \psi_{i,2}, \ldots, \psi_{i,n})$, where $i = 1, 2, \ldots, K$, of the T-S fuzzy system in order to make predictions of $y$. The exact conditions and steps of the LEOA learning algorithm for identifying the T-S fuzzy systems are proposed in Sections 3.3.2 and 3.3.3.

### 3.3.2 LEOA Learning

In the same way as EMFSPO, LEOA consists of three structure evolving modules, i.e., rule generation, merging and pruning, as well as two parameter updating modules for learning the antecedent and consequent parameters. The starting point for getting the exact approaches of each module is the minimization of the local error function shown as:

$$
Err_i = \sum_{k=1}^{t+1} \theta_i^{(t+1)}(x_k)(y_k - xe_k \psi_i)^2, \tag{3.50}
$$

where $i = 1, 2, \ldots, K$, $k = 1, 2, \ldots, t + 1$ is the time, $x_k = (x_{k,1}, x_{k,2}, \ldots, x_{k,n})$ is the input at time $k$, $xe_k = (1, x_k)$ is the generalized input at time $k$. The $\theta_i^{(t+1)}(x_k) = \gamma_i^{(t+1)}(x_k) / \sum_{l=1}^{K} \gamma_l^{(t+1)}(x_k)$, $\gamma_i^{(t+1)}(\cdot)$ and $K^{(t+1)}$ are the normalized firing strength of $R_i$, firing strength of $R_i$ and rule numbers, respectively, at time $t + 1$. If no evolving and updating happens on the structure and antecedent, then the consequent parameters $\psi_i$ can be computed using WRLS, which can be induced from:

$$
\psi_i = \{R^i(t+1)\}^{-1}f^i(t+1), \tag{3.51}
$$
where \( \overline{R^i}(t+1) = \sum_{k=1}^{t+1} \theta_i^{(t+1)}(x_k)x_kx_k^T \), \( f^i(t+1) = \sum_{k=1}^{t+1} \theta_i^{(t+1)}(x_k)x_ky_k \).

However, the antecedent and consequent parameters of EFSs are actually changing, so WRLS cannot help to find the optimal solution of (3.50). In the remaining parts of this section, we utilize the point of view of minimizing (3.50) to find out the appropriate approaches for structure learning and parameter learning. All the following structure and consequent learning conditions and theorems are obtained under the assumption that the system can be approximated by a finite number of fuzzy rules \( (K < \infty, k = 1, 2, \ldots, t) \) in any accuracy in finite steps. Additionally, \( \{x_k\}_{k=1}^{\infty} \) and \( \{y_k\}_{k=1}^{\infty} \) are bounded.

### 3.3.2.1 Structure Learning

LEOA is suitable for online learning, and its rule base is initially empty and can be expanded and shrunk along with new coming data. This section gives the detailed description of how LEOA evolves its structure.

#### 3.3.2.1.1 Fuzzy Rule Generation

Adding fuzzy rules results in expanding the fuzzy rule base to cover the input space. Once the data stream is detected to change to a new state, then there is always a new fuzzy rule which needs to be generated. In order to get the exact fuzzy rule generation condition, we assume that the system remains unchanged until time \( t \) (i.e. \( c_i(1) = c_i(2) = \ldots = c_i(k), \sigma_i(1) = \sigma_i(2) = \ldots = \sigma_i(k) \), \( c_i(k) \) is the cluster centre of the rule \( R_i \) at time \( k \), \( \sigma_i(k) \) is the cluster radius of the \( i \)-th fuzzy rule at time \( k \), for \( i = 1, 2, \ldots, K(t) \) and \( k = 1, 2, \ldots, t \), \( K(1) = K(2) = \ldots = K(t) \), and a new fuzzy rule is generated at time \( t+1 \) with cluster centre \( c_{K(t+1)} = x_{t+1} \). Therefore, for \( i = 1, 2, \ldots, K(t) \), the objective function (3.50) can be transformed to (3.52):

\[
Err_i = \sum_{k=1}^{t} \theta_i^{(t+1)}(x_k)(y_k - xe_k\psi_i)^2 + \theta_i^{(t+1)}(x_{t+1})(y_{t+1} - xe_{t+1}\psi_i)^2, \tag{3.52}
\]

where

\[
\theta_i^{(t+1)}(x_k) = \frac{\gamma_i^{(t+1)}(x_k)}{\sum_{l=1}^{K(t+1)} \gamma_l^{(t+1)}(x_k)} = \frac{\gamma_i^{(t+1)}(x_k)}{\sum_{l=1}^{K(t)} \gamma_l^{(t+1)}(x_k) + \gamma_i^{(t+1)}(x_k)}. \tag{3.53}
\]
Then, $\overline{R}'(t+1)$, $f^i(t+1)$ and $\psi_i^{(t+1)}$ can be represented by (3.54), (3.55) and (3.58), respectively.

$$\overline{R}'(t+1) = \sum_{k=1}^{t} \theta_i^{(t+1)}(x_k)x_k x_k^T + \theta_i^{(t+1)}(x_{t+1})x_{t+1} x_{t+1}^T$$

$$= \overline{R}'(t) + \sum_{k=1}^{t} A_i^{(t+1)} x_k x_k^T + \theta_i^{(t+1)}(x_{t+1})x_{t+1} x_{t+1}^T, \quad (3.54)$$

and

$$f^i(t+1) = \sum_{k=1}^{t} \theta_i^{(t+1)}(x_k)x_k y_k + \theta_i^{(t+1)}(x_{t+1})x_{t+1} y_{t+1}$$

$$= f^i(t) + \sum_{k=1}^{t} A_i^{(t+1)} y_k + \theta_i^{(t+1)}(x_{t+1})x_{t+1} y_{t+1}, \quad (3.55)$$

where

$$A_i^{(t+1)} = \frac{\gamma_i^{(t+1)}(x_k) - \gamma_i^{(t+1)}(x_{t+1})}{(\sum_{l=1}^{K} \gamma_l^{(t+1)}(x_k))(\sum_{l=1}^{K} \gamma_l^{(t+1)}(x_{t+1}))} x_k,$$

$$\overline{R}'(t) = \sum_{k=1}^{t} \theta_i^{(t)}(x_k)x_k x_k^T \quad \text{and} \quad f^i(t) = \sum_{k=1}^{t} \theta_i^{(t)}(x_k)x_k y_k. \quad (3.57)$$

Thereafter, $\psi_i^{(t+1)}$ can be computed by:

$$\psi_i(t+1) = \overline{R}'^{-1}(t+1)f^i(t+1)$$

$$= \psi_i(t) - \overline{R}'^{-1}(t+1)\{\sum_{k=1}^{t} A_i^{(t+1)} x_k x_k^T + \theta_i^{(t+1)}(x_{t+1})x_{t+1} x_{t+1}^T\} \psi_i(t)$$

$$- \sum_{k=1}^{t} A_i^{(t+1)} y_k - \theta_i^{(t+1)}(x_{t+1})x_{t+1} y_{t+1}. \quad (3.58)$$

Based on (3.58), if Condition 3.5 is satisfied, then we can get a simplified updating equation of $\psi_i$ shown in (3.59).

**Condition 3.5.** If $\gamma_i^{(t)}(x_{t+1}) < \varepsilon^{(a)}$, then a new cluster would be generated with centre $c_{K(t+1)} = x_{t+1}$, radius $\sigma_{K(t+1)}^2 = n \min\{\frac{(c_{ij} - k_0 c_{K(t+1)})^2}{(c_{ij} + k_0 c_{K(t+1)})^2}, (c_{ij} - k_0 c_{K(t+1)})^2\}$, where $j = 1, 2, \ldots, n$, $i = 1, 2, \ldots, K$, $k_0 = \sqrt{-2 \log \frac{\varepsilon}{n}} > 0$, $\varepsilon^{(a)} > 0$, $\varepsilon_0 > 0$ is the tolerance degree.
In Condition 3.5, the radius \( \sigma_{K(t+1),j} \) of the new cluster is initialized to make sure that these fuzzy clusters have a low degree of overlapping and that they avoid rule confliction. Therefore, data points in other clusters have a low firing strength in the new generated cluster with centre \( c_{K(t+1)} \). To ensure the optimality, there is a need to make sure that any historical data \( x \) has a firing strength smaller than \( \frac{\epsilon_0}{T} \) in the new cluster (i.e. \( \gamma_{K(t+1)}(x) \leq \frac{\epsilon_0}{T} \)). Furthermore, as \( \gamma_{K(t+1)} \) is a multivariate Gaussian density. Hence, on each dimension, the formula \( \sigma_{K(t+1),j}^2 = n \min \left\{ \frac{\min\{(c_{i,j}-k_0\sigma_{i,j}-c_{K(t+1),j})^2, (c_{i,j}+k_0\sigma_{i,j}-c_{K(t+1),j})^2\}}{-2\log \frac{\epsilon_0}{T}} \right\} \) holds can induce \( \gamma_{K(t+1)}(x) \leq \frac{\epsilon_0}{T} \) holds.

Furthermore, threshold \( \epsilon(a) \) is set based on the \( \alpha \)-cut [191] of the fuzzy sets. The value of \( \epsilon(a) \) is 0.05. Here, the \( \alpha \)-cut of each fuzzy set \( \Gamma_i \) can be presented by \( \mathcal{A}_{\alpha}^i = \{ x \in \Gamma_i | \gamma(x) \geq \alpha \} \), where \( i = 1, 2, \ldots, K \), \( \alpha = \epsilon(a) = 0.05 \). Condition 3.5 indicates that if there exists a data point \( x^* \), which is located outside all the \( \alpha \)-cuts of fuzzy sets \( \Gamma_i \), \( i = 1, 2, \ldots, K \), then a new fuzzy rule \( R_{K+1} \) should be built such that \( x^* \in \mathcal{A}_{\alpha}^{K+1} \). This condition ensures that \( \{x_t^T \}_{t=1} \subset \bigcup_i \mathcal{A}_{\alpha}^i \), in which \( \{x_t^T \}_{t=1} \) stands for the set formed by all the historical data points. Besides, tolerance degree \( \epsilon_0 \) is a small value to ensure the optimality of consequent parameters, but \( \epsilon_0 \) can not be set too small in case the radius of the new generated rule becomes 0.

**Definition 3.1.** (\( \epsilon \)-completeness [74]) For any input \( x \in \Omega \), there exists at least one fuzzy rule such that the activation degree of \( x \) is no less than \( \epsilon \).

**Theorem 3.4.** Assume the Condition 3.5 is satisfied and rule \( R_{K(t+1)} \) is the new generated rule. Then, \( \psi^{(t+1)}_i \) which minimizes (3.50) can be updated using:

\[
\psi_i(t + 1) = \psi_i(t) + \frac{\theta_i^{(t+1)}(x_{t+1})x_{t+1}}{\epsilon_{t+1}^2} \{ y_{t+1} - x_{t+1}^T \psi_i(t) \}, \quad (3.59)
\]

for all \( i = 1, 2, \ldots, K(t+1) - 1 \), and \( \epsilon \)-completeness in Definition 3.1 holds.

**Proof.** See Appendix A.1.4.

### 3.3.2.1.2 Fuzzy Rule Pruning

In order to prevent overfitting, the fuzzy rule pruning procedure is proposed. Similar to Section 3.3.2.1.1, we aim to get the appropriate condition under which the optimum solution of minimizing \( \text{Err}_t \) can be calculated by simple recursive steps. Assume that the \( i^* \)-th fuzzy rule \( R_{i^*} \) is discarded at time \( t + 1 \) and for \( k = 1, 2, \ldots, t, K^{(1)} = K^{(2)} = \ldots =
\( K(t) \) holds, and \( K(t+1) = K(t) - 1 \). In order to minimize (3.50), the form of \( R^t(t + 1) \) and \( f^i(t + 1) \) are given in (3.60) and (3.61):

\[
R^t(t + 1) = R^t(t) + \sum_{k=1}^{t} B^t_{i,k} x_k e_k^T + \theta_i^{(t+1)}(x_{t+1}) x_{t+1} e_{t+1}^T, \quad (3.60)
\]

\[
f^i(t + 1) = f^i(t) + \sum_{k=1}^{t} B^t_{i,k} y_k + \theta_i^{(t+1)}(x_{t+1}) x_{t+1} y_{t+1}, \quad (3.61)
\]

in which

\[
B^t_{i,k} = \frac{\gamma_i^{(t+1)}(x_k) \gamma_i^{(t)}(x_k)}{\left( \sum_{l=1}^{K(t)} \gamma_i^{(t)}(x_k) \right) \left( \sum_{l=1, l \neq i}^{K(t)} \gamma_i^{(t+1)}(x_k) \right)} x_k. \quad (3.62)
\]

From the expressions of \( R^t(t + 1) \) and \( f^i(t + 1) \), it can be seen that the EWRLS optimum solution of (3.50) can be obtained by (3.63) in Theorem 3.5, when Condition 3.6 is satisfied.

**Condition 3.6.** If \( \sum_{p=t+1}^{k} \gamma_i^{(p)}(x_p) < \epsilon^{(p)} \) and \( k - t^* > M \) \( (M > 0 \) is a large number) hold, then the rule \( R_i \) should be removed. The \( t^* \) is the time, at which the rule \( R_i \) is built, \( k \) is the time stamp, \( \epsilon^{(p)} > 0 \) is the tolerance degree.

The threshold \( \epsilon^{(p)} \) for cumulative firing strength is set at a small value: 0.05. Threshold \( M \), which controls the age, is recommended to be larger than 50, as a sample size above 50 can be regarded as a large sample in statistics. A tiny threshold may lead to some useful fuzzy rules being deleted too early, which would make it hard to guarantee the optimality of the consequent parameters. A huge threshold allows the redundant fuzzy rules to be used for a long time, which would increase the computational burden. Therefore, for online tracking applications, \( M \) can be set smaller than 50 in order to satisfy the need of giving quick online predictions and by putting the global optimality on a secondary status.

**Theorem 3.5.** Assume Condition 3.6 is satisfied, and rule \( R_i \) is removed at step \( t + 1 \). Then, \( \psi_i(t + 1) \) which minimizes (3.50) can be updated by:

\[
\psi_i(t + 1) = \psi_i(t) + \frac{1}{R^t(t + 1)} \theta_i^{(t+1)}(x_{t+1}) x_{t+1} \left\{ y_{t+1} - x_{t+1}^T \psi_i(t) \right\}, \quad (3.63)
\]

for all \( i = 1, 2, \ldots, K^{(t+1)} - 1 \).
Proof. See Appendix A.1.5.

3.3.2.1.3 Fuzzy Rule Merging

After calibrating the cluster centres and radii with the continuous coming data, there would exist some fuzzy rules which would grow to be similar. In this situation, it is neither necessary nor appropriate to apply several fuzzy rules in order to approximate the data points in the same cluster, because this process may cause a conflict of rules and increase the computation burden. Following the same process shown in Section 3.3.2.1.1 and 3.3.2.1.2, we assume that the rules $R_{l_1}$ and $R_{l_2}$ are merged in step $t + 1$. Write the merged result to be rule $R_{l_0}$, which will be used to replace the rule $R_{l_i}$ after merging process. Furthermore, $R_{l_2}$ will be deleted. To begin with, $\overline{R}'(t + 1)$ and $f(t + 1)$ can be presented by (3.64) and (3.65):

$$\overline{R}'(t + 1) = \overline{R}'(t) + \sum_{k=1}^{l} C_{t,k}^i x e_k^T + \frac{\gamma_i^{(t+1)}(x_{t+1})}{\sum_{l=1}^{K^{(t+1)}} \gamma_l^{(t+1)}(x_{t+1})} x e_{t+1} x e_k^T, \quad (3.64)$$

$$f'(t + 1) = f'(t) + \sum_{k=1}^{l} C_{t,k}^i y_k + \frac{\gamma_i^{(t+1)}(x_{t+1})}{\sum_{l=1}^{K^{(t+1)}} \gamma_l^{(t+1)}(x_{t+1})} x e_{t+1} y_{t+1}, \quad (3.65)$$

$$C_{t,k}^i = \frac{\gamma_i^{(t)}(x_k) [\gamma_i^{(t)}(x_k) + \gamma_{l_2}^{(t)}(x_k) - \gamma_{l_0}^{(t)}(x_k)]}{\sum_{l=1}^{K^{(t+1)}} \gamma_l^{(t+1)}(x_k) \sum_{l=1}^{K^{(t+1)}} \gamma_l^{(t)}(x_k)} x e_k. \quad (3.66)$$

The method used to judge the similarity of two rules and the antecedent parameters of the merged rule can be found in the Condition 3.7:

**Condition 3.7.** When $k = t + 1$ and there exist “large enough” small number $\varepsilon_c > 0$ and $\varepsilon_\sigma > 0$ such that $\| c_{l_1} - c_{l_2} \| < \varepsilon_c$ and $\| \sigma_{l_1}^2 - \sigma_{l_2}^2 \| < \varepsilon_\sigma$, where $\| \cdot \|$ is the Euclidean norm, then rule $R_{l_1}$ and $R_{l_2}$ are merged to $R_{l_0}$ ($l_1 = l_0$ numerically). The centre and the radius of rule $R_{l_0}$ are calculated by (3.67) and (3.68):

$$c_{l_0} = \frac{\sum_{l_0} x(k_0)}{N_{l_0}} = \frac{N_{l_1} c_{l_1} + N_{l_2} c_{l_2}}{N_{l_1} + N_{l_2}}, \quad (3.67)$$

$$\left(\sigma_{l_0,j}\right)^2 = \frac{\sum_{l_0} (x_{k_0,j} - c_{l_0,j})^2}{N_{l_0} - 1} = \frac{(N_{l_1} - 1)(\sigma_{l_1,j})^2 + (N_{l_2} - 1)(\sigma_{l_2,j})^2 + \frac{N_{l_1} N_{l_2} (c_{l_1,j} - c_{l_2,j})^2}{N_{l_1} + N_{l_2}}}{N_{l_1} + N_{l_2} - 1}. \quad (3.68)$$
CHAPTER 3. EVOLVING FUZZY SYSTEMS: OPTIMUM APPROACHES

The statistical theory behind the above two formulas (3.67) and (3.68) used in Condition 3.7 are the unbiased estimators $\hat{c}$ and $\hat{\sigma}^2$ for expected value $E(x)$ and variance $\text{Var}(x)$ of a random variable $x$, separately, shown as:

$$\hat{c} = \frac{\sum_{k=1}^{N} x_k}{N}, \quad \hat{\sigma}^2 = \frac{\sum_{k=1}^{N} (x_k - \hat{c})^2}{N-1},$$  \hspace{1cm} (3.69)

where $x_1, x_2, \ldots, x_N$ are the samples; $N$ is the number of samples. The mathematical deduction for (3.68) is presented in Appendix A.1.7.

In practice, it is hard to choose very small control parameters $\varepsilon_c$ and $\varepsilon_\sigma$ to ensure that two clusters have a high overlapping level. Therefore, an alternative criterion for merging rules $R_{l1}$ and $R_{l2}$ is given in (3.70) and (3.71):

$$1 - \max_{k=1,2,\ldots,n} \left\{ \max \left\{ \frac{\|c_{l1,k} - c_{l2,k}\|}{\sigma_{l1,k}}, \frac{\|c_{l1,k} - c_{l2,k}\|}{\sigma_{l2,k}} \right\} \right\} > \varepsilon_c^*,$$

$$\min_{k=1,2,\ldots,n} \left\{ \min \left\{ \frac{\sigma_{l1,k}}{\sigma_{l2,k}}, \frac{\sigma_{l2,k}}{\sigma_{l1,k}} \right\} \right\} > \varepsilon_\sigma^*,$$\hspace{1cm} (3.70) (3.71)

where $\varepsilon_c^*$ and $\varepsilon_\sigma^*$ are control parameters indicating the similarity of two clusters. It is obvious that the more $\varepsilon_c^*$ and $\varepsilon_\sigma^*$ are close to 1, the larger is the overlapping part of the corresponding two clusters. Threshold $\varepsilon_c^*$ is recommended to be selected based on the $\alpha$-cut ($\alpha = 0.95$) of the fuzzy set with $\varepsilon_c^* = \sqrt{-\frac{2\log(0.95)}{n}}$, and the suggested value of $\varepsilon_\sigma^*$ is 0.9.

Theorem 3.6, which gives an approach to updating the consequent parameters when rule merging happens, can be obtained from (3.64) — (3.65) and the Condition 3.7.

**Theorem 3.6.** Assume the Condition 3.7 is satisfied, and the rule $R_{l0}$ is the merging results obtained from rule $R_{l1}$ and $R_{l2}$ at time $t + 1$. Then, $\psi^{(t+1)}_i$ which minimizes (3.50) can be updated by (3.72) — (3.75):

$$\psi_i(t+1) = \tilde{\psi}_i^{(l_2)}(t) + \tilde{R}_{l2}(t)^{-1} \theta_i^{(t+1)}(x_{t+1}) x e_{t+1} \{ y_{t+1} - x e_{t+1}^T \psi_i^{(l_2)}(t) \},$$ \hspace{1cm} (3.72)

where $\tilde{\psi}_i^{(l_2)}(t)$ and $\tilde{R}_{l2}(t)^{-1}$ are shown in (3.73) and (3.75):

$$\tilde{\psi}_i^{(l_2)}(t) = \tilde{\psi}_i^{(l_p)}(t - 1) + \tilde{L}_i^{(l_p)}(t) \{ y_t - x e_t^T \tilde{\psi}_i^{(l_p)}(t - 1) \},$$ \hspace{1cm} (3.73)

$$\tilde{P}_i^{(l_p)}(t) = \tilde{P}_i^{(l_p)}(t - 1) - \frac{M_i^{(l_p)}(x_t) \tilde{P}_i^{(l_p)}(t - 1) x e_t x e_t^T \tilde{P}_i^{(l_p)}(t - 1)}{1 + M_i^{(l_p)}(x_t) x e_t^T \tilde{P}_i^{(l_p)}(t - 1) x e_t},$$ \hspace{1cm} (3.74)
\[ T_i^{(l_p)} = \frac{M_i^{(l_p)}(x_t) \tilde{P}_i^{(l_p)}(t) x_t}{1 + M_i^{(l_p)}(x_t) x_t^T \tilde{P}_i^{(l_p)}(t - 1) x_t}, \quad (3.75) \]

with \( l_p = l_2, \tilde{R}_i^{(l_p)}(t) = \sum_{k=1}^{i} M_i^{(l_p)}(x_k) x_k x_k^T, \)

\[ M_i^{(l_p)}(x_k) = \theta_i^{(t)}(x_k) + \frac{\gamma_i^{(t)}(x_k) \gamma_i^{(t)}(x_k)}{[\sum_{l=1}^{K(t)} \gamma_i^{(t)}(x_k)] [\sum_{l=1}^{K(t)} \gamma_i^{(t)}(x_k) - \gamma_i^{(t)}(x_k)]}. \]

**Proof.** See Appendix A.1.6.

### 3.3.2.2 Parameters Learning

This section is used to present the parameters for updating approaches of LEOA. This parameters learning process is discussed in two aspects: Antecedent parameters learning, and consequent parameters learning in Section 3.3.2.2.1 and 3.3.2.2.2, respectively.

#### 3.3.2.2.1 Antecedent Parameters Learning

Depending on the antecedent learning procedure for minimizing (3.50), the updating method of existing cluster centres and radii can be obtained and shown in Condition 3.8 as follows:

**Condition 3.8.** If there exists an \( i^* \in \{1, 2, \ldots, K^{(t)}\} \) and \( \gamma_i^{(t)}(x_{t+1}) \geq \epsilon^{(a)} \) (\( \epsilon^{(a)} > 0 \) is the same value as what is used in the Condition 3.5), then the centre \( c_{i^*} \) and the radius \( \sigma_{i^*} \) can be updated by (3.76) and (3.77):

\[
c_{i^*,j}(t + 1) = c_{i^*,j}(t) + \frac{x_{t+1,j} - c_{i^*,j}(t)}{N_{i^*}(t) + 1} \quad (3.76)
\]

\[
(\sigma_{i^*,j}(t + 1))^2 = (\sigma_{i^*,j}(t))^2 + \frac{(x_{t+1,j} - c_{i^*,j}(t + 1))^2 - (\sigma_{i^*,j}(t))^2}{N_{i^*}(t)} + (c_{i^*,j}(t + 1) - c_{i^*,j}(t))^2, \quad (3.77)
\]

in which \( N_{i^*}(t) \) is the total number of data points which satisfy \( \gamma_i^{(t)}(x_{t+1}) \geq \epsilon \) until time \( t \). (Note that what only have to be ensured are \( c_{i^*} \rightarrow c_{i^*}^0 \) and \( \sigma_{i^*} \rightarrow \sigma_{i^*}^0 \) when \( N_{i^*} \rightarrow \infty \), where \( c_{i^*}^0 \) and \( \sigma_{i^*}^0 \) are the real centre and radius of rule \( R_{i^*} \).)
Updating formulas of rule centres and radii in (3.76) and (3.77) are also induced based on (3.69).

### 3.3.2.2 Consequent Parameters Learning

The recursive least square method has been widely used in control theory to minimize the error function [192–195]. In order to serve the same optimality purpose as the antecedent learning procedure of LEOA in Section 3.3.2.1, the consequent parameters can be updated by an EWRLS method. According to Theorem 3.4, 3.5 and 3.6 in Section 3.3.2.1, the updating formulas of EWRLS would be influenced by whether or not a merging process has occurred. All the following formulas can be induced following the WRLS derivation procedure in [190]. EWRLS updating formulas can be summarized as:

(I) If no fuzzy rules are merged, consequent parameters should be learned by WRLS with formulas (3.78) and (3.80):

\[
\psi_i(t+1) = \psi_i(t) + \theta_i^{(t+1)}(x_{t+1}) P_i(t+1) x e_{t+1} (y_{t+1} - x e_{t+1}^T \psi_i(t)), \tag{3.78}
\]

\[
L_i(t+1) = \frac{\theta_i^{(t+1)}(x_{t+1}) P_i(t) x e_{t+1}}{1 + \theta_i^{(t+1)}(x_{t+1}) x e_{t+1}^T P_i(t) x e_{t+1}}, \tag{3.79}
\]

\[
P_i(t+1) = P_i(t) - \frac{\theta_i^{(t+1)}(x_{t+1}) P_i(t) x e_{t+1} P_i(t)}{1 + \theta_i^{(t+1)}(x_{t+1}) x e_{t+1}^T P_i(t) x e_{t+1}}. \tag{3.80}
\]

(II) Otherwise, assuming that two fuzzy rules \(R_{l_0}\) and \(R_{l_1}\) are merged to \(R_{l_0}\), the EWRLS updating formulas can be presented as those in (3.81), (3.82) and (3.83):

\[
\psi_i(t+1) = \tilde{\psi}_i^{(l_0)}(t) + L_i(t+1) (y_{t+1} - x e_{t+1}^T \tilde{\psi}_i^{(l_0)}(t)), \tag{3.81}
\]

\[
L_i(t+1) = \frac{\theta_i^{(t+1)}(x_{t+1}) \tilde{P}_i^{(l_0)}(t) x e_{t+1}}{1 + \theta_i^{(t+1)}(x_{t+1}) x e_{t+1}^T \tilde{P}_i^{(l_0)}(t) x e_{t+1}}, \tag{3.82}
\]

\[
P_i(t+1) = \tilde{P}_i^{(l_0)}(t) - \frac{\theta_i^{(t+1)}(x_{t+1}) \tilde{P}_i^{(l_0)}(t) x e_{t+1} \tilde{P}_i^{(l_0)}(t)}{1 + \theta_i^{(t+1)}(x_{t+1}) x e_{t+1}^T \tilde{P}_i^{(l_0)}(t) x e_{t+1}}. \tag{3.83}
\]

where \(\tilde{\psi}_i^{(l_0)}(t)\) and \(\tilde{P}_i^{(l_0)}(t)\) are updated using (3.73) and (3.74). One should be aware that \(\tilde{\psi}_i^{(l_0)}(t)\) and \(\tilde{P}_i^{(l_0)}(t)\) need to be calculated and updated since \(k=3\), and there is some extra information which needs to be recorded in each learning step.
For each $i = 1, 2, \ldots, K$, $l = 1, 2, \ldots, K$, $l \neq i$, $\tilde{\psi}_i^l(t)$ and $\tilde{P}_i^l(t)$ should be reserved and used when fuzzy rule merging occurs.

### 3.3.3 Mechanism of LEOA

![Diagram of LEOA mechanism]

**Fig. 3.3:** Computational steps of LEOA.
In order to give a clear idea of how LEOA connects the structure and parameters learning methods shown in Section 3.3.2, the exact learning steps and a flowchart (Fig. 3.3) are presented in this section.

Before presenting the specific steps, we need to introduce two indicators $\text{ind}_a$ and $\text{ind}_m$. These are the indicators for rule generation and merging respectively. The default value of $\text{ind}_a$ is 0 which stands for no new rule should be generated. Once a new rule is generated, the value of $\text{ind}_a$ should be switched to 1. In a similar way to $\text{ind}_a$, $\text{ind}_m$ also has the default value of 0, which should be changed to 1 when merging occurs, or otherwise kept as 0. Furthermore, a complexity analysis of LEOA is presented in the Appendix A.4.1.

step 1 Read new input data $x_{t+1}$, set $\text{ind}_a = 0$ and $\text{ind}_m = 0$.

step 2 Rule generation and updating

If $\gamma_i^{(t)}(x_{t+1}) < \varepsilon^{(a)}$ for all $i = 1, 2, \ldots, K^{(t)}$, then generate a new fuzzy rule $K^{(t+1)} = K^{(t)} + 1$, change $\text{ind}_a = 1$. Otherwise, update the centre and radius of the rule $R_{i^*}$ by (3.76) and (3.77), where $i^* = \arg \max_i \{ \gamma_i^{(t)}(x_{t+1}) \}$.

step 3 Rule pruning

If $t_{max} = \max (t + 1 - t^*_i) > M$, $i^* = \arg \max_i (t + 1 - t^*_i)$ and $\sum_{k = t^*_i}^{t+1} \gamma_i^{(t)}(x_k) < \varepsilon^{(p)}$, then remove rule $R_{i^*}$.

step 4 Rule merging

If $\text{ind}_a = 0$, and the antecedent parameters of the rule $R_{i^*}$ are updated, then this merging step should be applied. If there exist $l_q \in \{1, 2, \ldots, R_t\}$ and $l_q \neq i^*$ such that $c_{l_q}^{(t)}, c_{l_q}^{(t)}$, $\sigma_{l_q}$ and $\sigma_{l_q}^{(t)}$ satisfy (3.70), then rule $R_{l_q}$ and rule $R_{i^*}$ should be merged to rule $R_{l_0}$ ($i^* = l_0$ numerically) with the centre and the radius calculated through (3.67) and (3.68). At this time, the indicator should be set as $\text{ind}_m = 1$.

step 5 Output

Compute the output using $\hat{y}_{t+1} = \sum_{i=1}^{K^{(t+1)}} \theta_i^{(t+1)}(x_{t+1}) x_{t+1} \psi_i(t)$.

step 6 EWRLS

If $\text{ind}_m = 0$, then use (3.78), (3.79) and (3.80) to update the consequent parameters $\psi_i^{(l)}$, otherwise apply (3.81), (3.82) and (3.83). For each $i = 1, 2, \ldots, K^{(t+1)}$ and $l = 1, 2, \ldots, K^{(t+1)}$ with $l \neq i$, update the $\bar{\psi}_i^{(l)}(t)$ and $\tilde{P}_i^{(l)}(t)$ using (3.73), (3.74) and (3.75).
3.3.4 Numerical Examples

Four numerical examples include classical benchmark examples and real-world data predictions which are carried out to assess LEOA. The datasets are the Dow Jones Industrial Average (DJIA) daily closing prices, the datasets generated by two nonlinear dynamic systems and the Mackey-Glass chaotic time series. Furthermore, the sensitivity analysis of LEOA is given in Section 3.3.4.1. The motivation for testing LEOA on these datasets is to test LEOA on a variety of datasets that are nonstationary, nonlinear, as well as having a degree of uncertainty by nature. As well-known benchmark examples, these datasets can effectively evaluate the learning ability and the global optimality of LEOA. In the following subsections, LEOA is compared with many existing state-of-the-art methods. The performances are evaluated by the RMSE and non-dimensional error index (NDEI) (see (B.2) in the Appendix B.1), which have the form (B.1) and (B.2), respectively. Note that the “global accuracies” in this section are obtained by the following steps: 1) LEOA runs in an online mode on the training data set and stops to evolve its structure after this phase; 2) The final model is used to make estimations of all the historical data and to obtain the “global accuracy”.4

3.3.4.1 Sensitivity Analysis

There are two predefined thresholds (\(\varepsilon_0\) and \(M\)) of LEOA. Various values of \(\varepsilon_0\) and \(M\) are chosen in order to make sure that LEOA is not problem-dependent. The well known Box-Jenkins gas furnace dataset [35, 36, 55, 75, 76, 95, 196, 197] is cast to investigate whether LEOA is sensitive to \(\varepsilon_0\) and \(M\). The description of the dataset and the input-output setting can be found in Appendix B.1. LEOA runs in an online mode. All 290 outputs are used to evaluate the performance of LEOA. In Section 3.3.2.1.1, threshold \(\varepsilon_0\) is suggested to be a small value, hence, \(\varepsilon_0\) is taken as \([10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]\). Besides, sample size threshold \(M\) is chosen from \([50, 70, 90, 110, 130]\). Other thresholds are kept as their default settings. Online prediction results are presented in Table 3.3. To test the global accuracy of LEOA, we use the final model learned from the data to make estimations of all the historical data, the results of which are presented in the Table 3.4. The results are evaluated by NDEI. The number of rules are given in the bracket in Table 3.3 and 3.4.

4Results of LEOA are calculated in the environment of intel(R) core (TM) i7-4790 CPU with a 3.6 GHz processor and a 16.0 GB memory.
TABLE 3.3: Sensitivity of predefined thresholds to online testing accuracy.

<table>
<thead>
<tr>
<th>$\varepsilon_0 = 10^{-1}$</th>
<th>$\varepsilon_0 = 10^{-2}$</th>
<th>$\varepsilon_0 = 10^{-3}$</th>
<th>$\varepsilon_0 = 10^{-4}$</th>
<th>$\varepsilon_0 = 10^{-5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M = 50$</td>
<td>0.2192(42)</td>
<td>0.2197(39)</td>
<td>0.2199(38)</td>
<td>0.2199(38)</td>
</tr>
<tr>
<td>$M = 70$</td>
<td>0.2206(59)</td>
<td>0.2209(60)</td>
<td>0.2209(58)</td>
<td>0.2209(58)</td>
</tr>
<tr>
<td>$M = 90$</td>
<td>0.2209(78)</td>
<td>0.2199(78)</td>
<td>0.2200(78)</td>
<td>0.2200(78)</td>
</tr>
<tr>
<td>$M = 110$</td>
<td>0.2210(91)</td>
<td>0.2161(90)</td>
<td>0.2185(89)</td>
<td>0.2182(92)</td>
</tr>
<tr>
<td>$M = 130$</td>
<td>0.2185(107)</td>
<td>0.2141(105)</td>
<td>0.2163(104)</td>
<td>0.2162(107)</td>
</tr>
</tbody>
</table>

TABLE 3.4: Sensitivity of predefined thresholds to global accuracy.

<table>
<thead>
<tr>
<th>$\varepsilon_0 = 10^{-1}$</th>
<th>$\varepsilon_0 = 10^{-2}$</th>
<th>$\varepsilon_0 = 10^{-3}$</th>
<th>$\varepsilon_0 = 10^{-4}$</th>
<th>$\varepsilon_0 = 10^{-5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M = 50$</td>
<td>0.2265(42)</td>
<td>0.2698(39)</td>
<td>0.2896(38)</td>
<td>0.2896(38)</td>
</tr>
<tr>
<td>$M = 70$</td>
<td>0.2210(59)</td>
<td>0.2554(60)</td>
<td>0.2744(58)</td>
<td>0.2745(58)</td>
</tr>
<tr>
<td>$M = 90$</td>
<td>0.2067(78)</td>
<td>0.2411(78)</td>
<td>0.2412(78)</td>
<td>0.2446(78)</td>
</tr>
<tr>
<td>$M = 110$</td>
<td>0.1921(91)</td>
<td>0.2256(90)</td>
<td>0.2338(89)</td>
<td>0.2362(92)</td>
</tr>
<tr>
<td>$M = 130$</td>
<td>0.1871(107)</td>
<td>0.2264(105)</td>
<td>0.2340(104)</td>
<td>0.2362(107)</td>
</tr>
</tbody>
</table>

It is obvious from Table 3.3 that the effect of different values of $\varepsilon_0$ and $M$ is tiny with regard to the accuracy. Table 3.4 shows that larger $M$ helps the model learned by LEOA to keep more historical information, thus giving better global accuracies. However, it always leads to a more complex fuzzy rule base and a heavier computation burden. Furthermore, small $\varepsilon_0$ does not promote the learned system to give better global fitting results. As a result, in the numerical examples presented in Section 3.3.4, $\varepsilon_0$ is chosen from 0.1, 0.01 and 0.001, and $M$ is selected to be smaller than 100 in order to balance the accuracy and complexity.

3.3.4.2 Example 1: Online Prediction of DJIA Daily Closing Price

This example is an online learning example applied to demonstrate that LEOA can make both one-step-ahead predictions and global approximations accurately. In this example, LEOA and MEPL (model 1) [198] have been used to make an online prediction of the DJIA daily closing prices. The DJIA data were collected from Wharton Research Data Services (WRDS) between 04.01.1960 and 31.12.2007 with 12118 data points in total. Both LEOA and MEPL have been applied to make predictions of the logarithm of the dataset based on the following three models, $M1$, $M2$ and $M3$, shown in (3.84),
(3.85) and (3.86):

\[ M_1 : y_{t+1} = f(y_{t-3}, y_{t-2}, y_{t-1}, y_t), \]  
\[ M_2 : y_{t+1} = f(y_{t-4}, y_{t-3}, y_{t-2}, y_{t-1}, y_t), \]  
\[ M_3 : y_{t+1} = f(y_{t-5}, y_{t-4}, y_{t-3}, y_{t-2}, y_{t-1}, y_t), \]

where \( y_t \) presents the logarithm closing price of DJIA, and \( \{y_t\}_{t=1}^{12118} \) stands for the collected dataset. In this empirical example, both LEOA and MEPL were run in an online mode throughout the whole dataset. This experimental setting can help us to fulfill the dual objectives of comparing both online learning ability and global optimality. Both the one-step-ahead online prediction results and the global prediction results are shown in Table 3.5. The RMSEs in Table 3.5 have been calculated using one-step-ahead predictions, whilst the structure and parameters of the model used in prediction vary from time to time. The global RMSEs have been calculated using the estimation of every historical data point. The estimation of the historical data points has been computed by the final system learnt from the data without updating any structure and parameters. The large global RMSE indicates that there is an “unlearning effect”. The control parameters of LEOA have been set as \( \varepsilon_0 = 10^{-3} \) and \( M = 20 \) based on loads of numerical experiments.

<table>
<thead>
<tr>
<th>Method</th>
<th>Rule Type</th>
<th>RMSEs</th>
<th>Global RMSEs</th>
<th>Rule Num.</th>
<th>Parameter Num.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEPL [198]</td>
<td>T-S</td>
<td>0.0119</td>
<td>0.0417</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td>LEOA</td>
<td>T-S</td>
<td>0.0133</td>
<td>0.0133</td>
<td>27</td>
<td>108</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Rule Type</th>
<th>RMSEs</th>
<th>Global RMSEs</th>
<th>Rule Num.</th>
<th>Parameter Num.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEPL [198]</td>
<td>T-S</td>
<td>0.0119</td>
<td>0.0423</td>
<td>20</td>
<td>80</td>
</tr>
<tr>
<td>LEOA</td>
<td>T-S</td>
<td>0.0139</td>
<td>0.0138</td>
<td>22</td>
<td>88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Rule Type</th>
<th>RMSEs</th>
<th>Global RMSEs</th>
<th>Rule Num.</th>
<th>Parameter Num.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEPL [198]</td>
<td>T-S</td>
<td>0.0121</td>
<td>0.0395</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td>LEOA</td>
<td>T-S</td>
<td>0.0145</td>
<td>0.0144</td>
<td>22</td>
<td>88</td>
</tr>
</tbody>
</table>

RMSEs in Table 3.5 demonstrate that both LEOA and MEPL can make accurate online predictions, and that MEPL has always performed a little better than LEOA with regard to the online performance. LEOA has given accurate global prediction results which are similar to its one-step-ahead prediction results for all the three models.
CHAPTER 3. EVOLVING FUZZY SYSTEMS: OPTIMUM APPROACHES

Nonetheless, it is obvious that the number of fuzzy rules used by LEOA are always larger than those used by MEPL. One reason for this phenomenon is that MEPL minimizes the error functions (3.50) to get the estimations of the consequent parameters by assuming that the fuzzy rules are unchanged. Furthermore, MEPL mainly focuses on tracking the behaviour of the most recent data, and puts global accuracy in a secondary place. However, making accurate one-step-ahead predictions without losing too much memory of the knowledge learnt from the historical data was the starting point for proposing the LEOA. Taking $M_2$ as an example, Fig. 3.4 depicts the estimation results of all the historical data using the final models learned by LEOA and MEPL. It can be seen from Fig. 3.4 that LEOA resulted in a significantly accurate prediction model than MEPL.

3.3.4.3 Example 2: Nonlinear Dynamic Plant (term 4 in the Appendix B.1)

The dataset is generated using model (B.9). Although LEOA does not need a training phase in order to make comparison with other algorithms, LEOA runs in an online mode on the training dataset and ceased evolving its structure after this phase. The numerical results for predicting the testing data samples are displayed in Table 3.6. Table 3.6 shows that LEOA has achieved higher accuracy than other state-of-art methods measured

\footnote{“—” stands for having not been listed in the source paper.}
by the RMSEs. The prediction results for the 200 testing data, which are presented in Fig. 3.5, indicate that LEOA is a powerful tool for making accurate predictions. Thereafter, the final model was used to make estimations of all the 5200 historical data. The predicting RMSE has reached 0.0033, which is still a small value. Fig. 3.6 presents the prediction errors of LEOA for the first 1000 historical data. It is obvious that LEOA has only given inaccurate predictions for the first 10 data and, after that, the prediction errors fluctuated in a narrow region around 0. The second figure in Fig. 3.6 portrays the trace of the prediction errors from the 101st to 1000th data, and shows that prediction errors have become stable and fluctuate between $-0.0085$ and $0.0073$.

<table>
<thead>
<tr>
<th>Table 3.6: RMSEs of example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>eTS [27]</td>
</tr>
<tr>
<td>simpl_eTS [95]</td>
</tr>
<tr>
<td>SAFIS [94]</td>
</tr>
<tr>
<td>FLEXFIS var A [28]</td>
</tr>
<tr>
<td>FLEXFIS var B [28]</td>
</tr>
<tr>
<td>SOFMLS [47]</td>
</tr>
<tr>
<td>eMG [54]</td>
</tr>
<tr>
<td>OSAMNN [84]</td>
</tr>
<tr>
<td>DeTS [35]</td>
</tr>
<tr>
<td>LEOA</td>
</tr>
</tbody>
</table>

![Fig. 3.5: Example 2: prediction results for testing data.](image)

![Example 2: prediction results for testing data.](image)
Due to the space limitation, only the first 5 fuzzy rules \((R_1, R_2, \ldots, R_5)\) of the final 20 rules that learned by LEOA are presented as examples to show the interpretability:

\[
R_1: \text{If } x_1 \text{ is } \Gamma_{1,1}, \text{ and } x_2 \text{ is } \Gamma_{1,2}, \text{ and } x_3 \text{ is } \Gamma_{1,3},
\text{ Then } y_1 = -0.1892 + 0.5772x_1 - 0.1059x_2 + 0.8538x_3; \quad (3.87)
\]

\[
R_2: \text{If } x_1 \text{ is } \Gamma_{2,1}, \text{ and } x_2 \text{ is } \Gamma_{2,2}, \text{ and } x_3 \text{ is } \Gamma_{2,3},
\text{ Then } y_2 = 0.1275 + 0.1385x_1 - 0.1555x_2 + 0.3722x_3; \quad (3.88)
\]

\[
R_3: \text{If } x_1 \text{ is } \Gamma_{3,1}, \text{ and } x_2 \text{ is } \Gamma_{3,2}, \text{ and } x_3 \text{ is } \Gamma_{3,3},
\text{ Then } y_3 = -0.0284 + 0.1755x_1 - 0.1733x_2 + 0.4338x_3; \quad (3.89)
\]

\[
R_4: \text{If } x_1 \text{ is } \Gamma_{4,1}, \text{ and } x_2 \text{ is } \Gamma_{4,2}, \text{ and } x_3 \text{ is } \Gamma_{4,3},
\text{ Then } y_4 = -0.1147 + 0.2540x_1 - 0.0975x_2 + 0.5281x_3; \quad (3.90)
\]

\[
R_5: \text{If } x_1 \text{ is } \Gamma_{5,1}, \text{ and } x_2 \text{ is } \Gamma_{5,2}, \text{ and } x_3 \text{ is } \Gamma_{5,3},
\text{ Then } y_5 = -0.1480 + 0.3301x_1 - 0.0164x_2 + 0.6129x_3. \quad (3.91)
\]

The centres and radii of the above rules are:

\[
c_1 = [1.0431, 1.0059, 0.8232] \text{ and } \sigma_1 = [0.0262, 0.0461, 0.0374]; \quad (3.92)
\]

\[
c_2 = [0.4485, 0.7697, 0.1253] \text{ and } \sigma_2 = 10^{-3}[0.0095, 0.5567, 0.0988]; \quad (3.93)
\]

\[
c_3 = [0.1154, 0.4485, -0.1253] \text{ and } \sigma_3 = 10^{-3}[0.0975, 0.0384, 0.0035]. \quad (3.94)
\]
\[ c_4 = [-0.1417, 0.1154, -0.3681] \text{ and } \sigma_4 = 10^{-3} [0.0046, 0.0040, 0.0621]; \]  
\[ c_5 = [-0.3580, -0.1417, -0.5878] \text{ and } \sigma_5 = 10^{-3} [0.0434, 0.0684, 0.0051]. \]  

3.3.4.4 Example 3: High-dimensional System Identification Problem (term 3 in the Appendix B.1)

| Table 3.7: RMSEs of example 3 |
|-----------------------------|--|--|--|--|
| Method  | Rule Type | RMSEs | Rule Num. | Parameter | Time(s) |
| FLEXFIS [28]  | T-S | 0.0085 | 15 | 510 | - |
| eTS [27]  | T-S | 0.0075 | 14 | 476 | - |
| eMG(\(\sum_{init} = 2 \times 10^{-1}I_{11}\)) [54]  | T-S | 0.0288 | 9 | 1296 | - |
| eMG(\(\sum_{init} = \times 10^{-1}I_{11}\)) [54]  | T-S | 0.0050 | 13 | 1872 | - |
| Gen-smart-EFS(fac=0.6) [58]  | T-S | 0.0042 | 9 | 1296 | - |
| LEOA  | T-S | 0.0027 | 61 | 2074 | 358.79 |

Fig. 3.7: Example 3: prediction results for testing and all historical data.

The data used in this example are generated from model (B.7). The experiment’s setting is exactly the same as that presented in term 3 in Appendix B.1. In a similar way to Example 2 in Section 3.3.4.3, LEOA also runs in an online mode. The prediction results for the testing data are displayed in the Table 3.7. It can be observed that LEOA can make better predictions than other approaches judged by RMSEs. However,
one should notice that LEOA has used 61 rules — a larger number than other methods which have been used to make comparisons. In order to observe the global optimal behavior of LEOA, the final system learned from the data has remained unchanged in both structure and parameters in order to make prediction of the whole 3300 data points and get the RMSE equals to 0.0620. This means that LEOA can remember its previous behaviour well whilst evolving its structure and updating its parameters, in applying the information extracted from the new data. Fig. 3.7 shows the prediction results for the 300 testing data points \(y_{3001}, y_{3002}, \ldots, y_{3300}\), and the estimation results for using the final model to estimate all the historical data. Due to there is a limited space to present the prediction results for all the 3300 historical data, only the prediction results for the first 300 data points \(y_1, y_2, \ldots, y_{300}\) are shown in Fig. 3.7 as examples. Fig. 3.7 gives a visible explanation of the learning ability of LEOA. Because there are 61 final rules learned by LEOA in this example, only the first two rules are listed below to show the interpretability of LEOA:

\[
R_1: \text{If } x_1 \text{ is } \Gamma_{1,1} \text{ and } x_2 \text{ is } \Gamma_{1,2} \text{ and, } \ldots, \text{ and } x_{11} \text{ is } \Gamma_{1,11}, \nonumber
\]

\[
\text{Then } y_1 = -0.2012 + \sum_{i=1}^{11} \psi_{1,i} x_i; \quad (3.97)
\]

\[
R_2: \text{If } x_1 \text{ is } \Gamma_{2,1} \text{ and } x_2 \text{ is } \Gamma_{2,2} \text{ and, } \ldots, \text{ and } x_{11} \text{ is } \Gamma_{2,11}, \nonumber
\]

\[
\text{Then } y_2 = -0.2828 + \sum_{i=1}^{11} \psi_{2,i} x_i. \quad (3.98)
\]

The parameters in (3.97) and (3.98) are:

Centre of \(R_1\) is: \(c_1 = [-0.1030, 0.0000, 0, 0, 0, 0, 0, 0, 0, 0, -0.2989]\) \(\quad (3.99)\)

Centre of \(R_2\) is: \(c_2 = [1.7286, 1.6725, 1.5576, 1.3974, 1.2168, 1.0006, 0.7632, 0.5513,\nonumber

\[0.4097, 0.3607, 1.0000]\) \(\quad (3.100)\)

Radius of \(R_1\) is: \(\sigma_1 = [0.1081, 0.0833, 0.0833, 0.0833, 0.0833, 0.0833, 0.0833, 0.0833, \nonumber

\[0.0833, 0.0833, 0.1559]\) \(\quad (3.101)\)

Radius of \(R_2\) is: \(\sigma_2 = 10^{-3}[0.0024, 0.0797, 0.0636, 0.2440, 0.1360, 0.5801, 0.0988,\nonumber

\[0.0392, 0.0006, 0.0058, 0.0182]\) \(\quad (3.102)\)

Consequent parameter of \(R_1\) is: \(\psi_1 = [-0.2012, 0.3218, 0.0825, 0.0717, 0.1052, 0.1396,\nonumber

\[0.1515, 0.1303, 0.0739, -0.0149, -0.1286, 0.1589]\) \(\quad (3.103)\)

Consequent parameter of \(R_2\) is: \(\psi_2 = [-0.2828, 0.4343, 0.2053, 0.1277, 0.0988, 0.0827,\nonumber

\[11\sum_{i=1}^{11} \psi_{1,i} x_i; \quad (3.97)
\]

\[11\sum_{i=1}^{11} \psi_{2,i} x_i. \quad (3.98)
\]
3.3.4.5 Example 4: Mackey-Glass Chaotic Time Series (term 5 in the Appendix B.1)

This experiment follows from the steps described in term 5 in Appendix B.1. LEOA is compared with DENFIS [26], eTS+ [29], Simple_eTS+ [37], GENEFIS [56], PANFIS [55], eT2RFNN [89], GSETSK [69], SPLAFIS [96] and DeTS [35]. Performances are judged by NDEIs. Fig. 3.8 depicts the error trace of LEOA whilst running online on the training dataset. From Fig. 3.8, it can be seen that the absolute values of the prediction errors decrease from nearly 0.3 to about 0.1 after 1500 training steps. Furthermore, the errors vary approximately between $-0.1$ and 0.1 from 1501 to 3000 training steps. Table 3.8 summarizes the numerical results of all the state-of-the-art approaches and LEOA. The result of LEOA presented in Table 3.8 is obtained by choosing the thresholds as $\varepsilon_0 = 0.1$ and $M = 50$. Furthermore, the final model learned from the whole dataset has been used to make estimations of all the 3500 historical data with neither fuzzy rule nor parameters updating. Numerical result reports that LEOA can present its characteristics of global optimality with NDEI = 0.2423.

As can be seen from Table 3.8, LEOA has achieved the best accuracy. However, LEOA has applied more fuzzy rules to make predictions and has required a longer execution time than many of the other approaches. This phenomenon is determined by the designing viewpoint and structure of LEOA. LEOA is always trying to keep more historical information in reserve for optimality while achieving a high testing accuracy. In a similar way to Sections 3.3.4.5 and 3.3.4.5, the first two fuzzy rules of the final 42 rules learned by LEOA are listed in (3.105) and (3.106), in order to show that LEOA can be represented using these rules.

$$R_1 : \text{If } x_1 \text{ is } \Gamma_{1,1}, \text{ and } x_2 \text{ is } \Gamma_{1,2}, \text{ and } x_3 \text{ is } \Gamma_{1,3}, \text{ and } x_4 \text{ is } \Gamma_{1,4},$$

Then $y_1 = 0.1889 - 0.1596x_1 + 0.0764x_2 + 0.3101x_3 + 0.5402x_4$; \hspace{1cm} (3.105)

$$R_2 : \text{If } x_1 \text{ is } \Gamma_{2,1}, \text{ and } x_2 \text{ is } \Gamma_{2,2}, \text{ and } x_3 \text{ is } \Gamma_{2,3}, \text{ and } x_4 \text{ is } \Gamma_{2,4},$$

Then $y_2 = 0.1987 - 0.1323x_1 + 0.0704x_2 + 0.3355x_3 + 0.5746x_4$. \hspace{1cm} (3.106)

The corresponding centres and radii of $R_1$ and $R_2$ are: $c_1 = [1.0518, 0.8295, 0.6600, 0.7683]$ and $\sigma_1 = [0.0058, 0.0182, 0.0133, 0.0127]$; $c_2 = [1.1553, 1.2740, 1.1772, 0.8069]$ and $\sigma_2 =$
10^{-4}[0.0005, 0.3050, 0.0701, 0.0851].

<table>
<thead>
<tr>
<th>Method</th>
<th>Rule Type</th>
<th>RMSEs</th>
<th>Rule Num.</th>
<th>Parameter</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DENFIS [26]</td>
<td>T-S</td>
<td>0.278</td>
<td>58</td>
<td>886</td>
<td>–</td>
</tr>
<tr>
<td>eTS+ [29]</td>
<td>T-S</td>
<td>0.392</td>
<td>10</td>
<td>130</td>
<td>–</td>
</tr>
<tr>
<td>Simple eTS+ [37]</td>
<td>T-S</td>
<td>0.375</td>
<td>15</td>
<td>150</td>
<td>–</td>
</tr>
<tr>
<td>GENEFIS(C) [56]</td>
<td>T-S</td>
<td>0.280</td>
<td>19</td>
<td>475</td>
<td>4.46</td>
</tr>
<tr>
<td>GENEFIS(B) [56]</td>
<td>T-S</td>
<td>0.339</td>
<td>9</td>
<td>225</td>
<td>3.02</td>
</tr>
<tr>
<td>PANFIS [55]</td>
<td>T-S</td>
<td>0.301</td>
<td>19</td>
<td>475</td>
<td>4.5</td>
</tr>
<tr>
<td>eT2RFNN [89]</td>
<td>T-S</td>
<td>0.32</td>
<td>3</td>
<td>108</td>
<td>–</td>
</tr>
<tr>
<td>GSE STSK [69]</td>
<td>T-S</td>
<td>0.347</td>
<td>19</td>
<td>247</td>
<td>–</td>
</tr>
<tr>
<td>SPLAFIS [96]</td>
<td>T-S</td>
<td>0.279</td>
<td>30</td>
<td>390</td>
<td>–</td>
</tr>
<tr>
<td>DeTS [35]</td>
<td>T-S</td>
<td>0.440</td>
<td>3</td>
<td>39</td>
<td>–</td>
</tr>
<tr>
<td>LEOA</td>
<td>T-S</td>
<td>0.2480</td>
<td>42</td>
<td>546</td>
<td>144.78</td>
</tr>
</tbody>
</table>

Example 4: error trace.

Fig. 3.8: Example 4: error trace.

3.4 Chapter Summary

Mamdani and T-S fuzzy rules have been considered, and the corresponding learning approaches EMFSPO and LEOA have been proposed from an optimization viewpoint:
all the structure and parameters learning approaches have been induced backwards to make sure that the estimated system is the optimal solution for minimizing the local error functions. Furthermore, an EWRLS method has been derived from the induction steps. LEOA has been proven to satisfy the $\varepsilon$-completeness property. Although LEOA has been induced to reach the local optimal, employed from the numerical results; LEOA has also been found to be able to capture the global behaviour of the data. The feasibility and accuracy of the two approaches are validated by both artificial and real-world datasets across both system identification and time series prediction areas. However, EMFSPO and LEOA have been tested to be, on the whole, more complex than many of the other state-of-the-art methods. As a result, future work should be carried out in order to reserve accuracy and optimality, while using a more simple and interpretable model.
Chapter 4

Evolving Fuzzy Systems: Self-Learning Approaches for Dynamic Thresholds

4.1 Chapter Introduction

In the previous chapter, Chapter 3, it concentrated on the optimality problems of EFSs online learning approaches. Due to that the proposed local optimal approaches in Chapter 3 are usually more complex than the traditional suboptimal approaches, in this chapter, the traditional suboptimal approaches that can process the data swiftly is investigated. As mentioned in term (3) in Section 2.2.4, automatic tuning problem of the thresholds that controls system evolving remains unsolved. Furthermore, term (4) in Section 2.2.4, demonstrated that existing geometric similarity measures, which used in rule merging, are heuristic and inaccurate. In order to cope with these shortcomings, two EFS online learning methods is proposed. These methods treats the thresholds as dynamical parameters which are varying with the evolution of systems being learnt. By utilizing the online training errors as an indicator to reflect the underfitting and overfitting risks of an EFS, the proposed approaches adjust the values of threshold parameters automatically and dynamically. Furthermore, an accurate $L^2$-based geometric similarity measure is proposed.

The first work is SEFS. Unlike the fixed thresholds commonly used in EFSs, SEFS uses online training errors, which measures the quality of an identified model, to set/tune a dynamic threshold automatically for rule generation. This self-tuning parameter,
which controls the speed and coverage for fuzzy rule generation relying on the underfitting/overfitting risks indicated by the online training errors: (1) Large training errors present an underfitted model, which is too coarse to represent the highly complicated and rapidly dynamic (e.g. highly nonlinear, nonstationary) behaviour of the data segment. Then, finer rules need to be generated; and (2) Tiny training errors maybe reflect a high risk of an overfitted model for noisy data. If this happens, then coarse rule base should be used. Besides, an $L^2$-distance-based geometric similarity measure is proposed. With this similarity measure, the SEFS computes the similarity between Gaussian membership functions accurately without making an approximation of the Gaussian membership function beforehand. In addition, a WRLS method with a variable forgetting factor (VFF-WRLS), which minimizes the mean square of the noise-free posterior error signal, is applied to learn the consequent parameters.

Depending on SEFS, an improved work, EFS-SLAT, has been done. EFS-SLAT enables the thresholds, which control both rule generation and simplification, to be learnt and updated online. The thresholds used in EFS-SLAT are designed to make the rule generation and simplification speed restrict each other depending on the risk of underfitting and overfitting indicated by the online training errors. Furthermore, EFS-SLAT also has achievement made on the $L^2$-distance-based geometric similarity measure for avoiding inappropriate rule merging. Several benchmark examples across both artificial and real-life datasets can verify that SEFS and EFS-SLAT have the ability to give better performance compared with many state-of-the-art approaches.

As the same identification problem is investigated by SEFS and EFS-SLAT, this common problem for these two approaches is presented together in a separated section, Section 4.2. Section 4.3 presents the SEFS. The improved work, EFS-SLAT, is presented in Section 4.4. This chapter is summarized in Section 4.5. \footnote{Sections 4.3 and 4.4 are based on our published work [199] and the submitted manuscript [200], respectively.}

\section{Problem Statement}

Both SEFS and EFS-SLAT are the approaches to learn the EFS with T-S fuzzy rules. In a similar way to Chapter 3, $R_i$ stands for the $i$-th fuzzy rule. The form of rule $R_i$ can
be shown as:

\[ R_i : \text{If } x_1 \text{ is } \Gamma_{i,1} \text{ and } x_2 \text{ is } \Gamma_{i,2} \text{ and } \ldots \text{ and } x_n \text{ is } \Gamma_{i,n}, \text{ then } y_i = \psi_{i,0} + \sum_{j=1}^{n} \psi_{i,j} x_j, \quad (4.1) \]

where \( i = 1, 2, \ldots, K \), \( K \) is the number of fuzzy rules, \( x = (x_1, x_2, \ldots, x_n) \), in which \( x_j \) is the \( j \)-th input, \( x \in \Omega = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n] \subset \mathbb{R}^n \), \( y_i \) is the output of rule \( R_i \), \( \psi_i = (\psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,n}) \) is the vector of consequent parameters, and \( n \) is the dimension of the input vectors.

The membership function of \( \Gamma_{i,j} \) is \( \mu_{i,j}(x_j) \). It is a Gaussian membership function:

\[ \mu_{i,j}(x_j) = \exp\left\{ -\frac{(x_j - c_{i,j})^2}{2(\sigma_{i,j})^2} \right\}, \quad (4.2) \]

in which \( c_{i,j} \) and \( \sigma_{i,j} \) are cluster centre and radius, respectively. Furthermore, for rule \( R_i \), the firing strength \( \gamma_i(x) \) and the normalized firing strength \( \theta_i(x) \) are presented by (4.3) and (4.4), respectively:

\[ \gamma_i(x) = \prod_{j=1}^{n} \mu_{i,j}(x_j), \quad (4.3) \]
\[ \theta_i(x) = \frac{\gamma_i(x)}{\sum_{j=1}^{K} \gamma_j(x)}, \quad (4.4) \]

and the final output of the system can be computed by (4.5):

\[ y = \sum_{i=1}^{K} \theta_i(x) y_i. \quad (4.5) \]

The remainder of this chapter is going to present SEFS and EFS-SLAT that identify the evolving T-S fuzzy system from three aspects: rule number \( K \), antecedent parameters \( c_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,n}) \) and \( \sigma_i = (\sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,n}) \), and consequent parameters \( \psi_i = (\psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,n})^T \), where \( i = 1, 2, \ldots, K \). For the convenience to present, some assumptions should be made: the \( t \)-th real input and corresponding output are \( x_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,n}) \) and \( y_t \), respectively; \( \hat{y}_t^{\text{train}} \) is the estimated training output; and \( \hat{y}_t \) stands for the estimated testing output.
4.3 A Self-Learning Approach with Dynamic Rule Generation Threshold

This section will present the technique details of SEFS. The main content of this section is based on our published approach in [199]. The major highlights of SEFS can be summarized as: (1) The threshold that controls rule generation is designed to be varying along with the online training errors, which expands the threshold from a fixed value to a value dynamically changing in a certain interval. (2) A new geometric similarity measure is proposed, in order to measure the $L^2$ distance between the Gaussian membership functions directly. (3) The local version of the robust approach, recursive least square method with a variable forgetting factor (VFF-RLS), is derived and applied in learning the consequent parameters.

The rest of this section is arranged as follows. Section 4.3.1 explains the learning details of the SEFS in fuzzy rule generation and merging, and antecedent and consequent parameters updating. Numerical examples used to evaluate the SEFS are displayed in Section 4.3.2.

4.3.1 Incremental Learning Algorithm of SEFS

In order to solve the evolving T-S system identification problem presented in Section 4.2, SEFS is proposed with three modules: rule generation and updating, rule merging and consequent parameters learning. All these three learning modules are presented in this section.

4.3.1.1 Fuzzy Rule Generation and Updating

The most widely used criteria to judge whether there is a need to generate a new fuzzy rule is: “If $\forall i = 1, 2, \ldots, K, \gamma_i(x_t) < \epsilon$ holds, then a new rule should be generated” (see (2.16) in Section 2.2.3.1. Most of the previous research works set $\epsilon$ as a fixed-value threshold, including EMFSPO and LEOA that have been presented in Chapter 3. Based on these existing works, this section makes the improvement on exploring the method that enables to set the threshold to be a time-varying and a self-tuning one as $\epsilon_t$. In this way, the learning approach can capture the dynamics of the data stream, and adjusts its
value by self-learning to follow these dynamics. As the online training errors contain the risks of underfitting and overfitting of the EFS, the threshold $e_t$ is designed to be the function of the online training errors. Besides, due to the fact that the most recent data always have higher influence on the future behaviour of the data stream than the old data, it is natural and reasonable that newer training errors should have more influence on the threshold than the older ones. Furthermore, on the one hand, when the cumulative online training error is big, the learnt EFS is too coarse to catch up the nonlinearity and nonstationary of the data stream, then, more fuzzy rules are required to overcome the underfitting problem. In this case, a big threshold $e_t$ enables the EFS to evolve rapidly to increase the accuracy. On the other hand, small (or tiny) cumulative online training errors demonstrate that the EFS is complex (or over complicated) to approximate the data stream. In this situation, increasing the rule numbers in a high speed is likely to cause overfitting; thus, a small threshold $e^{(a)}_t$ should be set to slow down the rule generation speed and avoid overfitting. As a result, in order to precisely depict the phenomenon of the variation of the threshold along with the ability of the EFS to fit the dynamics of the data stream, $e^{(a)}_t$ (4.6) is designed as the monotonic increasing function of the cumulative online training error $\sum_{k=1}^{t} \lambda^{t-k} e_k$. Because the direct use of the online training errors $y_k - \hat{y}_{train}^k$ to compute the cumulative error will lead to the positive and negative values compensate each other, therefore, the absolute online training errors $e_k = |y_k - \hat{y}_{train}^k|$ are applied as:

$$
\begin{align*}
\epsilon^{(a)}_t &= \epsilon_{max} - (\epsilon_{max} - \epsilon_{min})E_t, \text{ where} \\
E_t &= \exp\{-\sum_{k=1}^{t} \lambda^{t-k} e_k\},
\end{align*}
$$

$\lambda \in [\lambda_0, 1)$ is the forgetting factor for indicating the importance of each absolute online training error, and the interval $[\lambda_0, 1)$ is the admissible forgetting factor interval. Smaller $\lambda$ permits faster forgetting of the old errors and put more focus on information contained in the recent errors. The lower and upper bounds of the threshold $\epsilon_t$ are $\epsilon_{min}$ and $\epsilon_{max}$, respectively. Assume $E_t = \exp\{-A(t)\}$ and $A(t) = \sum_{k=1}^{t} \lambda^{t-k} e_k$; then $A(t+1)$ can be updated by $A(t+1) = \lambda A(t) + e_{t+1}$.

**Proposition 4.1.** $\epsilon_t$ is monotonically increasing against $\sum_{k=1}^{t} \lambda^{t-k} e_k$.

**Proof.** See Section A.2.1, Appendix A.
CHAPTER 4. EVOLVING FUZZY SYSTEMS: SELF-LEARNING APPROACHES FOR DYNAMIC THRESHOLDS

Assume the new coming data pair is \((x_{t+1}, y_{t+1})\) at time \(t + 1\). The rule generation method of SEFS is as follows:

**Rule adding method.** If \(\gamma_i(x_{t+1}) < \varepsilon^{(a)}_{t+1}, \forall i = 1, 2, \ldots, K\), then generate a new fuzzy rule with cluster centre \(x_{t+1}\). Radius is \(\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)\) with \(\sigma_j = \frac{||c_j - x_{t+1}||_2 - 2\log(\varepsilon^{(a)}_{t+1})}{k_0} (k_0 = \sqrt{-2\log(\varepsilon^{(a)}_{t+1})}, \ j = 1, 2, \ldots, n)\), in which \(i^* = \arg\max_i \gamma_i(x_{t+1})\). \(\varepsilon^{(a)}_{t+1}\) is a time varying threshold with the function form shown in (4.6).\(^2\)

The rule generation method helps determine the radius of the new cluster by ensuring that \(c_{i^*}\), which is the nearest cluster centre of \(x_{t+1}\), satisfies \(\exp\{ -\sum_{j=1}^n \frac{(c_{j^*} - x_{t+1})^2}{2\sigma_j^2}\} = \varepsilon^{(a)}_{t+1}\). According to the bounds of \(\varepsilon^{(a)}_{t+1}\), the corresponding dynamic parameter \(k_0 = k_{t+1} = \sqrt{-2\log(\varepsilon^{(a)}_{t+1})}\) is in the interval \([\sqrt{-2\log(\varepsilon_{\max})}, \sqrt{-2\log(\varepsilon_{\min})}]\). The form of \(k_{t+1}\) in the rule generation method is the generalized form of which in [55,78] and [56].

The radius setting method given by the rule generation method is applicable when \(K > 0\). However, at the very beginning of SEFS leaning process, the first input \(x_1\) is set as the centre of the first rule, and the radius can not be set by the rule generation method. In order to deal with this, the original radius of the first cluster is set as 0 until the second input \(x_2\) comes. A new rule should be built with centre \(x_2\), and the radii of the first and second cluster are set as \(\sigma_{1,j} = \sigma_{2,j} = \frac{\sqrt{n}}{k_0}||c_1 - c_2||_2 (k_1 = \sqrt{-2\log(\varepsilon^{(a)}_{1})}), \ j = 1, 2, \ldots, n\). If the rule generation method is not satisfied, then the SEFS updates the cluster centre and radius of rule \(R_{i^*}\) according to the rule updating method; the formulas are based on the sample mean and variance shown in (2.43) and (2.44) in Section 2.2.3.1.2.A.

**Rule updating method.** If \(\gamma_i(x_{t+1}) \geq \varepsilon^{(a)}_{t+1}\), then \(c_{i^*}\) and \(\sigma_{i^*}\) should be updated by the following recursive formulas:

\[
c_{i^*,j}(t+1) = c_{i^*,j}(t) + \frac{x_{t+1,j} - c_{i^*,j}(t)}{N_{i^*}(t)}\frac{1}{N_{i^*}(t)+1}, \tag{4.8}
\]

\[
(\sigma_{i^*,j}(t+1))^2 = (\sigma_{i^*,j}(t))^2 + \frac{N_{i^*}(t)}{N_{i^*}(t) + 1} \left(\frac{(x_{t+1,j} - c_{i^*,j}(t+1))^2}{} - (\sigma_{i^*,j}(t))^2 + N_{i^*}(t)(c_{i^*,j}(t+1) - c_{i^*,j}(t))^2\right)\frac{1}{N_{i^*}(t) + 1}, \tag{4.9}
\]

where \(N_{i^*}(t + 1)\) is the number of inputs with firing strengths larger than \(\varepsilon^{(a)}_{t+1}\), \(j = \ldots\)

---

\(^2k_0\) is obtained by \(\varepsilon^{(a)}_{t+1} = \exp\{-(k_0)^2/2\}\). \(\|\cdot\|_2\) is the Euclidean norm.
4.3.1.2 Fuzzy Rule Merging

Fuzzy rules with similar antecedent parts but different consequent parts are very likely to cause rule confliction. Merging fuzzy rules with similar antecedent parts is an effective approach to get rid of rule confliction and redundancy. More importantly, in the context of online learning, rule generation without merging will lead to the rule number keeping increase and result in an unnecessarily complicated fuzzy system. Therefore, it is crucial to make judgement of the similarity between the fuzzy rules and then merge the similar rules. As summarized in the introduction, there are many existing works on both set-theoretic and geometric similarity measures. Set-theoretic similarity measures require to compute the intersection and the union of the fuzzy sets. Due to the nonlinearity of the Gaussian membership function, it is difficult to compute the set-theoretic similarity measures\[^99\],[^51]. Therefore, the commonly used approach is to use the piecewise linear approximation of the Gaussian membership function, and two typical examples to make this approximation are to use triangle\[^51\] or trapezoidal\[^99\] membership functions. There have been a small number of research studies in recent years, for instance,\[^50\], attempted to compute the intersection and union of the fuzzy sets using Gaussian membership function directly, without making any approximations in advance. However, this method is computationally expensive, and under the risk of the curse of dimensionality. Compared with set-theoretic similarity measures, geometric measures are easier to compute and widely used in many existing works of the EFS. The most widely used geometric measure is based on the distance between the parameters of the membership functions, and the examples can be found from\[^30,55,56\] and\[^89\]. However, it is still hard to make the accurate judgement for whether the distance between the firing strengths is small from only comparing the distance between the antecedent parameters. As a result, no matter set-theoretic or geometric, similarity measures cannot measure the distance between the firing strengths (or Gaussian membership functions)

\[^3\]N_i for the rule R_i is regarded as the number of inputs, which determine the real position and shape of the corresponding cluster of rule R_i. N_i should be recorded from the time that the rule R_i is built. Once rule updating method holds for the rule R_i, then N_i(t+1) = N_i(t) + 1.

\[^4\]The induction of formulas (4.8) and (4.9) is presented in Section A.2.2, Appendix A.
directly and accurately. In addition, to our best knowledge, there is no analytic form to compute the distance between the firing strengths computed using Gaussian membership functions presented in the existing research works.

In the SEFS, an analytic form of the similarity measure between the firing strengths based on the $L^2$ distance is proposed, which is a direct and accurate method of computing the similarity. Furthermore, this new similarity measure can be worked out very fast. Follow from the formal definition of the geometric similarity measure presented in [97], the similarity measure for rule $R_1$ and $R_2$ can be presented by $S(R_1, R_2) = \frac{1}{1 + D(R_1, R_2)}$, where $D(R_1, R_2)$ is the $L^2$ distance between the firing strengths of rule $R_1$ and $R_2$. Set $D(\cdot, \cdot)$ to be the $L^2$ distance; The Definition 4.1 shows how $S(\cdot, \cdot)$ is computed.

**Definition 4.1.** Assume that $\gamma_{l1}$ and $\gamma_{l2}$ are the firing strengths of rules $R_{l1}$ and $R_{l2}$, respectively. The similarity between these two fuzzy rules is defined as $S(R_{l1}, R_{l2})$ presented as follows:

$$S(R_{l1}, R_{l2}) = \frac{1}{1 + \|\gamma_{l1} - \gamma_{l2}\|_{L^2}},$$

(4.10)

where $\| \cdot \|_{L^2}$ is the $L^2$ norm.

The following part of this section will develop the method on how to compute the $L^2$ distance $\|\gamma_{l1} - \gamma_{l2}\|_{L^2}$ fast and precisely.

The $L^2$ distance in (4.10) can be represented as:

$$\|\gamma_{l1} - \gamma_{l2}\|_{L^2}^2 = \int_{\Omega} \left| \gamma_{l1}(x) - \gamma_{l2}(x) \right|^2 dx$$

$$= \int_{a_1}^{b_1} \cdots \int_{a_n}^{b_n} \left[ \prod_{j=1}^{n} \mu_{l1,j}(x_j) - \prod_{j=1}^{n} \mu_{l2,j}(x_j) \right]^2 dx_1 \cdots dx_n$$

$$= \prod_{j=1}^{n} \int_{a_j}^{b_j} [\mu_{l1,j}(x_j)]^2 dx_j + \prod_{j=1}^{n} \int_{a_j}^{b_j} [\mu_{l2,j}(x_j)]^2 dx_j - 2 \prod_{j=1}^{n} \int_{a_j}^{b_j} [\mu_{l1,j}(x_j)\mu_{l2,j}(x_j)] dx_j,$$

(4.11)

$\mu_{l1,j}$ and $\mu_{l2,j}$ are both Gaussian membership functions defined by (4.2). In the following parts, those three terms in (4.11) are calculated separately.
Let \( t_j = \frac{x_j - c_{l_1,j}}{\sigma_{l_1,j}} \), \( a_j = \sigma_{l_1,j} a_{l_1,j} + c_{l_1,j} \), \( b_j = \sigma_{l_1,j} b_{l_1,j} + c_{l_1,j} \), then it can obtain:

\[
\prod_{j=1}^{n} \int_{a_j}^{b_j} [\mu_{l_1,j}(x_j)]^2 dx_j = \prod_{j=1}^{n} \int_{a_j}^{b_j} \exp\left\{ -\frac{(x_j - c_{l_1,j})^2}{\sigma_{l_1,j}} \right\} dx_j \\
= \prod_{j=1}^{n} \sigma_{l_1,j} \int_{a_j}^{b_j} \exp\left\{ -t_j^2 \right\} dt_j \\
= \left( \frac{1}{2} \sqrt{\pi} \right)^n \prod_{j=1}^{n} \sigma_{l_1,j} G(a_{l_1,j}, b_{l_1,j}).
\] (4.12)

Function \( G(a, b) \) has the form shown as follows:

\[
G(a, b) = \begin{cases} 
\text{erf}(b) - \text{erf}(a) & 0 \leq a \leq b \\
\text{erf}(-a) + \text{erf}(b) & a \leq 0 \leq b \\
\text{erf}(-a) - \text{erf}(-b) & a \leq b \leq 0,
\end{cases}
\] (4.13)

where \( \text{erf}(\cdot) \) is the error function expressed as:

\[
\text{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} \exp\{-t^2\} dt = \frac{2}{\sqrt{\pi}} \int_{0}^{x} \exp\{-t^2\} dt.
\] (4.14)

The error function can be calculated by some very simple formulas with high accuracy. Two simple examples taken from [201] to compute \( \text{erf}(\cdot) \) are listed in (4.15) and (4.16) as follows:

\[
\text{erf}(x) \approx 1 - \frac{1}{\left(1 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 \right)^4},
\] (4.15)

where \( a_1 = 0.278393 \), \( a_2 = 0.230389 \), \( a_3 = 0.000972 \), \( a_4 = 0.078108 \), and

\[
\text{erf}(x) \approx 1 - (a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5) \exp\{-x^2\},
\] (4.16)

in which \( t = \frac{1}{1 + px} \), \( p = 0.3275911 \), \( a_1 = 0.254829592 \), \( a_2 = -0.284496736 \), \( a_3 = 1.421413741 \), \( a_4 = -1.453152027 \), and \( a_5 = 1.061405429 \). The maximum error obtained by (4.15) and (4.16) are 5 \times 10^{-4} and 1.5 \times 10^{-7}, respectively. In this thesis, the MATLAB built-in function \( \text{erf}(x) \), which is a rational function approximation shown in [202], is used to approximate the error function. In a similar way to the first term of
It can be seen from (4.19), the computation speed is decided by \( \exp(x) \) function, which is caused by the rational function approximation of the \( \text{erf}(x) \). This rational function approximation contains the multiplication of a polynomial of the 8th (or lower than 8th) degree and an exponential function, and achieves maximal relative errors ranging down to between \( 6 \times 10^{-19} \) and \( 3 \times 10^{-20} \) [202]. Because it is faster to compute \( \exp(x) \) for just a few times than a vast amount of times required by the multiple numerical integral, it is realistic and appropriate to use (4.19) to compute the similarity \( S(R_{l_1}, R_{l_2}) \) in online computing. Based on (4.19), the rule merging method of SEFS can be constructed as follows:

**Rule merging method.** If rule \( R_f \) is updated following the rule updating method, and there exist rules \( R_{l_1}, \ldots, R_{l_M} \) such that \( S(R_f, R_{l_k}) > \varepsilon^{(m)} (k = 1, 2, \ldots, M) \), then merge
rule $R_{i^*}$ with $R_{i_k^*}$ (where $i_k^* = \arg \max_i S(R_{i'}, R_{i_k})$) to one rule $R_{i'}$. The centre and radius of rule $R_{i'}$ are $c_{i'}$ and $\sigma_{i'}$ calculated by (4.20) and (4.21), respectively. The consequent parameters of rule $R_{i'}$ can be computed by $\psi_{i'} = \frac{N_{i'} \psi_{i*} + N_{i_k} \psi_{i_k*}}{N_{i'} + N_{i_k}}$. Remove the original two rule $R_{i_k^*}$, and then replace $R_{i'}$ by $R_{i_k^*}$. (Each time when two rules are merged, then the number of rules $K$ should be updated by $K - 1$.) Repeat the above whole process until there is no fuzzy rules, which have the similarity degree with rule $i^*$ larger than the threshold.

The rule merging method illuminates that once there exists a rule $R_{i'}$ being updated, then there is a need to consider whether it evolves similar to any other rules. It is likely that more than one fuzzy rule have the similarity degree with $R_{i^*}$ larger than the threshold. In this case, it is plausible to merge $R_{i'}$ with the rule, which has the largest similarity degree with rule $R_{i^*}$. If the fuzzy rule that has the largest similarity degree with rule $R_{i^*}$ is not unique, then merge the earliest built rule with the rule $R_{i^*}$. The formulas for computing the new cluster centre and radius are displayed by (4.20) and (4.21), respectively. Similar methodology can also be found in [39].

Additionally, when an EFS evolves as the rule generation method, the rule updating method and the rule merging method, then this fuzzy system satisfies the $\varepsilon$-completeness property, the definition of which is shown in the Definition 3.1 (see Section 3.3.2.1.1). The $\varepsilon$-completeness of SEFS is presented in Theorem 4.1.

**Theorem 4.1.** ($\varepsilon$-completeness of SEFS) Assume that the input space $\Omega$ is a bounded and closed subset of $\mathbb{R}^n$.

(1) Rule generation: Assume that the existing cluster centres and radii are $c_i$ and $\sigma_i$ with $i = 1, 2, \ldots, N'_0$. Furthermore, assume that fuzzy rules are generated according to the rule generation method; then, $\forall x \in \Omega$, $\exists i$ and $K < \infty$ ($i \in \{1, 2, \ldots, K\}$) such that the firing strength of $x$ satisfies $\gamma_i(x) \geq \varepsilon_{min}$.

\[ c_{i'} = \frac{N_{i'} c_{i'} + N_{i_k} c_{i_k^*}}{N_{i'} + N_{i_k}}, \quad (4.20) \]
\[ (\sigma_{i', j})^2 = \frac{N_{i'} \sigma_{i', j}^2 + N_{i_k} \sigma_{i_k^*, j}^2 + N_{i'} (c_{i'} - c_{i_k^*})^2}{N_{i'} + N_{i_k}} + \frac{N_{i_k} (c_{i'} - c_{i_k^*})^2}{N_{i'} + N_{i_k}}, \quad (4.21) \]

where $N_{i'}$ and $N_{i_k^*}$ are defined the same as the rule updating method, and $j = 1, 2, \ldots, n$.\(^5\)

\(^5\)The induction of formulas (4.20) and (4.21) is presented in Section A.2.2, Appendix A.
(2) Rule updating: Assume $\exists K < \infty$ such that $\forall x \in \Omega$, $\exists i \in \{1, 2, \ldots, K\}$ with $\gamma_i(x) \geq \varepsilon_{\text{min}}$. Furthermore, assume the $i^\text{th}$ rule is updated by $x_{t+1}$. Then, for all $x \in B(c_1(t), k_{\text{min}} \sigma_1(t))$, it can be induced that $x \in B(c_1(t+1), k_{\text{min}} \sigma_1(t+1))$, $k_{\text{min}} = \sqrt{-2 \log \varepsilon_{\text{min}}}$ when $N_1 \rightarrow \infty$.

(3) Rule merging: Assume $\exists K < \infty$ such that $\forall x \in \Omega$, $\exists i \in \{1, 2, \ldots, K\}$ with $\gamma_i(x) \geq \varepsilon_{\text{min}}$. Furthermore, assume $\exists i_1, i_2 \in \{1, 2, \ldots, K\}$ such that $\| \gamma_{i_1}(x) - \gamma_{i_2}(x) \|_2 < \varepsilon$, and these two rules are merged to one rule $i_0$. Then, for all $x \in B(c_{i_1}(t), k_{\text{min}} \sigma_{i_1}(t)) \cup B(c_{i_2}(t), k_{\text{min}} \sigma_{i_2}(t))$, it can be induced that $x \in B(c_{i_0}(t+1), k_{\text{min}} \sigma_{i_0}(t+1))$ when $N_{i_1}, N_{i_2} \rightarrow \infty$.

Proof. They can be proven using the definition of the compact set, and the convergency of the centres and radii. Detailed proof can be found from Section A.2.4, Appendix A.

### 4.3.1.3 Consequent Learning

Consequent parameters are updated by the VFF-RLS proposed in [203]. The VFF-RLS is an improved version of the RLS method with better performance on tracking sudden system changes. The VFF-RLS has an optimal dynamic forgetting factor obtained from minimizing the mean-square noise-free posterior error, and usually has better performance in both stationary and nonstationary environments [203]. The local version of VFF-RLS, the weighted recursive least-squares algorithm with variable forgetting factor (VFF-WRLS), is computed by the aim of minimizing the error functions:

$$Err_i = \sum_{k=1}^{t+1} \theta_i(k) \lambda_i^{t+1-k} (y_k - x_e \psi_i)^2,$$

(4.22)

where $x_e = (1, x_k) = (1, x_{k,1}, x_{k,2}, \ldots, x_{k,n})$, $\theta_i(k) = \theta_i(x_k)$, $i = 1, 2, \ldots, K$. Assume that the new input is $x_{t+1} = (x_{t+1,1}, x_{t+1,2}, \ldots, x_{t+1,n})$, and the corresponding output is $y_{t+1}$. The WRLS updating formulas$^6$ of $\psi_i$ are:

$$\psi_i(t+1) = \psi_i(t) + \frac{\theta_i(t+1) P_i(t) x_{e+1} (y_{t+1} - x_e \psi_i(t))}{\lambda_i(t+1) + \theta_i(t+1) x_e^T P_i(t) x_e^{t+1}},$$

(4.23)

$$P_i(t+1) = \frac{1}{\lambda_i(t+1)} (P_i(t) - \frac{\theta_i(t) P_i(t) x_{e+1} x_{e+1}^T P_i(t)}{\lambda_i(t+1) + \theta_i(t+1) x_e^T P_i(t) x_e^{t+1}}),$$

(4.24)

$^6$The induction of (4.23) and (4.24) is shown in Section A.2.5, Appendix A.
in which $\lambda_i(t+1)$ is the time-varying forgetting factor. Following [203], the initial value of $P_i(t+1)$ is set as a big unit matrix $P_i(0) = M_0 * I_{(n+1) \times (n+1)}$ ($M_0$ is a big number). Furthermore, $\lambda_i(t+1)$ is given in (4.25), which is the optimum forgetting factor obtained by minimizing $E[(e'_i(t+1))^2]$ where $e'_i(t+1) = \theta_i(t+1)[xe_{t+1}h(t) - xe_{t+1}\psi_i(t+1)]$ and $h(t)$ are the noise-free posterior error and the impulse response, respectively. Furthermore, [203] stated that the optimum $\lambda_i(t+1)$ can be computed as:

$$
\lambda_i(t+1) \approx 1 - \frac{2E[e'_i(t+1)^2]}{M(E[e'_i(t+1)^2] + \sigma_v^2)},
$$

(4.25)

where $M$ is a big value, $E[e'_i(t+1)^2]$ is the excess mean square error (EMSE), $e_i(t+1) = \theta_i(t+1)[y_{t+1} - xe_{t+1}\psi_i(t)] = e'_i(t+1) + \theta_i(t+1)v_{t+1}$ is the prior error signal, $\sigma_v^2$ is the variance of Gaussian noise signal $v_{t+1}$, $y_{t+1} = xe_{t+1}h(t) + v_{t+1}$ and $e'_i(t+1) = \theta_i(t+1)[xe_{t+1}h(t) - xe_{t+1}\psi_i(t)]$. The noise-free prior error signal $e'_i(t+1)$ is estimated by the noniterative shrinkage method. This method has been used in [204] and [205] to deal with the image denoising problem. This method recovers a noise-free signal $\rho_f$ from a noisy signal $\rho = \rho_f + \epsilon$, in which $\epsilon$ is a white Gaussian noise signal, from solving the following $l_1 - l_2$ minimization problem:

$$
\min_{\rho_f} \alpha \|\rho_f\|_1 + 0.5 \|\Lambda \rho_f - \rho\|_2^2.
$$

(4.26)

In (4.26), $\alpha$ is the threshold, and $\Lambda$ is an orthogonal matrix. In the VFF-RLS algorithm, $\rho_f = e'_i(t+1)$, $\rho = e_i(t+1)$, and $\Lambda = 1$. Bhotto and Antoniou [203] have shown the optimal solution of (4.26) can be presented as:

$$
e'_i(t+1) = \text{sign}(e_i(t+1)) \max\{|e_i(t+1)| - \alpha, 0\}.
$$

(4.27)

The EMSE $E[e'_i(t+1)^2]$ is estimated by the time average of $(e'_i(t+1))^2$ and presented as:

$$
E[e'_i(t+1)^2] = \beta_2E[e'_i(t)^2] + (1 - \beta_2)(e'_i(t+1))^2.
$$

(4.28)

The threshold parameter $\alpha$ is taken as the time average of the variance of the white noise $v_{t+1}$. That is $\alpha = \sqrt{\beta_1 \sigma_v^2}$. As $y_{t+1} = xe_{t+1}h(t) + v_{t+1}$; then, the variance of $v_{t+1}$ can be updated by:

$$
\sigma_v^2 = \frac{t}{t+1} \sigma_v^2 + \frac{e_i(t+1)^2}{t+1}.
$$

(4.29)
The learnt EFS: $F(t)$.

1. **Read** $(x_{t+1}, y_{t+1})$, set $\text{ind}_a = 0$.

2. **Update** $c_{i^*}$, $(\sigma_{i^*})^2$.

3. **Add new rule** with centre $x_{t+1}$, $\text{ind}_a = 1$.

4. **Compute the prediction of** $y_{t+1}$: $\hat{y}_{t+1} = \sum_{i=1}^{K} \theta_i (x_{t+1}) x_{t+1} \psi_i(t)$.

5. **Update** the EFS by $F(t+1)$.

**Fig. 4.1:** The flowchart of SEFS.

### 4.3.1.4 Algorithm: SEFS

Based on the antecedent, consequent, and rule number learning methods in Section 4.3.1, the flowchart of SEFS algorithm is shown in Fig. 4.1. The indicator for fuzzy rule generation is presented as $\text{ind}_a$, the default value of which is 0. The online training result $\hat{y}_{t+1}^{\text{train}}$ is calculated by $\hat{y}_{t+1}^{\text{train}} = \sum_{i=1}^{K} \theta_i (x_{t+1}) x_{t+1} \psi_i(t+1)$. If a fuzzy rule is generated, then $\text{ind}_a = 1$. Furthermore, Fig. 4.1 presents the online learning and updating framework of the SEFS when $(x_{t+1}, y_{t+1})$ comes. In Fig. 4.1, $\epsilon_t^{(a)}$ is the self-learning threshold, while $\epsilon(t)$ that controls rule merging should be selected based on the data.
CHAPTER 4. EVOLVING FUZZY SYSTEMS: SELF-LEARNING APPROACHES FOR DYNAMIC THRESHOLDS

TABLE 4.1: Sensitivity analysis (Example 1 in section 4.3.2.2).

<table>
<thead>
<tr>
<th>( \varepsilon_{\text{max}} )</th>
<th>0.5</th>
<th>0.55</th>
<th>0.6</th>
<th>0.65</th>
<th>0.7</th>
<th>s=0.2600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Num.</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>RMSE</td>
<td>0.0474</td>
<td>0.0452</td>
<td>0.0428</td>
<td>0.0439</td>
<td>0.0506</td>
<td></td>
</tr>
<tr>
<td>NDEI</td>
<td>0.0459</td>
<td>0.0438</td>
<td>0.0414</td>
<td>0.0425</td>
<td>0.0490</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \varepsilon_{\text{min}} )</th>
<th>0.4</th>
<th>0.45</th>
<th>0.5</th>
<th>0.55</th>
<th>0.6</th>
<th>s=0.2744</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Num.</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>RMSE</td>
<td>0.0476</td>
<td>0.0458</td>
<td>0.0428</td>
<td>0.0461</td>
<td>0.0511</td>
<td></td>
</tr>
<tr>
<td>NDEI</td>
<td>0.0461</td>
<td>0.0444</td>
<td>0.0414</td>
<td>0.0447</td>
<td>0.0494</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \varepsilon^{(m)} )</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>s=0.3718</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Num.</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>RMSE</td>
<td>0.0500</td>
<td>0.0523</td>
<td>0.0428</td>
<td>0.0483</td>
<td>0.0487</td>
<td></td>
</tr>
<tr>
<td>NDEI</td>
<td>0.0484</td>
<td>0.0507</td>
<td>0.0414</td>
<td>0.0467</td>
<td>0.0471</td>
<td></td>
</tr>
</tbody>
</table>

The \( \varepsilon^{(m)} \) is suggested to be chosen in \([0.5, 1]\).

4.3.2 Numerical Results

Benchmark examples across nonlinear system identification, Mackey-Glass chaotic time series prediction and real nonstationary time series (S&P 500) prediction are shown in this section. These examples are applied to demonstrate that SEFS can effectively solve online regression problems. In these examples, the SEFS is running in an online mode. The numerical results are evaluated by RMSEs and NDEIs. Due to the upper and lower bound of \( \varepsilon \) and the threshold \( \varepsilon^{(m)} \) that judges the similarity are still need to be selected, sensitivity analysis of these parameters should be given before numerical examples.\(^7\)

4.3.2.1 Sensitivity Analysis

Thresholds \( \varepsilon_{\text{max}}, \varepsilon_{\text{min}} \), and \( \varepsilon^{(m)} \) \((\varepsilon_{\text{max}} \geq \varepsilon_{\text{min}})\) need to be identified beforehand. Inspired by \([79, 80, 206]\) the sensitivity \( s \) of each parameter \( \xi_i \) \((i=1,2,3,4 \text{ in this paper})\) is computed as:

\[
s = \frac{1}{\kappa(\pi_{\text{max}} - \pi_{\text{min}})} \sum_{j=1}^{\kappa-1} \left| \text{RMSE}(\xi_i^{(j+1)}) - \text{RMSE}(\xi_i^{(j)}) \right|, \tag{4.30}
\]

\(^7\)Numerical results of SEFS are calculated in the environment of intel(R) core (TM) i7-4790 CPU with a 3.6 GHz processor and a 16.0 GB memory.
where \( \kappa \) is the number of samples for a certain parameter (e.g., \( \varepsilon_{\text{max}}, \varepsilon_{\text{min}}, \varepsilon^{(m)} \)), \( \text{RMSE}(\xi_i) \) is the RMSE computed when using a certain parameter \( \xi_i \), and \( \pi_{\text{max}} \) and \( \pi_{\text{min}} \) are the upper and lower bounds of RMSEs. Parameters should be in \([0, 1]\) and neither be too big nor too small. In this example, \( \pi_{\text{max}} \) and \( \pi_{\text{min}} \) are set as the maximum and the minimum value of RMSEs obtained from all the different setting of parameters. In this example, \( \pi_{\text{max}} \) and \( \pi_{\text{min}} \) are 0.0523 and 0.0428, respectively. The thresholds \( \varepsilon_{\text{max}} \) and \( \varepsilon_{\text{min}} \) are geared at \([0.5, 0.55, 0.6, 0.65, 0.7]\) and \([0.4, 0.45, 0.5, 0.55, 0.6]\), respectively. The \( \varepsilon^{(m)} \) varies in \([0.5, 0.6, 0.7, 0.8, 0.9]\). When each of these parameters are varying in their domains, other parameters are kept as \( \varepsilon_{\text{max}} = 0.6, \varepsilon_{\text{min}} = 0.5, \varepsilon^{(m)} = 0.7 \).

Because the “Nonlinear Dynamic System with Time-varying Characteristics” of the Example 1 in Section 4.3.2.2 has obvious concept drift based on the three-state-shift, the data generated from (B.5) and (B.6) in Section B.1 of Appendix B are applied to analyze the sensitivity of the predefined parameters. All 3000 data pairs are applied for evaluation.

It can be seen from Table 4.1 that \( \varepsilon^{(m)} \) is a little bit more sensitive than \( \varepsilon_{\text{max}} \) and \( \varepsilon_{\text{min}} \). When \( \varepsilon^{(m)} < 0.7 \), the fuzzy system uses four rules to track the data stream and get worse accuracy, and when \( \varepsilon^{(m)} > 0.7 \), the fuzzy system applies more than eight rules to track the data stream and get worse accuracy than \( \varepsilon^{(m)} = 0.7 \). The underfitting and overfitting phenomenon is a possible reason to cause this problem. In addition, \( \varepsilon_{\text{min}} \) and \( \varepsilon_{\text{max}} \) suffer the similar problems.

Therefore, in all the numerical examples (see Section 4.3.2.2 - 4.3.2.4), \( \varepsilon_{\text{max}} = 0.6, \varepsilon_{\text{min}} = 0.5, \varepsilon^{(m)} = 0.7 \) are kept the same. Based on the existing research, the forgetting factor is set as \( \lambda = 0.9 \). Besides, the parameters in the VFF-WRLS are set as \( \beta_1 = 8, \beta_2 = 0.9, M_0 = 10^4, \) and \( M = 10^3 \) based on [203].

4.3.2.2 Example 1: Nonlinear Dynamic System with Time-varying Characteristics (term 2 in Section B.1, Appendix B)

This is an online learning example with all the data generated by the model (B.5) in Section B.1 of Appendix B are used for evaluation. The experiment is setup following the instruction in term 2 in the bullet of Section B.1, Appendix B. Numerical results are shown in Table 4.2. Fig. 4.2 displays the results from \( t = 900 \) to \( t = 2100 \), which contains the time of the two drifts at \( t = 1000 \) and \( t = 1500 \). Fig. 4.2 indicates that the SEFS can adapt to the changes of the state successfully with high accuracy. The running
time for the SEFS is 0.416035s.

TABLE 4.2: Example 1: Nonlinear dynamic plant identification with time varying characteristic.

<table>
<thead>
<tr>
<th>Method</th>
<th>No. of rules (AVG.)</th>
<th>No. of parameters (AVG.)</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSETSK [69]</td>
<td>11</td>
<td>77</td>
<td>0.0661</td>
</tr>
<tr>
<td>DENFIS [26]</td>
<td>15</td>
<td>105</td>
<td>0.1749</td>
</tr>
<tr>
<td>cTS [27]</td>
<td>11</td>
<td>77</td>
<td>0.0682</td>
</tr>
<tr>
<td>SEFS</td>
<td>6(3.5610)</td>
<td>63(24.9270)</td>
<td>0.0428</td>
</tr>
</tbody>
</table>

Fig. 4.2: Online identification results in example 1. ($t \in [900,2100]$)

Theoretically, all the fuzzy rules updated and learned by SEFS can be listed between time $t = 1$ and $t = 3000$. The number of rules of SEFS listed in Table 4.2 is the maximum number of fuzzy rules used by SEFS during the life-long online learning and predicting procedure of SEFS. There are 3 final rules learned by SEFS at time $t = 3000$. They are demonstrated as follows, in order to show the interpretability of SEFS:

$R_1$: If $x_1$ is $\Gamma_{1,1}$ and $x_2$ is $\Gamma_{1,2}$, then $y_1 = -0.0177 - 0.0007x_1 + 0.0899x_2$; (4.31)

$R_2$: If $x_1$ is $\Gamma_{1,1}$ and $x_2$ is $\Gamma_{1,2}$, then $y_2 = 0.2368 + 0.1355x_1 - 0.0873x_2$; (4.32)

$R_3$: If $x_1$ is $\Gamma_{1,1}$ and $x_2$ is $\Gamma_{1,2}$, then $y_3 = 0.8405 + 0.9055x_1 + 1.1099x_2$. (4.33)

The centres and radii of rules $R_1$, $R_2$, and $R_3$ are:

$c_1 = [0.4140, 1.0989]$ and $\sigma_1 = [0.6536, 0.6421]$; (4.34)
\[ c_2 = [-0.0891, 0.0885] \text{ and } \sigma_2 = [0.4663, 0.5108]; \quad (4.35) \]
\[ c_3 = [-0.7031, -0.8238] \text{ and } \sigma_3 = [0.4134, 0.5979]. \quad (4.36) \]

### 4.3.2.3 Example 2: Mackey-Glass Chaotic Time Series (term 5 in Section B.1, Appendix B)

This experiment is follow from the steps presented in term 5 in Section B.1, Appendix B. Numerical results are displayed in Table 4.3. The maximum number of fuzzy rules learned and used by SEFS is 4. However, at time \( t = 3500 \) the final number of fuzzy rules is 2. These two rules can be presented as:

\[ R_1: \text{If } \mathbf{x}_1 \text{ is } \Gamma_{1,1}, \text{ and } \mathbf{x}_2 \text{ is } \Gamma_{1,2}, \text{ and } \mathbf{x}_3 \text{ is } \Gamma_{1,3}, \text{ and } \mathbf{x}_4 \text{ is } \Gamma_{1,4}, \]
\[ \text{Then } y_1 = -0.2484 + 0.2540x_1 + 0.9071x_2 + 0.0321x_3 + 0.0825x_4; \quad (4.37) \]
\[ R_2: \text{If } \mathbf{x}_1 \text{ is } \Gamma_{2,1}, \text{ and } \mathbf{x}_2 \text{ is } \Gamma_{2,2}, \text{ and } \mathbf{x}_3 \text{ is } \Gamma_{2,3}, \text{ and } \mathbf{x}_4 \text{ is } \Gamma_{2,4}, \]
\[ \text{Then } y_2 = -0.2486 + 0.2526x_1 + 0.9066x_2 + 0.0328x_3 + 0.0811x_4. \quad (4.38) \]

The centres and radii of these above rules are:

\[ c_1 = [0.9313, 0.9310, 0.9312, 0.9322], \sigma_1 = [0.3328, 0.3330, 0.3330, 0.3327]; \quad (4.39) \]
\[ c_2 = [0.6017, 0.4682, 0.7878, 1.0072], \quad \sigma_2 = [0.5039, 0.5039, 0.5039, 0.5039]. \quad (4.40) \]

| TABLE 4.3: Example 2: Mackey-Glass Chaotic time series (long term prediction). |
|-------------------------------|----------------|----------------|-------|
| DENFIS [26]                  | 58             | 886            | 0.278 |
| eTS+ [29]                    | 10             | 130            | 0.392 |
| Simple eTS+ [37]             | 15             | 150            | 0.375 |
| GENEFIS(C) [56]              | 19             | 475            | 0.280 |
| GENEFIS(B) [56]              | 9              | 225            | 0.339 |
| PANFIS [55]                  | 19             | 475            | 0.301 |
| eT2RFNN [89]                 | 3              | 108            | 0.32  |
| GSETSK [69]                  | 19             | 247            | 0.347 |
| SPLAFIS [96]                 | 30             | 390            | 0.279 |
| DeTS [35]                    | 3              | 39             | 0.440 |
| GEFNS [81]                   | 5              | 65             | 0.2635|
| SEFS                         | 4(1.3043)      | 52(16.9559)    | 0.1287|
In Fig. 4.3, the online prediction errors are shown. It can be seen that most of the prediction errors are varying between -0.05 and 0.05. It takes 0.351014s for the SEFS to compute the results.

Observe from Table 4.3 that the numbers of fuzzy rules used by evolving type-2 recurrent fuzzy neural network (eT2RFNN) and dynamically evolving Takagi-Sugeno (DeTS) are slightly smaller than that used by SEFS. However, considering the complexity of the system structure and the accuracy achieved, SEFS still has its advantages. The specific analysis is shown as follows: i) generalized Gaussian membership function is implemented in eT2RFNN, which makes eT2RFNN has a more complicated structure with more additional parameters get involved to measure the relationship between the data. In this aspect, SEFS enables more accurate predictions while using a much more simple system; and ii) although DeTS has a slightly more simple structure than SEFS, the proposed algorithm improves the prediction accuracy significantly from 0.440 to 0.1287. Although both DeTS and SEFS apply a small amount of fuzzy rules and simple structures, this significant improvement in the accuracy is worth with this slight increase of the complexity.

4.3.2.4 Example 3: S&P 500 Daily Closing Price (term 7 in Section B.1, Appendix B)

The dataset and the experiment are the same as what presented in term 7 in Section B.1, Appendix B. The results are listed in Table 4.4. Fig. 4.4 and Fig. 4.5 demonstrate
that SEFS predicts the time series preciously. Furthermore, the maximum absolute error is 0.1030, and most of the prediction errors are located between -0.05 and 0.05, and the computing time of SEFS is 2.346683s.

TABLE 4.4: Example 3: Online prediction of S&P 500 daily closing price.

<table>
<thead>
<tr>
<th>Rule</th>
<th>No. of parameters (AVG.)</th>
<th>NDEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTS [27]</td>
<td>14</td>
<td>75</td>
</tr>
<tr>
<td>SimpLeTS [31]</td>
<td>7</td>
<td>39</td>
</tr>
<tr>
<td>PANFIS [55]</td>
<td>4</td>
<td>144</td>
</tr>
<tr>
<td>GENEFIS [56]</td>
<td>2</td>
<td>72</td>
</tr>
<tr>
<td>eT2RFNN [89]</td>
<td>2</td>
<td>110</td>
</tr>
<tr>
<td>SEFS</td>
<td>2(1.2835)</td>
<td>32(20.5360)</td>
</tr>
</tbody>
</table>

Fig. 4.4: Online prediction results for example 3.

There are maximum 2 fuzzy rules get involved in the whole online learning procedure of SEFS shown in Table 4.4. The number and parameters of fuzzy rules are changing over time, for this reason, different fuzzy rules can be learned at different time point. For example, at the last step, the final rule number is 1, and this rule can be represented as (4.41).

\[ R_1 : \text{If } x_1 \text{ is } \Gamma_{1,1}, \text{ and } x_2 \text{ is } \Gamma_{1,2}, \text{ and } x_3 \text{ is } \Gamma_{1,3}, \text{ and } x_4 \text{ is } \Gamma_{1,4}, \text{ and } x_5 \text{ is } \Gamma_{1,5}, \text{ Then } y_1 = 0.0000 + 0.0272x_1 + 0.0276x_2 - 0.0379x_3 - 0.0099x_4 + 0.9921x_5, \] (4.41)

in which the centre and radius are: [0.2265, 0.2265, 0.2265, 0.2265, 0.2265] and [0.4496, 0.4496, 0.4496, 0.4496, 0.4496], respectively. Whereas there are 2 fuzzy rules learned
and used at $t = 5$. These rules can be presented as:

$$R_1 : \text{If } x_1 \text{ is } \Gamma_{1,1}, \text{ and } x_2 \text{ is } \Gamma_{1,2}, \text{ and } x_3 \text{ is } \Gamma_{1,3}, \text{ and } x_4 \text{ is } \Gamma_{1,4}, \text{ and } x_5 \text{ is } \Gamma_{1,5}, \text{ Then }$$
$$y_1 = 10^{-3} \times (0.1490 - 0.0779x_1 - 0.0483x_2 - 0.0646x_3 + 0.0133x_4 - 0.0072x_5);$$

(4.42)

$$R_2 : \text{If } x_1 \text{ is } \Gamma_{2,1}, \text{ and } x_2 \text{ is } \Gamma_{2,2}, \text{ and } x_3 \text{ is } \Gamma_{2,3}, \text{ and } x_4 \text{ is } \Gamma_{2,4}, \text{ and } x_5 \text{ is } \Gamma_{2,5}, \text{ Then }$$
$$y_2 = 10^{-4} \times (0.0651 - 0.7051x_1 - 0.3922x_2 - 0.6465x_3 + 0.1667x_4 - 0.2874x_5).$$

(4.43)

The centres and radii of the above two rules are:

$$c_1 = 10^{-3}[0.0990, 0.1679, 0.2174, 0.2389, 0.2626] \text{ and }$$
$$\sigma_1 = 10^{-3}[0.1299, 0.1128, 0.1147, 0.1106, 0.1087];$$

(4.44)

$$c_2 = 10^{-3}[0.2067, 0.2712, 0.2389, 0.2777, 0.0646] \text{ and }$$
$$\sigma_2 = 10^{-3}[0.2139, 0.2139, 0.2139, 0.2139, 0.2139].$$

(4.45)

### 4.4 A Self-Learning Approach for Dynamic Rule Generation & Reduction Thresholds

This section will present EFS-SLAT. The main content of this section is based on our published approach in [207]. The research work presented in this section is the improved
version of SEFS which has been presented in Section 4.3. Besides, most EFSs use only inputs to determine the structure evolving, while the outputs are always neglected. This paper deals with these problems from proposing an EFS-SLAT. EFS-SLAT has a self-learning strategy for the thresholds. It enables to automatically adjust the thresholds and dynamically control the structure and parameters evolving speed of an EFS, relying on the risk of underfitting and overfitting reflected by the online training errors.

The rest of this section is arranged as follows: Section 4.4.1 gives an overview about how the EFS-SLAT works. Section 4.4.2 presents the learning approaches of the EFS-SLAT from three blocks — rule generation and antecedent parameters updating, rule reduction and consequent parameters learning. Numerical examples used to evaluate EFS-SLAT are given in Section 4.4.3.

4.4.1 Our Approach

EFS-SLAT will control the system structure evolving, including both rule generation and reduction, relying on the risks of overfitting and underfitting demonstrated by the training errors. The reason is that the overtrained model usually suffers the high risk of tremendous generalization problems, while the poorly trained model is usually insufficient to perform well on both the training and testing dataset. To be more specific, the EFS-SLAT adjusts the speed of evolving based on the relationship between overfitting, underfitting, training error and testing error presented by Fig. 4.6 (see also \cite{208}). In Fig. 4.6, the underfitting corresponds to high bias and low variance, while the overfitting corresponds to the opposite side; the “best” is known as the “bias-variance tradeoff”, which is the intersection of the bias and variance. A similar approach utilizing the Fig. 4.6 to tune the model structure can be found from \cite{209}. Compared to the existing approaches, EFS-SLAT makes achievement from the following aspects:

- The dynamic thresholds, which control rule generation and reduction speed, are defined to evolve as functions of the cumulative absolute training error with a gradual forgetting. Rule generation is accelerated/decelerated based on the risk of underfitting/overfitting reflected by the big/tiny training error. Rule simplification speed is tuned to be fast or slow with the same trend of the rule generation speed, in order to balance the possible side-effects (overfitting or underfitting) caused by the too-fast or too-slow speed of rule generation.
CHAPTER 4. EVOLVING FUZZY SYSTEMS: SELF-LEARNING APPROACHES FOR DYNAMIC THRESHOLDS

145

Fig. 4.6: Underfitting vs overfitting.

- To make sure that the EFSs are evolved/updated in order to meet the requirement that the updated system should give a better track of the latest information, EFS-SLAT decides to execute the structure change or existing rule updating by voting the evolving approaches which can achieve better training error for the latest training data. This approach enables the information contained in the output to be utilized in order to guide the system towards evolving in the proper direction.

4.4.2 Learning Approaches of EFS-SLAT

Fuzzy system identification approaches of EFS-SLAT are specifically explained in this section. Based on the dynamic and time-varying property of EFS-SLAT, the fuzzy system learnt by EFS-SLAT has its structure (rule numbers) and all the threshold parameters changing over time. This phenomenon naturally leads to the problem of how to identify or update the rule base and the parameters. Depended on the discussion in section 4.4.1, the five learning modules, rule generation, antecedent parameters updating, rule merging, rule pruning and consequent parameters updating, are demonstrated in Sections 4.4.2.1 - 4.4.2.3.

4.4.2.1 Rule Generation and Antecedent Parameters Updating

Rule generation depends on the novelty content of the new sample, which means that new rule should be generated when the old rule base is insufficient to cover the information of the new data. On the contrary side, the old rule base should be updated using the new sample, when new data are located within the range of the existing clusters.
Assume the new input-output pair to be \((x_{t+1}, y_{t+1})\). The firing strength can indicate the membership degree of a data point covered by a certain cluster, according to which the firing strength is used as the major metric to measure the novelty content with regard to the new input. Besides, to avoid unnecessary rules being generated when the \((x_{t+1}, y_{t+1})\) is of very low firing strength, \(|e_i^{old}|\) and \(|e_i^a|\) are compared. Because EFS-SLAT is a one-pass approach which was established based on the assumption of no historical data being remembered, so only one single pair \((x_{t+1}, y_{t+1})\) is used to compute these two factors. The \(|e_i^{old}| = |y_{t+1} - F^{old}(x_{t+1})|\) is the online training error computed by the old system with no new rules been generated; while \(|e_i^a| = |y_{t+1} - F^{new}(x_{t+1})|\) is calculated using the new system that has been generated a new rule. As \(|e_i^{old}| \geq |e_i^a|\) indicates the existing system can depict the new data pair well, so new rules should not be generated to keep the system in a low complexity while minimizing the training error. To summarize, the final decision of whether a new rule should be generated is determined from two aspects: (1) The new sample has very low firing strength; and (2) The training error can be substantially declined without unnecessary increasing of the complexity. The exact method is shown in Criterion 4.1.

**Criterion 4.1. Rule generation method:** If \(\forall i = 1, 2, \ldots, K\) such that \(\theta_i(x_{t+1}) < e_i^{(a)}\) and \(|e_i^{old}| \geq |e_i^a|\) holds, then generate a new fuzzy rule \(R_{K+1} = x_{t+1}\), cluster radius \(\sigma_{K+1} = \|x_{t+1} - c_i^*\|/\sqrt{-2\log(\sup_{e^{(a)}_i})}\), consequent parameters \(\psi_{K+1} = \psi_i^*\), where \(i^* = \arg\min_{i=1,2,\ldots,K} \|x_{t+1} - c_i^*\|^2\).

Note that, even if a new rule were to be generated due to false alarms (noise or outliers), the generated cluster will be kept as a tiny size and will be finally removed by the rule pruning approach. On the opposite side of rule generation, assume there exists a rule \(r_i\) such that the firing strength regarding to \(x_{t+1}\) is large. In this case, \(x_{t+1}\) should be regarded as a data sample in the corresponding cluster of rule \(r_i\), hence, the antecedent parameters of \(r_i\) should be adjusted based on the Criterion 4.2.

**Criterion 4.2. Rule updating method:** If \(\exists i^* \in \{1, 2, \ldots, K\}\) such that \(i^* = \arg\max_{i=1,\ldots,K} \theta_i(x_{t+1})\) and \(\theta_i(x_{t+1}) \geq e_i^{(a)}\), then adjust the centre and the radius of rule \(r_{i^*}\) by (4.46) and (4.47):

\[
c_i^{new} = c_i^{old} + \frac{1}{N_i^{old} + 1}(x_{t+1} - c_i^{old}), \quad (4.46)
\]

\[
(\sigma_{i,j}^{new})^2 = (\sigma_{i,j}^{old})^2 + \frac{(x_{t+1} - c_i^{new})^2 - (\sigma_{i,j}^{old})^2}{N_i^{old} + 1} + \frac{N_i^{old} (c_{i,j}^{new} - c_{i,j}^{old})^2}{N_i^{old} + 1}. \quad (4.47)
\]
CHAPTER 4. EVOLVING FUZZY SYSTEMS: SELF-LEARNING APPROACHES FOR DYNAMIC THRESHOLDS

Similar method can also be found in [26, 28, 31, 37, 65, 188, 199]. The number of samples within cluster formed by \( r_i \) should be updated by \( N_{\text{new}}^{t} = N_{\text{old}}^{t} + 1 \).

Both criterion 4.1 and criterion 4.2 are controlled by the threshold \( \varepsilon_{t+1}^{(a)} \). Despite of setting \( \varepsilon_{t+1}^{(a)} \) as a fixed predefined parameter, EFS-SLAT learns and updates \( \varepsilon_{t+1}^{(a)} \) dynamically based on the cumulative online training absolute error \( A(t + 1) = \sum_{k=1}^{t+1} \lambda^{t+1-k} e_k \) \( (e_k = |y_k - \hat{y}_{\text{train}}^k|, \hat{y}_{\text{train}}^k = F_{\text{updated}}(x_k), F_{\text{updated}} \) is the updated system obtained when all the evolving modules are processed using \( (x_k, y_k) \), \( \lambda \in [0, 1] \) is the forgetting factor). The adjusting mechanism of \( \varepsilon_{t}^{(a)} \) is:

- Tune up \( \varepsilon_{t}^{(a)} \) when \( A(t) \) grows. Large cumulative online training error \( A(t) \) indicates the model is underfitting, hence an insufficient number of fuzzy rules are used. New rules should be generated quickly to catch up with the rapid changes of the data stream, and to help to improve the performance. This task can be achieved by increasing \( \varepsilon_{t}^{(a)} \).

- The \( \varepsilon_{t}^{(a)} \) should be set small, when \( A(t) \) has a small value. Although \( A(t) \) can not exactly reflect whether the overfitting really happens, it can act as an indicator to present the risk for overfitting (i.e. the smaller the \( A(t) \), the higher risk for the overfitting). Therefore, it is appropriate to slow down the speed for rule generation when \( A(t) \) declines.

Therefore, the \( \varepsilon_{t}^{(a)} \) is designed to be the monotonically increasing function of \( A(t) \) in order to achieve the above mechanism. The logistic function \( g(x) = 1/(1 + \exp\{-x\}) \), which is a frequently used monotonically increasing function for clamping the data into a certain range in machine learning, is used to construct the threshold variation function. Considering the k-sigma rule for the multivariate Gaussian distribution, and the 1-sigma which forms the core of a Gaussian fuzzy set with its explanatory term, the upper bound of \( \varepsilon_{t}^{(a)} \) is set to be \( \exp\{-0.5\} \). Besides, the natural lower bound is 0, which means no rules need to be generated. To summarize, \( \varepsilon_{n}^{(a)} \in [0, \exp\{-0.5\}] \) is calculated through:

\[
\varepsilon_{t}^{(a)} = 2e^{-0.5\{g(-\varepsilon_t) - \frac{1}{2}\}} = e^{-0.5\left\{\frac{2}{1 + \exp\{-A(t)\}} - 1\right\}}, \quad (4.48)
\]

where \( A(t) \) can be computed recursively by \( A(t) = \lambda A(t-1) + e_t \), and \( A(t+1) = \lambda A(t) + e_{t+1} \). As a result, this recursive formula can be used to update the \( \varepsilon_{t}^{(a)} \) by
(4.49):
\[
\varepsilon_{t+1}^{(a)} = e^{-0.5 \left\{ \frac{2}{1 + \exp\{-\lambda A(t) - \varepsilon_{t+1}\}} - 1 \right\}}. \tag{4.49}
\]

It can be observed that big \( e_i \) can make \( \varepsilon_{t}^{(a)} \) very close to its upper bound. Besides, the lower bound 0 would be touched at the beginning of learning; or it would also be reached when the data have a very simple behaviour. The \( A(t) \) will be reset as 0, once there is a new rule generated. This allows the rule generation speed to vary based on the different training phases. To ensure \( \varepsilon_{t}^{(a)} \) varying in a reasonable degree within \([0, \exp\{-0.5\}]\), the centralization of the dataset, which have big values e.g. the value of each data point is larger than 10, 100 or 1000, is recommended.

### 4.4.2.2 Rule Reduction

With the arrival of increasing number of data samples, sufficient knowledge of the data stream would be gained by the EFS-SLAT. Hence, the whole picture of the data behaviour would be reflected gradually. Under the background of online rule generation and adjusting, there should be two common phenomena: (1) Some of the constructed fuzzy rules as well as the clusters would become similar; (2) There also exist some fuzzy rules which grow to be redundant rules with tiny clusters, because they have seldom been activated since they were constructed. Both of the above cases may lead to overfitting, and rules with similar antecedent parts would cause rule conflict. To cope with these problems, a rule merging (Criterion 4.4) and a rule pruning approach (Criterion 4.5) are used.

#### 4.4.2.2.1 Merge Similar Rules

The similarity between two fuzzy rules \( S(\cdot, \cdot) \) is calculated by the \( L^2 \) distance between the firing strengths directly, rather than by heuristic approaches. The same as [199], the formula to compute the similarity between two rules \( R_{l_1} \) and \( R_{l_2} \) is presented in (4.50):
\[
S(R_{l_1}, R_{l_2}) = \frac{1}{1 + \| \gamma_{l_1}(x) - \gamma_{l_2}(x) \|_{L^2}}, \tag{4.50}
\]
which is the same as (4.10), and the \( \| \gamma_{l_1}(x) - \gamma_{l_2}(x) \|_{L^2} \) can be computed by (4.19).

However, simply merging rules with small similarity \( S(\cdot, \cdot) \) in formula (4.50) may
CHAPTER 4. EVOLVING FUZZY SYSTEMS: SELF-LEARNING APPROACHES
FOR DYNAMIC THRESHOLDS

cause inappropriate merging. This is because, when two fuzzy rules have very small radii and their centres are actually not “close” to each other, then the similarity between these two rules would also be of a small value. In order to avoid this, only those “touched” rules are considered to be merged. It is well known that if \( X \sim N(c, \Sigma)^8 \), then the \( k \)-sigma rule can be used through computing the Mahalanobis distance \( D(x, c) = \sqrt{(x - c)^T \Sigma^{-1} (x - c)} \). For the conventional T-S systems, the Mahalanobis distance would reduce to the standardized Euclidean distance. \( D(x, c) \leq 3 \) can make sure that 99.7% of the data in the cluster are inside the enclosed ellipsoid.

Based on the above theory, whether two rules are touched is judged by testing whether one of these two clusters can have its centre located within the ellipsoid, which is \( \sqrt{(X - c)^T \Sigma^{-1} (X - c)} \leq 3 \), of another cluster. Therefore, whether rule \( R_i^\tau \) and \( R_j^\tau \) are “touched” is judged by Critterion 4.3.

**Criterion 4.3.** Overlapping rules: If rule \( R_i^\tau \) and \( R_j^\tau \) satisfy that \( \theta_i^\tau (c_j^\tau) \geq \exp(-9/2) \) or \( \theta_j^\tau (c_i^\tau) \geq \exp(-9/2) \), then \( R_i^\tau \) and \( R_j^\tau \) are regarded to be the overlapping rules.

Following on formula (4.50) and Criterion 4.3, similar fuzzy rules are merged according to the strategy shown in Criterion 4.4.

**Criterion 4.4. Rule merging method:** If rule \( R_i^\tau \) is updated by Criterion 4.2, and \( \exists i'' \in \{1, 2, \ldots, K\} \), \( i'' \neq i' \), such that \( i'' = \arg \max_{i'' \neq i', \rho = 1, \ldots, K} \{S(R_i^\tau, R_i^\rho)\} \), \( S(R_i^\tau, R_i^\rho) > \epsilon_i^{(m)} \), \( R_i^\tau \) and \( R_i^\rho \) satisfy Criterion 4.3, and \( |e_i^{\tau \rho} - \epsilon_i^{(m)}| \) hold, then merge rule \( R_i^\tau \) with \( R_i^\rho \). \( S(R_i^\tau, R_i^\rho) \) is the similarity value between \( R_i^\tau \) and \( R_i^\rho \), and \( S(\cdot, \cdot) \in [0, 1] \).

In Criterion 4.4, \( |e_{t+1}^{\tau \rho} - F^{\tau \rho}(x_{t+1})| \) is the absolute training error computed by the system before merging, and \( |e_{t+1}^{m} - F^{new}(x_{t+1})| \) is computed by the new system when \( R_i^\tau \) and \( R_i^\rho \) are merged. Based on Criterion 4.4, assume the merged rule is \( R_i^\tau \). The centre and radius of \( R_i^\tau \) are \( c_i^\tau \) (4.51) and \( \sigma_i^\tau \) (4.52):

\[
c_i^\tau = \frac{N_i^\tau c_i^\tau + N_i^\rho c_i^\rho}{N_i^\tau + N_i^\rho},
\]

\[
(\sigma_i^\tau)^2 = \frac{1}{N_i^\tau + N_i^\rho} \{N_i^\tau (\sigma_{i,j})^2 + N_i^\rho (\sigma_{i,j})^2 + N_i^\tau (c_{i,j} - c_{\tau,j})^2 + N_i^\rho (c_{i,j} - c_{\tau,j})^2\},
\]

---

\(^8\) \( N(c, \Sigma) \) is the \( m \)-dimensional multivariate Gaussian distribution with mean \( c \) and covariance matrix \( \Sigma \).
similar method of which can also be found from [39, 199]. Consequent parameters are
\[ \psi_i' = \frac{N_i \psi_i + N_i' \psi_i'}{N_i + N_i'}, \]
the number of samples in the merged cluster is \( N_t = N_i + N_i'. \) The updated system after merging \( R_i \) and \( R_i' \) has both of these two rules deleted, but uses one rule \( R_i'' \) to replace them. Remark that after merging the rule numbers is updated by \( K - 1. \) In a similar way to generating new fuzzy rules, Criterion 4.4 demonstrates that rule merging is also controlled by a time varying threshold parameter \( \epsilon_t^{(m)} \). EFS-SLAT learns \( \epsilon_t^{(m)} \) based on the following theory:

- Large cumulative online training absolute error \( A(t) \) demonstrates the system is suffering underfitting, under which circumstance, \( \epsilon_t^{(a)} \) becomes large in order to allow the structure of the system to change rapidly to capture the dynamics of the data. A multiplicity of new rules are likely to generate mini-clusters which may lead to over-fitting. To shrink the rule base and prevent overfitting, \( \epsilon_t^{(m)} \) should be tuned down to allow more fuzzy rules to be merged.

- Tiny cumulative online training absolute error \( A(t) \), which is accompanied with a slow rule generation procedure, shows the system can perfectly reflect the behaviour of the data. The slow rule generation speed, which is controlled by a small \( \epsilon_t^{(a)} \), needs to be balanced by a slow rule merging speed with large \( \epsilon_t^{(m)} \) in case of under-fitting, due to which merging rules quickly would lead to some important historical information being discarded.

Therefore, \( \epsilon_t^{(m)} \) and \( \epsilon_t^{(a)} \) have a close relationship with each other. Furthermore, \( \epsilon_t^{(m)} \) is constructed to balance the possible side-effects caused by \( \epsilon_t^{(a)} \). As the threshold \( \epsilon_t^{(m)} \) for the geometric similarity should be varying within [0, 1], and 0.5 is the mean of the lower and upper bound of \( \epsilon_t^{(m)} \), so \( \epsilon_t^{(m)} \) is built as a function with range (0.5, 1]. Similar to the \( \epsilon_t^{(a)} \), the logistic function \( g(A(t)) \) is also used to formulate \( \epsilon_t^{(m)} \). The exact formula is shown in (4.53):

\[
\epsilon_t^{(m)} = 1 - \left\{ g(A(t)) - \frac{1}{2} \right\} = \frac{3}{2} - \frac{1}{1 + \exp\{-A(t)\}}. \tag{4.53}
\]

In addition, applying the recursive formula of \( A(t) \), the \( \epsilon_t^{(m)} \) can be computed by:

\[
\epsilon_t^{(m)} = \frac{3}{2} - \frac{1}{1 + \exp\{-\lambda A(t) - \epsilon_{t+1} \}}. \tag{4.54}
\]
4.4.2.2 Prune Redundant Rules

Due to the online property of EFS-SLAT, no historical data is known before the algorithm begins to learn. It is a fact that there would be some fuzzy rules that have seldom/never been activated since they were built, which makes these rules redundant and furthermore has a negative influence on the prediction accuracy. "Utility" is the metric, which measures whether the rules are useful through considering the cumulative firing strength and the "age". This "utility" metric is used in this section to pick those rules which need to be removed. EFS-SLAT does not throw out these redundant rules directly. Alternatively, it removes these rules by merging them with their nearest clusters. The reason behind this is that these small clusters are also learnt from the data, and it would have a negative influence on the learning accuracy when these clusters to be discarded directly. By adopting the merging operation, on the one hand, the performance and the historical information would be effectively reserved; On the other hand, the rule number would be shrunk to relieve the computation burden. To be more specific, EFS-SLAT applies the following rule pruning criterion, Criterion 4.5.

Criterion 4.5. Rule pruning method: Assume rule $R_i$ with $i^p = \arg\min_{i=1,\ldots,K} U_i$ is the fuzzy rule which has the smallest "utility". The "utility" of $R_i$ can be computed by (4.55):

$$U_i = \frac{\sum_{k=t_i}^{t+1} \theta_i(x_k)}{(t + 1 - t_i)}, \quad (4.55)$$

where $k = t_i, \ldots, t+1$ is the time stamp, $t_i$ is the time stamp when the rule $R_i$ is generated, and the $\sum_{k=t_i}^{t+1} \theta_i(x_k)$ is the cumulative firing strength. If the "utility" $U_{i^p}$ is smaller than $\varepsilon_i^{(a)}$ in the Criterion 4.1, then merge it with rule $R_{i^p}$, where $i^p = \arg\min_{i\neq i^p, i=1,\ldots,K} \|c_i - c_{i^p}\|$, according to Criterion 4.4.

4.4.2.3 Learning Consequent Parameters

Consequent parameters are learnt by WRLS. The objective functions are the local error functions (4.56),

$$E_i = \sum_{k=1}^{t+1} \theta_i(y_k - \hat{y}_k^{\text{train}})^2. \quad (4.56)$$
These error functions (4.56) are optimized by the WRLS in (4.57) and (4.58).

\[ \psi_i(t+1) = \psi_i(t) + \frac{\theta_i(x_{t+1}) P_i(t) x e_{t+1} (y_{t+1} - y_{\text{train}})}{1 + \theta_i(x_{t+1}) x e_{t+1}^T P_i(t) x e_{t+1}}, \tag{4.57} \]

\[ P_i(t+1) = P_i(t) - \frac{\theta_i(x_{t+1}) P_i(t) x e_{t+1} x e_{t+1}^T P_i(t)}{1 + \theta_i(x_{t+1}) x e_{t+1}^T P_i(t) x e_{t+1}}, \tag{4.58} \]

where \( x e_{t+1} = (1, x_{t+1}) \), \( i = 1, 2, \ldots, K \).

### 4.4.3 Numerical Examples

The evaluation of EFS-SLAT is carried out from the frequently used benchmark examples including S&P 500 closing price prediction, Mackey-Glass Chaotic time series prediction. Besides, EFS-SLAT is also tested on Delta Ailerons data from KEEL-dataset\(^9\), helicopter unmanned aerial vehicle streaming data [210], and MPG and Boston Housing data from UCI repository\(^10\). In all these examples, the forgetting factor \( \lambda \) is vary from [0.8, 0.85, 0.9, 0.95, 0.99]. The suggested range of the forgetting factor is [0.8, 1]. RMSE, NDEI and MAE (see (B.3) in the Section B.1, Appendix B) are applied to judge the accuracy.\(^11\) As a one-pass online approach, EFS-SLAT is evaluated by the typical online learning evaluation method, test-then-train (prequential) [211], on a single epoch. Each individual example is used to test the model before it is used for training. Prequential approach enables the algorithm to make maximum usage of the available data, and does not require any hold-out training set in essence [212]. It can be found from [211, 213] that the prequential error \( S \) is computed by the accumulated sum of the loss function:

\[ S = \sum_{t=1}^{N} L(y_{t+1}, \hat{y}_{t+1}), \tag{4.59} \]

where \( L(y_{t+1}, \hat{y}_{t+1}) \) is the loss function \( L(y_{t+1}, \hat{y}_{t+1}) = (y_{t+1} - \hat{y}_{t+1})^2 \). For all these examples, EFS-SLAT has been operated on an online manner. Not only the testing accuracy computed based on the testing set, but also the online testing RMSEs, NDEIs with regard to to all the data in each data set, as well as the prequential errors are presented.

---

\(^9\)https://sci2s.ugr.es/keel/datasets.php.
\(^10\)http://archive.ics.uci.edu/ml/.
\(^11\)Numerical results of EFS-SLAT are calculated in the environment of intel(R) core (TM) i7-4790 CPU with a 3.6 GHz processor and a 16.0 GB memory.
Additionally, the full trajectories of the online learning errors and thresholds $\varepsilon_t^{(a)}$ and $\varepsilon_t^{(m)}$ with regard to the best results (for Example 1–4) are presented in the Fig. 4.7 and Fig. 4.8, respectively.

![Fig. 4.7: Online testing errors.](image)

**4.4.3.1 Example 1: S&P 500 Closing Price Prediction (term 7 in Section B.1, Appendix B)**

The dataset and experiment setup used in this example can be found from the term 7 in Section B.1, Appendix B. The comparison results are shown in Table 4.5 which indicates that EFS-SLAT achieves precious prediction results. Moreover, the online testing RMSE is 0.0045, NDEI is 0.0157 for all the values of $\lambda$ except 0.8. When $\lambda = 0.8$, the NDEI equals to 0.0156. For $\lambda$ equals to [0.8, 0.85, 0.9, 0.95, 0.99], the prequential error $S$ is about [0.5972, 0.5995, 0.5994, 0.6035, 0.6003]. Although EFS-SLAT uses more rules than some of the other approaches, the EFS-SLAT can achieve more accurate results than all of the others, no matter which forgetting factor is used and without requirement for setting the thresholds. In this sense, EFS-SLAT is still a preferable approach, as the high accuracy is the priority goal in system learning and identification. Based on the prequential error, EFS-SLAT achieves the most accurate result when
...whereas the same testing results as $\lambda = [0.8, 0.85, 0.9, 0.95]$ can be obtained with a smaller number of rules when $\lambda = 0.99$. Additionally, observing from Fig. 4.7 and 4.8 it can be concluded that the thresholds have larger fluctuations when time series have more significant nonstationary phenomenon than the smooth areas.

In order to show the interpretability of EFS-SLAT, the case $\lambda = 0.99$ is taken as an example. There are 9 fuzzy rules learned and used at the last step of online learning, with maximum rule numbers learned by EFS-SLAT is 23 shown in Table 4.5. Due to the space limitation, the first 3 fuzzy rules out of 9 rules learned by EFS-SLAT when it stops are shown as follows:

\begin{align*}
R_1 & : \text{If } x_1 \text{ is } \Gamma_{1,1}, \text{ and } x_2 \text{ is } \Gamma_{1,2}, \text{ and } x_3 \text{ is } \Gamma_{1,3}, \text{ and } x_4 \text{ is } \Gamma_{1,4}, \text{ and } x_5 \text{ is } \Gamma_{1,5}, \text{ Then } \quad y_1 = 0.0000 - 0.0045x_1 - 0.0083x_2 + 0.0295x_3 - 0.1190x_4 + 1.1018x_5; (4.60) \\
R_2 & : \text{If } x_1 \text{ is } \Gamma_{2,1}, \text{ and } x_2 \text{ is } \Gamma_{2,2}, \text{ and } x_3 \text{ is } \Gamma_{2,3}, \text{ and } x_4 \text{ is } \Gamma_{2,4}, \text{ and } x_5 \text{ is } \Gamma_{2,5}, \text{ Then } \quad y_2 = 0.0000 + 0.0384x_1 - 0.0354x_2 - 0.0175x_3 - 0.0195x_4 + 1.0337x_5; (4.61) \\
R_3 & : \text{If } x_1 \text{ is } \Gamma_{3,1}, \text{ and } x_2 \text{ is } \Gamma_{3,2}, \text{ and } x_3 \text{ is } \Gamma_{3,3}, \text{ and } x_4 \text{ is } \Gamma_{3,4}, \text{ and } x_5 \text{ is } \Gamma_{3,5}, \text{ Then } \quad y_3 = -0.0000 + 0.0476x_1 - 0.0561x_2 + 0.0529x_3 + 0.0102x_4 + 0.9461x_5. (4.62)
\end{align*}
The centres and radii of rules $R_1$, $R_2$, and $R_3$ are:

$$c_1 = [0.0431, 0.0431, 0.0431, 0.0431, 0.0431] \text{ and }$$
$$\sigma_1 = [0.0218, 0.0218, 0.0218, 0.0218, 0.0218]; \quad (4.63)$$

$$c_2 = [0.2267, 0.2268, 0.2268, 0.2268, 0.2268] \text{ and }$$
$$\sigma_2 = [0.0494, 0.0494, 0.0494, 0.0494, 0.0494]; \quad (4.64)$$

$$c_3 = [0.4938, 0.4937, 0.4936, 0.4936, 0.4937] \text{ and }$$
$$\sigma_3 = [0.0768, 0.0765, 0.0763, 0.0761, 0.0761]. \quad (4.65)$$

**TABLE 4.5:** Example 1: S&P 500 daily closing prices.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Num. (AVG.)</th>
<th>NDEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENEFIS [56]</td>
<td>2</td>
<td>0.07</td>
</tr>
<tr>
<td>PANFIS [55]</td>
<td>4</td>
<td>0.09</td>
</tr>
<tr>
<td>eT2RFNN [65]</td>
<td>2</td>
<td>0.04</td>
</tr>
<tr>
<td>eTS [27]</td>
<td>14</td>
<td>0.04</td>
</tr>
<tr>
<td>SimpLeTS [95]</td>
<td>7</td>
<td>0.04</td>
</tr>
<tr>
<td>ANFIS [86]</td>
<td>32</td>
<td>0.02</td>
</tr>
<tr>
<td>LEOA [188]</td>
<td>52</td>
<td>0.1229</td>
</tr>
<tr>
<td>SEFS [199]</td>
<td>2(1.2835)</td>
<td>0.0182</td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.8$</strong></td>
<td>31(19.6604)</td>
<td>0.0156</td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.85$</strong></td>
<td>30(20.0015)</td>
<td>0.0156</td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.9$</strong></td>
<td>27(17.3987)</td>
<td>0.0156</td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.95$</strong></td>
<td>25(13.8840)</td>
<td>0.0156</td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.99$</strong></td>
<td>23(10.7570)</td>
<td><strong>0.0156</strong></td>
</tr>
</tbody>
</table>

### 4.4.3.2 Example 2: Mackey-Glass Chaotic Time Series Prediction (term 5 in Section B.1, Appendix B)

This experiment follows term 5 in Section B.1, Appendix B. Numerical results are displayed in Table 4.6. Table 4.6 illustrates that EFS-SLAT with $\lambda = 0.8$ achieves the best results, and the results obtained by other values of $\lambda$ are also comparable with the state-of-the-art approaches. The online learning results regarding all the data points are presented in Table 4.7. The best results are obtained by $\lambda = 0.8$ regarding to RMSE, NDEI and the prequential errors. Fig. 4.7 shows that the online testing accuracy for the whole data set varies between -0.1226 and 0.1141; Fig. 4.8 also presents the full
trajectories for $\varepsilon_t^{(a)}$ and $\varepsilon_t^{(m)}$. It can be observed that the largest value of $\varepsilon_t^{(a)}$ and the smallest value of $\varepsilon_t^{(m)}$ are 0.0873 and 0.9280, respectively.

**TABLE 4.6:** Example 2: Mackey-Glass Chaotic time series.

<table>
<thead>
<tr>
<th>Rule Num. (AVG.)</th>
<th>NDEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>DENFIS [26]</td>
<td>58</td>
</tr>
<tr>
<td>GENEFIS(C) [56]</td>
<td>19</td>
</tr>
<tr>
<td>PANFIS [55]</td>
<td>19</td>
</tr>
<tr>
<td>eT2RFNN [65]</td>
<td>3</td>
</tr>
<tr>
<td>GSETSK [69]</td>
<td>19</td>
</tr>
<tr>
<td>eTS [27]</td>
<td>9</td>
</tr>
<tr>
<td>SimpLeTS [95]</td>
<td>11</td>
</tr>
<tr>
<td>SAFIS [94]</td>
<td>6</td>
</tr>
<tr>
<td>SPLAFIS [96]</td>
<td>30</td>
</tr>
<tr>
<td>DeTS [35]</td>
<td>3</td>
</tr>
<tr>
<td>GEFNS [81]</td>
<td>5</td>
</tr>
<tr>
<td>eMG($\Sigma_{init} = 10^{-3}I_4, w = 20$) [54]</td>
<td>58</td>
</tr>
<tr>
<td>LEOA [188]</td>
<td>42</td>
</tr>
<tr>
<td>SEFS [199]</td>
<td>4(1.3043)</td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.8$</strong></td>
<td><strong>8(5.2529)</strong></td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.85$</strong></td>
<td>7(4.8231)</td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.9$</strong></td>
<td>6(3.8314)</td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.95$</strong></td>
<td>8(3.0637)</td>
</tr>
<tr>
<td><strong>EFS-SLAT $\lambda = 0.99$</strong></td>
<td>8(2.5749)</td>
</tr>
</tbody>
</table>

**TABLE 4.7:** Example 2: Online testing results for all the data.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>RMSE</th>
<th>NDEI</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td><strong>0.0314</strong></td>
<td><strong>0.1411</strong></td>
<td><strong>3.4440</strong></td>
</tr>
<tr>
<td>0.85</td>
<td>0.0319</td>
<td>0.1435</td>
<td>3.5612</td>
</tr>
<tr>
<td>0.9</td>
<td>0.0327</td>
<td>0.1470</td>
<td>3.7409</td>
</tr>
<tr>
<td>0.95</td>
<td>0.0350</td>
<td>0.1575</td>
<td>4.2929</td>
</tr>
<tr>
<td>0.99</td>
<td>0.0347</td>
<td>0.1561</td>
<td>4.2147</td>
</tr>
</tbody>
</table>

The final model ($t = 3500$) learned by EFS-SLAT ($\lambda = 0.8$) can be presented by the following 5 rules:

$R_1$: If $x_1$ is $\Gamma_{1,1}$, and $x_2$ is $\Gamma_{1,2}$, and $x_3$ is $\Gamma_{1,3}$, and $x_4$ is $\Gamma_{1,4}$, Then

$$y_1 = 0.2119 + 0.9887x_1 - 0.2894x_2 + 0.3600x_3 - 0.3123x_4; \quad (4.66)$$

$R_2$: If $x_1$ is $\Gamma_{2,1}$, and $x_2$ is $\Gamma_{2,2}$, and $x_3$ is $\Gamma_{2,3}$, and $x_4$ is $\Gamma_{2,4}$, Then

$$y_2 = 0.2011 + 1.2960x_1 - 0.5139x_2 + 0.2630x_3 - 0.2314x_4; \quad (4.67)$$

$R_3$: If $x_{k,1}$ is $\Gamma_{3,1}$, and $x_2$ is $\Gamma_{3,2}$, and $x_3$ is $\Gamma_{3,3}$, and $x_4$ is $\Gamma_{3,4}$, Then
\[ y_3 = 0.3665 + 0.8705x_{k,1} + 0.0137x_{k,2} - 0.2024x_3 - 0.0785x_4; \]  
\[ R_4 : \text{If } x_1 \text{ is } \Gamma_{4,1}, \text{ and } x_2 \text{ is } \Gamma_{4,2}, \text{ and } x_3 \text{ is } \Gamma_{4,3}, \text{ and } x_4 \text{ is } \Gamma_{4,4}, \text{ Then} \]
\[ y_4 = 0.2644 + 0.8997x_1 - 0.1537x_2 + 0.2473x_3 - 0.2979x_4; \]  
\[ R_5 : \text{If } x_1 \text{ is } \Gamma_{5,1}, \text{ and } x_2 \text{ is } \Gamma_{5,2}, \text{ and } x_3 \text{ is } \Gamma_{5,3}, \text{ and } x_4 \text{ is } \Gamma_{5,4}, \text{ Then} \]
\[ y_5 = 0.3680 + 0.8697x_1 + 0.0108x_2 - 0.1938x_3 - 0.0854x_4. \]

The centres and radii of these rules are:

\[ c_1 = [0.9781, 1.0901, 1.1419, 1.1170], \sigma_1 = [0.1371, 0.1028, 0.0938, 0.1230]; \]  
\[ c_2 = [0.6776, 0.6920, 0.8263, 0.9908], \sigma_2 = [0.1328, 0.1478, 0.1875, 0.1293]; \]  
\[ c_3 = [1.0980, 0.8900, 0.6916, 0.6256], \sigma_3 = [0.1074, 0.1332, 0.1134, 0.1121]; \]  
\[ c_4 = [1.1909, 1.1655, 0.9944, 0.7562], \sigma_4 = [0.0735, 0.0940, 0.1179, 0.1109]; \]  
\[ c_5 = [0.7724, 0.5483, 0.5336, 0.8268], \sigma_5 = [0.1680, 0.1548, 0.1581, 0.1756]. \]

\[ \text{4.4.3.3 Example 3: Delta Ailerons Data (term 8 in Section B.1, Appendix B)} \]

Example based on the Delta Ailerons data follows from the steps in term 8 in Section B.1, Appendix B. Table 4.8 indicates that, no matter which forgetting factor is selected, the EFS-SLAT can give better results compared to other state-of-the-art approaches. Besides, \( \lambda = 0.9 \) gives the most accurate results observed from both Table 4.8 and Table 4.9. The trajectories of the errors, \( e^{(a)} \) and \( e^{(m)} \) are shown in Fig. 4.7 and Fig. 4.8, respectively.

<table>
<thead>
<tr>
<th>Rule num. (AVG.)</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAFIS [94]</td>
<td>14</td>
</tr>
<tr>
<td>eTS [27]</td>
<td>4</td>
</tr>
<tr>
<td>SimpLeTS [95]</td>
<td>4</td>
</tr>
<tr>
<td>GEFNS [81]</td>
<td>3</td>
</tr>
<tr>
<td>SEFS [199]</td>
<td>7(1.5112)</td>
</tr>
<tr>
<td>EFS-SLAT ( \lambda = 0.8 )</td>
<td>5(1.8174)</td>
</tr>
<tr>
<td>EFS-SLAT ( \lambda = 0.85 )</td>
<td>5(1.7312)</td>
</tr>
<tr>
<td>EFS-SLAT ( \lambda = 0.9 )</td>
<td>4(1.6287)</td>
</tr>
<tr>
<td>EFS-SLAT ( \lambda = 0.95 )</td>
<td>4(1.5879)</td>
</tr>
<tr>
<td>EFS-SLAT ( \lambda = 0.99 )</td>
<td>4(1.5932)</td>
</tr>
</tbody>
</table>
CHAPTER 4. EVOLVING FUZZY SYSTEMS: SELF-LEARNING APPROACHES FOR DYNAMIC THRESHOLDS

Despite of the results presented in Tables 4.8 and 4.9, the final model \( t = 7129 \) learned by EFS-SLAT (\( \lambda = 0.9 \)) can be presented by 1 fuzzy rule:

\[
R_1 : \text{If } x_1 \text{ is } \Gamma_{1,1}, \text{ and } x_2 \text{ is } \Gamma_{1,2}, \text{ and } x_3 \text{ is } \Gamma_{1,3}, \text{ and } x_4 \text{ is } \Gamma_{1,4}, \text{ and } x_5 \text{ is } \Gamma_{1,5}, \text{Then} \quad y_1 = 0.8956 - 0.2340x_1 - 0.0195x_2 + 0.0111x_3 - 0.1118x_4 - 0.3388x_5. \quad (4.76)
\]

The centre and radius of the above rule are:

\[
c_1 = [0.5524, 0.4714, 0.3565, 0.5214, 0.6391] \quad \text{and} \quad \sigma_1 = [0.1857, 0.1380, 0.1486, 0.1828, 0.1313]. \quad (4.77)
\]

4.4.3.4 Example 4: Helicopter Unmanned Aerial Vehicle Streaming Data (term 9 in Section B.1, Appendix B)

Using the dataset and experiment steps shown in the term 9 in Section B.1, Appendix B. Comparison results and the online learning accuracies are presented in Table 4.10 and Table 4.11, respectively. EFS-SLAT with \( \lambda = 0.95 \) achieves the best result.

### TABLE 4.10: Example 4: Helicopter unmanned aerial vehicle streaming data.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Rule Num. (AVG.)</th>
<th>RMSE</th>
<th>NDEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTS [27]</td>
<td>3</td>
<td>0.0535</td>
<td>0.8352</td>
</tr>
<tr>
<td>Simpl_eTS [95]</td>
<td>3</td>
<td>0.0534</td>
<td>0.8336</td>
</tr>
<tr>
<td>GENEFIS [56]</td>
<td>2</td>
<td>0.0355</td>
<td>0.5541</td>
</tr>
<tr>
<td>PANFIS [55]</td>
<td>9</td>
<td>0.0362</td>
<td>0.5652</td>
</tr>
<tr>
<td>Type-1 PALM(L) [210]</td>
<td>6</td>
<td>0.0363</td>
<td>0.5668</td>
</tr>
<tr>
<td>Type-1 PALM(G) [210]</td>
<td>11</td>
<td>0.0313</td>
<td>0.4886</td>
</tr>
<tr>
<td><strong>EFS-SLAT</strong> ( \lambda = 0.8 )</td>
<td>9(5.8460)</td>
<td>0.0311</td>
<td>0.4846</td>
</tr>
<tr>
<td><strong>EFS-SLAT</strong> ( \lambda = 0.85 )</td>
<td>8(5.4463)</td>
<td>0.0312</td>
<td>0.4862</td>
</tr>
<tr>
<td><strong>EFS-SLAT</strong> ( \lambda = 0.9 )</td>
<td>7(4.7357)</td>
<td>0.0309</td>
<td>0.4820</td>
</tr>
<tr>
<td><strong>EFS-SLAT</strong> ( \lambda = 0.95 )</td>
<td>11(4.2770)</td>
<td><strong>0.0305</strong></td>
<td><strong>0.4758</strong></td>
</tr>
<tr>
<td><strong>EFS-SLAT</strong> ( \lambda = 0.99 )</td>
<td>9(2.0945)</td>
<td>0.0316</td>
<td>0.4920</td>
</tr>
</tbody>
</table>
TABLE 4.11: Example 4: Online testing results for all the data.

<table>
<thead>
<tr>
<th>( \lambda )</th>
<th>0.8</th>
<th>0.85</th>
<th>0.9</th>
<th>0.95</th>
<th>0.99</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE</td>
<td>0.0252</td>
<td>0.0253</td>
<td>0.0252</td>
<td>0.0245</td>
<td>0.0247</td>
</tr>
<tr>
<td>NDEI</td>
<td>0.2030</td>
<td>0.2035</td>
<td>0.2024</td>
<td>0.1975</td>
<td>0.1991</td>
</tr>
<tr>
<td>S</td>
<td>3.8184</td>
<td>3.8367</td>
<td>3.7961</td>
<td>3.6138</td>
<td>3.6749</td>
</tr>
</tbody>
</table>

TABLE 4.12: Example 5: MPG and Boston Housing data.

<table>
<thead>
<tr>
<th></th>
<th>MPG</th>
<th>Boston Housing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE±STD</td>
<td>max</td>
</tr>
<tr>
<td>genfis2 [214]</td>
<td>2.23±0.85</td>
<td>3.88</td>
</tr>
<tr>
<td>ANFIS [86]</td>
<td>2.41±0.84</td>
<td>4.07</td>
</tr>
<tr>
<td>SparseFIS uncon. [41]</td>
<td>2.14±0.78</td>
<td>4.08</td>
</tr>
<tr>
<td>FLEXFIS [28]</td>
<td>2.17±0.73</td>
<td>3.59</td>
</tr>
<tr>
<td>Gen-Smart-EFS [58]</td>
<td>2.09±0.62</td>
<td>3.60</td>
</tr>
<tr>
<td>EFS-SLAT( \lambda = 0.8 )</td>
<td>2.097±0.393</td>
<td>2.836</td>
</tr>
<tr>
<td>EFS-SLAT( \lambda = 0.85 )</td>
<td>2.102±0.316</td>
<td>2.578</td>
</tr>
<tr>
<td>EFS-SLAT( \lambda = 0.9 )</td>
<td>2.092±0.395</td>
<td>2.866</td>
</tr>
<tr>
<td>EFS-SLAT( \lambda = 0.95 )</td>
<td>2.102±0.259</td>
<td>2.556</td>
</tr>
<tr>
<td>EFS-SLAT( \lambda = 0.99 )</td>
<td>2.088±0.195</td>
<td>2.386</td>
</tr>
</tbody>
</table>

The final model \( t = 6000 \) learned by EFS-SLAT\( (\lambda = 0.95) \) can be demonstrated by 2 fuzzy rules:

\[
R_1 : \text{If } x_1 \text{ is } \Gamma_{1,1}, \text{ and } x_2 \text{ is } \Gamma_{1,2}, \text{Then } y_1 = 0.0168 + 0.0195x_1 + 0.9334x_2; \quad (4.78)
\]

\[
R_2 : \text{If } x_1 \text{ is } \Gamma_{2,1}, \text{ and } x_2 \text{ is } \Gamma_{2,2}, \text{Then } y_2 = -0.0066 + 0.0624x_1 + 0.9738x_2. \quad (4.79)
\]

The centres and radii of the above \( R_1 \) and \( R_2 \) are:

\[
c_1 = [0.2837, 0.3467] \text{ and } \sigma_1 = [0.1226, 0.1144]; \quad (4.80)
\]

\[
c_2 = [0.1472, 0.1509] \text{ and } \sigma_2 = [0.0642, 0.0658]. \quad (4.81)
\]

### 4.4.3.5 Example 5: Data sets from UCI repository (term 10 in Section B.1, Appendix B)

MPG and Boston Housing data taken from UCI repository are used. Datasets are described in the term 10 in Section B.1, Appendix B. Following on [58] and [215], the
evaluation procedure follows a 10-fold cross-validation. The results mean absolute error (MAE) of EFS-SLAT in Table 4.12 are the best results based on 1000 Monte Carlo CV procedure; “STD” and the “rule” are the corresponding standard deviation towards the CV folds and the average rule numbers, respectively; Besides, “max MAE” is the corresponding maximum error towards all the data (the worst-case prediction error) [215]. The results of other methods are copied from [58], and they are picked as their best results. It can be seen from Table 4.12 that EFS-SLAT gives either higher accuracy, or smaller standard deviation and maximum MAE with a small amount of fuzzy rules.

4.5 Chapter Summary

By the aim of solving the thresholds setting problem of EFSs, two approaches SEFS and EFS-SLAT, have been proposed. SEFS has the threshold that controls the rule generation tuning automatically online relying on the risk of underfitting and overfitting reflected by the online training errors. Furthermore, a geometric-based similarity measure has been proposed in the merging process of SEFS. A direct, accurate and easily computed analytic formula of the $L^2$ distance between firing strengths of two rules has been induced and proposed to deal with the difficulty in calculating the $L^2$ distance, when membership functions are Gaussian type. Moreover, consequent parameters are updated by an improved WRLS method, a local version of VFF-RLS — VFF-WRLS.

As an enhanced approach of SEFS, EFS-SLAT has improved in the following aspects: Firstly, the online updating approach for the thresholds that control the rule generation and simplification are proposed from designing a strategy to make the rule generation and simplification to cooperate with but restrict to each other. Secondly, the geometric similarity measure have been proposed in SEFS has been enhanced with the Mahalanobis distance, in order to avoid the inappropriate merging of two small clusters. Lastly, EFS-SLAT has made decisions for the system evolving considering the information for the input and output, in order to avoid unnecessary structure change.
Chapter 5

Functional Fuzzy System — An Offline Function-on-Function Regression Approach

5.1 Chapter Introduction

Motivated by the limitations of existing research works of functional regressions, which have been demonstrated in the points 1 and 2 highlighted in Section 2.3.6.1, and inspired by the flexible structure of the fuzzy systems for data modelling, in this chapter, a new kind of fuzzy system — FFS is proposed. As a nonlinear functional regression model, FFS provides a new and novel pathway for learning the general function-on-function regression model with a nonlinear form. In order to identify an FFS, an FFS learning approach is proposed. In FFS learning approach, the k-means clustering approach is applied for grouping functional data (i.e., functions). The antecedent parameters are estimated based on functional mean and variance in the $L^2$ space (see Section 2.3.2.1). Least squares estimators of the consequent parameters are induced through minimizing the mean squared prediction errors. In order to verify that FFS is able to make accurate predictions for functions, the proposed FFS are compared with many existing state-of-the-art function-on-function regression models upon benchmark examples using both artificial and real datasets.

The novelties of FFS can be summarized as follows:
(i) FFS with function inputs and outputs is essentially a new type of fuzzy systems. FFS makes it possible to enlarge the input and output space of the fuzzy systems from finite dimension to infinite dimension, and enables us to model the nonlinear relationship between functional data. To be more specific, the shape of the curves conveyed by the data pairs from the domain and the target set is fully investigated. Furthermore, it provides another possible way for modelling big data using fuzzy system based model, through treating the discrete observations as curves.

(ii) By using FFS to model the relationship between curves, the predictions for curves (i.e., functions in certain time intervals) can be made. The predictions of curves are likely to provide more useful information than the traditional one- or multi-step ahead discrete predictions. People should benefit from the predictions of the whole picture of the output functions in their definition domains, and take corresponding actions based on the behaviour of the functions. This novelty of FFS can benefit many real applications. For instance, the full trajectory of a certain signal from the brain may help one to identify the illness; the curves of energy consumption may help power generators to avoid providing too little or too much energy, thus ensuring to meet the demand or saving resources.

(iii) Taking the advantages of fuzzy systems, FFS is able to model the non-linear relationship between functions. Furthermore, FFS has a flexible structure built as a collection of the “If-Then” fuzzy rules. These characteristics enable the FFS to use different rules for describing different behaviours of the relationship between functions, making it flexible enough to model the functional data. It thus enables FFS to be transparent and understandable.

The following content of this chapter is arranged as follows: Section 5.2 presents the main problems that will be solved in this chapter. The structure of the FFS and its detailed identification methodology are displayed in Section 5.3 and Section 5.4, respectively. In section 5.5, FFS is compared with the state-of-the-art functional regression models and approaches in the simulated and the real-life dataset.

---

5.2 Problem Statement

Generally, there are two problems need to solved:

1. **The full theory and the model of FFS should be constructed.** Based on the knowledge of fuzzy system, the definition of the functional fuzzy set, membership, and functional fuzzy rule should be given.

2. **In order to make FFS an applicable approach, learning approach that identifies FFS should be proposed.** To do so, there is a need to propose the approaches for identifying the rule number, antecedent parameters and consequent parameters.

5.3 Functional Fuzzy Systems

FFS is a fuzzy system with input and output in a functional space, which has a great difference with the conventional fuzzy systems with input and output in $\mathbb{R}^n$. Therefore, in order to construct FFS, some essential definitions such as functional fuzzy set and functional membership should be defined, firstly.

**Definition 5.1. Functional Fuzzy Set:** A functional fuzzy set $\Gamma$ in a given functional space $X (X \neq \emptyset)$ is a set of pairs:

$$
\Gamma = \{(x(s), \mu(x(s)))| x(s) \in X\}, \quad (5.1)
$$

where $\mu(x(s)) : X \rightarrow [0,1]$ is a functional membership mapping associated with $\Gamma$. It can indicate the membership degree of each element $x(s) \in X$ to the functional fuzzy set $\Gamma$.

**Definition 5.2. Gaussian Type of Mapping:** Assume $\Gamma = \{(x(s), \mu_\Gamma(x(s)))| x(s) \in X\}$ is a functional fuzzy set, the following (5.2) is known as a Gaussian type of mapping:

$$
\mu_\Gamma(x(s)) = \exp\left\{-\frac{||x(s) - c(s)||^2_X}{2\sigma^2}\right\}, \quad (5.2)
$$

where centre $c(s)$ is a functional, radius $\sigma$ is a real number, $\|\cdot\|_X$ is a norm on functional space $X$.

---

Remark that each $x(s) \in X$ is a functional.
Definition 5.3. **Functional Fuzzy Rule:** The general form of a functional fuzzy rule $R$ is defined as:

$$R: \text{If } x(s) \text{ is } \Gamma, \text{ Then } y(t) \text{ is } y^*(t),$$

(5.3)

where $x(s)$ and $y(s)$ are functionals, $\Gamma$ is the functional fuzzy set, $y^*(t)$ presents the exact functional form of $y(t)$.

With these definitions, it can define an FFS. An FFS $F(x(s)) : X \rightarrow Y$ is an operator that maps a functional space $X$ to another functional space $Y$. An $F(x(s))$ is formed as a collection of the “If-Then” functional fuzzy rules, i.e., $F(x(s)) : \{R_1, \ldots, R_k, \ldots, R_K\}$. Furthermore, assume that $X$ and $Y$ are subspaces of $L^2$ function spaces. The $k$-th functional fuzzy rule $R_k$ ($k = 1, 2, \ldots, K$) can be presented by (5.4):

$$R_k : \text{If } x(s) \text{ is } \Gamma_k, \text{ then } y^{(k)}(t) = \sum_{q=1}^{Q} h_{kq} B_q^Y(t),$$

(5.4)

where $x(s)$ is the function input and $y^{(k)}(t)$ is function output of the rule $R_k$, $\Gamma_k$ is the functional fuzzy set, $B_q^Y(t)$ ($q = 1, 2, \ldots, Q$) are the basis functions of $Y$ ($Y$ is the set formed as all the outputs $y(t)$), and $h_{kq} (q = 1, 2, \ldots, Q)$ are real consequent parameters. The functional membership of the $k$-th functional fuzzy set is the Gaussian type of mapping $\mu_k(x) : L^2 \rightarrow \mathbb{R}$ (5.5):

$$\mu_k(x) = \exp\left\{ -\frac{\|x(s) - c_k(s)\|_{L^2}^2}{2\sigma_k^2} \right\},$$

(5.5)

in which $\| \cdot \|_{L^2}$ is the $L^2$ norm; $c_k(s)$ is the clustering centre, which is a function; and $\sigma_k$ is the radius, which is a real number. The final output of the system can be computed by the weighted sum of the functional outputs $y^{(k)}(t)$ ($k = 1, 2, \ldots, K$), i.e., (5.6):

$$y(t) = F(x(s)) = \sum_{k=1}^{K} \theta_k(x(s)) y^{(k)}(t),$$

(5.6)

where $\theta_k(x(s)) = \frac{\mu_k(x(s))}{\sum_{k=1}^{K} \mu_k(x(s))}$ is the normalized firing strength of $R_k$. Note that the firing strength of rule $R_k$ is the membership $\mu_k(x(s))$. 


\textbf{CHAPTER 5. FUNCTIONAL FUZZY SYSTEM — AN OFFLINE FUNCTION-ON-FUNCTION REGRESSION APPROACH}  

\begin{figure}[h]  
\centering  
\includegraphics[width=\textwidth]{framework.png}  
\caption{The framework for identifying an FFS.}  
\end{figure}

\section{Learning Approach for FFS}

In the same way as the T-S and the Mamdani fuzzy systems discussed in the Chapter 3 and Chapter 4, FFS should be identified from three aspects: (1) Rule numbers \( K \); (2) Antecedent parameters including functional centres \( c_k(s) \) and real radii \( \sigma_k \); (3) Consequent real parameters \( h_{kq} \). After an FFS has been identified by the training data, this FFS will get ready to be used. This section provides details of the techniques concerning how to estimate these parameters. The FFS can be identified following the general framework shown in Fig. 5.1, in which each step is demonstrated in detail from Section 5.4.1 to Section 5.4.3.

\subsection{Data Preprocessing}

Assuming that \( \{B_p^X(s)\}_{p=1}^P \) and \( \{B_q^Y(t)\}_{q=1}^Q \) are the basis functions of subsets \( X \subset L^2 \) and \( Y \subset L^2 \), respectively. Therefore, for each input function \( x(s) \in X \) and output function \( y(t) \in Y \), \( \forall \varepsilon > 0 \), there exist \( \hat{x}(s) = \sum_{p=1}^P \alpha_p B_p^X(s) \) and \( \hat{y}(t) = \sum_{q=1}^Q \beta_q B_q^Y(t) \) such that \( \|x(s) - \hat{x}(s)\|_{L^2} < \varepsilon \) and \( \|y(t) - \hat{y}(t)\|_{L^2} < \varepsilon \). Furthermore, assume that the available functional data are \( N \) pairs of input and output functions \( \{(x_i(s), y_i(t))\}_{i=1}^N \), and the observations of each input \( x_i(s) \) and output \( y_i(t) \) are \( \{(s_j, x_i(s_j))\}_{j=1}^{J_i} \) and \( \{(t_l, y_i(t_l))\}_{l=1}^{L_i} \), respectively. Note that these \( J_i \) and \( L_i \) need not be the same, which will be further explained using the following example. Assume that everyday temperature curves \( \{x_i(s)\}_{i=1}^N \) are applied to predict everyday city electricity usage \( \{y_i(t)\}_{i=1}^N \). One one hand, if each temperature curve \( x_i(s) \) is recorded every 30 minutes, and each electricity curve \( y_i(t) \) is recorded hourly, then it can obtain \( J_i = 48 \) observations of each \( x_i(s) \) and \( L_i = 24 \) observations of \( y_i(t) \). It is obvious that \( 48 \neq 24 \). On the other hand, taking \( x(s) \) as an example, assume \( x_{i_1}(s) \) and \( x_{i_2}(s) \) are temperature curves of day \( i_1 \) and \( i_2 \), and \( \{(s_j, x_{i_1}(s_j))\}_{j=1}^{J_{i_1}} \) and \( \{(s_j, x_{i_2}(s_j))\}_{j=1}^{J_{i_2}} \) are respectively the observations on \( x_{i_1}(s) \) and \( x_{i_2}(s) \). If there
exist some data points on curve \( x_i(s) \) are forgot to be recorded (or missing), then \( J_i \neq 48 = J_i \). In order to effectively deal with these raw samples (observations) and preserve the information of the whole trajectories, preprocessing is essential for getting a group of “functions” which are intrinsically the linear combinations of the function bases.

\[
\{B_p^X(s)\}_{p=1}^P \text{ and } \{B_q^Y(t)\}_{q=1}^Q \text{ can be any basis: For example, spline basis, wavelet basis, fourier basis. The choice of the basis should depend on the data type. For instance, the Fourier basis usually performs better to fit the periodic data. Once a group of bases is selected, the trajectories of } x_i(s) \text{ and } y_i(t) \text{ will be approximated using } \sum_{p=1}^P \alpha_{i,p}B_p^X(s) \text{ and } \sum_{q=1}^Q \beta_{i,q}B_q^Y(t), \text{ respectively. In order to predetermine } P \text{ and } Q, \text{ the coefficients } \alpha_{i,p} \text{ and } \beta_{i,q} \text{ can be identified by the weighted least squares method with a regularization to control the smoothness. There exist many built-in functions in the programme software which can accomplish this task, e.g. the function “smooth.basis” in the “fda” package of R. } P \text{ and } Q \text{ have a crucial influence on the approximation accuracy of the bases. As } P \text{ and } Q \text{ are selected by the same procedure, the identification method for } P \text{ is shown as an example. } \{(s_j,x_i(s_j))\}_{j=1}^{J_i} \text{ are the samples for } x_i(s), \text{ then set } P_i(1) = \lfloor J_i/2 \rfloor \text{ and use } \{B_p^X(s)\}_{p=1}^{P_i} \text{ as the bases to fit the samples. At the } n\text{-th step, if the RMSE}(n) \text{ is large, then increase the number of bases to } P_i(n) = P_i(n-1) + \max(1,\lfloor J_i/2^n \rfloor). \text{ Otherwise, decrease the number of basis to } P_i(n) = P_i(n-1) - \max(1,\lfloor J_i/2^n \rfloor)\text{. Repeat the whole process until the accuracy reaches to a certain level or } P_i = J_i. \text{ To conclude, this process can be summarized by the algorithm 2. The thresholds in algorithm 2 are set as } \varepsilon_{\min} = \tau \cdot \min_i \{\max(X_i(s)) - \min(X_i(s))\} \text{ and } \varepsilon_{\max} = \tau \cdot \max_i \{\max(X_i(s)) - \min(X_i(s))\}, \text{ where } \tau \text{ is a tuning parameter.}

As a result, the uniform } P \text{ and } Q \text{ are elected by } P = \max_{i=1,\ldots,N} P_i \text{ and } Q = \max_{i=1,\ldots,N} Q_i. \text{ Based on the uniform bases } \{B_p^X(s)\}_{p=1}^P \text{ and } \{B_q^Y(t)\}_{q=1}^Q, \text{ the coefficients } \alpha \text{ and } \beta \text{ for the approximated } \{\hat{x}_i(s)\}_{i=1}^N \text{ and } \{\hat{y}_i(t)\}_{i=1}^N \text{ can be written as } \{(\alpha_{i,1},\alpha_{i,2},\ldots,\alpha_{i,P})\}_{i=1}^N \text{ and } \{(\beta_{i,1},\beta_{i,2},\ldots,\beta_{i,Q})\}_{i=1}^N.

### 5.4.2 Learning Method of Antecedent Parameters

This section presents the approach for learning the number of rules \( K \) and the antecedent parameters \( c_k(s) \) and \( \sigma_k \).

Assuming \( \{(x_i(s),y_i(t))\}_{i=1}^N \) with \( N \) pairs of input-output functions is the training
set. After using the data preprocess approach on this set, it can get the coefficients of the bases with regard to the \( x_i(s) \) and \( y_i(s) \) in real spaces \( \mathbb{R}^P \) and \( \mathbb{R}^Q \), respectively. Furthermore, it assumes that the coefficients obtained by approximating \( x_i(s) \) and \( y_i(t) \) are \( (\alpha_{1,1}, \alpha_{1,2}, \ldots, \alpha_{1,P}) \) and \( (\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,Q}) \), respectively. Therefore, the coefficient parameters of the input-output functions \( \{(x_i(s),y_i(t))\}_{i=1}^N \) can be presented as \( \{(\alpha_{1,1}, \alpha_{1,2}, \ldots, \alpha_{1,P}, \beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,Q})\}_{i=1}^N \). Afterwards, the problem of clustering the raw samples can be transformed into the problem of clustering the approximated functions. In a similar way to [107], k-means approach is applied on clustering the coefficients of input-output functions, \( \{(\alpha_{1,1}, \alpha_{1,2}, \ldots, \alpha_{1,P}, \beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,Q})\}_{i=1}^N \). Assume the set of functions grouped in cluster \( k \) is \( \{x_i(s),y_i(t)\}_{i=1}^{n_k} \), then the cluster centre of the cluster formed by \( \{x_i(s)\}_{i=1}^{n_k} \) is \( c_k(s) \) which can be computed by:

\[
c_k(s) = \frac{1}{n_k} \sum_{i=1}^{n_k} \{\alpha_{i,1}B_1^X(s) + \alpha_{i,2}B_2^X(s) + \ldots + \alpha_{i,P}B_P^X(s)\}.
\] (5.7)

Thus, the coefficients of \( c_k(s) \) are \( (c_{k,1}, c_{k,2}, \ldots, c_{k,P}) \) with \( c_{k,p} = \sum_{i=1}^{n_k} \alpha_{i,p} / n_k, \ p = 1,2,\ldots, P \). Furthermore, the radius \( \sigma_k \) of cluster \( k \) can be obtained by:

\[
\sigma_k = \sqrt{\frac{\sum_{i=1}^{n_k} ||x_i(s) - c_k(s)||^2_{L^2}}{n_k}}.
\] (5.8)

Because what can be observed is the finite number of samples \( \{x_i(s_j)\}_{j=1}^{l_i} \) on the trajectory of \( x_i(s) \), the \( L^2 \) distance \( ||x_i(s) - c_k(s)||_{L^2} \) in (5.8) can be approximated using the

---

**Algorithm 2:** Identify the number of basis.

```
repeat
    Use \( \{B_p^X(s)\}_{p=1}^{P_i} \) to fit \( \{(s_j,x_i(s_j))\}_{j=1}^{l_i} \), get \( RMSE \).
    if \( RMSE(n) < \epsilon_{\min} \) then
        if \( RMSE(n-1) > \epsilon_{\max} \) AND \( P_i(n) = P_i(n-1) + 1 \) then
            STOP
        else
            \( P_i(n) = P_i(n-1) - \max(1, \lfloor J_i/2^n \rfloor) \)
        end if
    else if \( RMSE(n) > \epsilon_{\max} \) then
        \( P_i(n) = P_i(n) + \max(1, \lfloor J_i/2^n \rfloor) \)
    end if
    until \( \epsilon_{\min} \leq RMSE(n) \leq \epsilon_{\max} \), or \( P_i(n) = J_i \), or \( P_i(n) = 1 \)
```
trapezoidal rule, and thus is calculated by:

\[
\|x_i(s) - c_k(s)\|_{L^2} \approx \sqrt{\int_S |x_i(s) - c_k(s)|^2 ds}
\]

\[
\approx \sqrt{\sum_{j=1}^{J_i-1} \left\{ (x_i(s_j) - c_k(s_j))^2 + (x_i(s_{j+1}) - c_k(s_{j+1}))^2 \right\} \triangle_j}
\]

where \( \triangle_j = s_{j+1} - s_j \), inf\((S) = s_1 < s_2 < \ldots < s_{J_i} = \text{sup}(S) \) is a partition of \( S \) (\( S \) is a closed interval in \( \mathbb{R} \)).

Considering the whole regression problem in the training process, the number of clusters \( K \) is determined based on the 10-fold cross-validation.

### 5.4.3 Learning Method of Consequent Parameters

The consequent parameters are estimated by the least squares method. The error function which needs to be minimized is shown in (5.10):

\[
Err = \frac{1}{N} \sum_{i=1}^{N} \int_{\Omega} |y_i(t) - \hat{y}_i(t)|^2 dt,
\]

where \( \hat{y}_i(t) = \sum_{k=1}^{K} \theta_k(x_i) y^{(k)}(t) = \sum_{k=1}^{K} \theta_k(x_i) \sum_{q=1}^{Q} h_{kq} B_q^y(t) \). Therefore, through vectorising the formula (5.10), the \( h_{kq} \) can be estimated by solving the following minimization problem:

\[
\min \frac{1}{N} \int_{\Omega} |y(t) - \hat{y}(t)|^2 dt
\]

\[
= \min \frac{1}{N} \int_{\Omega} (y(t) - \hat{y}(t))^T (y(t) - \hat{y}(t)) dt
\]

\[
= \min \frac{1}{N} \int_{\Omega} \{y(t) - \tilde{B}(t) h\}^T \{y(t) - \tilde{B}(t) h\} dt,
\]

\[
\text{In order to maximum the usage of the information contained in the real observations, the observations instead of the represented functions are used to compute the integral.}
\]
where \( y(t) = (y_1(t), y_2(t), \ldots, y_N(t))^T \), \( h = (h_{11}, h_{21}, \ldots, h_{K1}, \ldots, h_{1Q}, \ldots, h_{KQ})^T \), and \( \tilde{B}(t) \) is shown in (5.12):

\[
\tilde{B}(t) = (\tilde{B}_1(t), \tilde{B}_2(t), \ldots, \tilde{B}_Q(t)), \text{ with each } \tilde{B}_q(t) \text{ is:}
\]

\[
\tilde{B}_q(t) = \begin{pmatrix}
\theta_1(x_1)B_q(t) & \ldots & \theta_K(x_1)B_q(t) \\
\theta_1(x_2)B_q(t) & \ldots & \theta_K(x_2)B_q(t) \\
\vdots & \ddots & \vdots \\
\theta_1(x_N)B_q(t) & \ldots & \theta_K(x_N)B_q(t)
\end{pmatrix},
\]

\( q = 1, 2, \ldots, Q \). \( Q \) should depend on the data and should not to be too large, for example 10, 20, or 30, etc. The final estimation for \( h \) relies on the real observations of the \( y(t) \).

Two cases should be considered regarding the real situation of these observations. The first case is that if each function \( y_i(t) \) has the same number of observations, and \( t_l \) of these observations are evenly spaced knots (or at least has no relationship with \( i \)) on \( \Omega \), then, with the help of Riemann sum, the minimization problem shown in (5.11) can be transformed to a discrete form for minimizing the error function in \( \mathbb{R} \). This new error function can be presented as (5.14):

\[
\min \frac{1}{N} \int_{\Omega} |y(t) - \tilde{y}(t)|^2 dt 
\approx \min \frac{1}{N} \sum_{l=1}^{L} \{y(t_l) - \tilde{B}(t_l)\hat{h}\}^T \{y(t_l) - \tilde{B}(t_l)\hat{h}\} \Delta t, \tag{5.14}
\]

where \( t_l, l = 1, 2, \ldots, L \), are knots on \( \Omega \), \( \Delta t = t_l - t_{l-1} \). Then, the estimation formula of the consequent parameters \( h_{kq} \) is (5.15):

\[
\hat{h} = \left\{ \sum_{l=1}^{L} \tilde{B}(t_l)^T \tilde{B}(t_l) \Delta t \right\}^{-1} \left\{ \sum_{l=1}^{L} \tilde{B}(t_l)^T \{y(t_l) - \tilde{B}(t_l)\hat{h}\} \right\}. \tag{5.15}
\]

Only the simplest case can be computed by (5.15). However, in real applications, the number of observations of each \( y_i \) are likely to be unequal, and the corresponding \( t_l \) is likely to be unevenly spaced knots on \( \Omega \). This is the second situation that should be considered. In this case, a formula, which has few differences from (5.15), needs to be used in order to estimate the \( h \). Compute the derivative of (5.11) with regard to \( h \), then the formula of the least squares estimation of \( h \) can be presented by (5.16):

\[
\hat{h} = \left\{ \int_{\Omega} \tilde{B}(t)^T \tilde{B}(t) dt \right\}^{-1} \left\{ \int_{\Omega} \tilde{B}(t)^T \{y(t) - \tilde{B}(t)\hat{h}\} dt \right\}. \tag{5.16}
\]
CHAPTER 5. FUNCTIONAL FUZZY SYSTEM — AN OFFLINE FUNCTION-ON-FUNCTION REGRESSION APPROACH

Data preprocessing (section 5.4.1).

Training data (functions).

Testing data (functions).

Identify the FFS (following Fig. 5.1 in section 5.4).

Identified FFS.

Compute function predictions by (5.6).

Fig. 5.2: The training and testing procedure of an FFS.

Assuming there are $L_0$ evenly spaced knots on $\Omega$, $\Delta_t$ is the distance between two adjacent knots. Furthermore, assume $\{(t_1^{(i)}, y_1^{(i)}(t))\}_{i=1}^{L_i}$ are the observations of function $y_i(t)$; $t_1^{(i)} (l = 1, 2, \ldots, L_i)$ are the knots on $\Omega$, at which places have the observations. As a result, the $h$ can be computed by:

$$
\hat{h} = \{\sum_{l=1}^{L_0} \hat{B}(t_l)^T \hat{B}(t_l) \Delta_t\}^{-1} B, \quad (5.17)
$$

$$
B = (\sum_{i=1}^{N} a_1(x_i) B_1^{(i)}, \ldots, \sum_{i=1}^{N} a_K(x_i) B_1^{(i)}, \ldots, \sum_{i=1}^{N} a_1(x_i) B_Q^{(i)}, \ldots, \sum_{i=1}^{N} a_K(x_i) B_Q^{(i)})^T
$$

$$
B_q^{(i)} = \sum_{l=1}^{L_i-1} \{B_q(t_l+1) y_1(t_l+1) + B_q(t_l) y_1(t_l)\} (t_{l+1} - t_l)^{1/2}
$$

and $q = 1, 2, \ldots, Q$, $B = \{\int_\Omega B(t)^T y(t) dt\}$, and each $B_q^{(i)}$ is computed by the trapezoidal rule.

5.5 Numerical Examples

In this section, various examples acrossing the real applications and the simulations are used to test FFS and its corresponding identification approach. The training and testing procedure of FFS follow on from the framework in Fig. 5.2. The models are validated by the mean squared prediction errors (MSPEs) [169, 170]. The MSPE should be computed depending on the discrete observations and can be computed by:

$$
MSPE = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{L_i} \sum_{l=1}^{L_i} \{y_i(t_1^{(i)}) - \hat{y}_i(t_1^{(i)})\}^2, \quad (5.18)
$$
where $\hat{y}_i(t^{(i)}_l)$ are the predicted values of $y_i$ on $t^{(i)}_l$ computed by the determined FFS.

All the basis functions in the numerical examples, from Section 5.5.1 to Section 5.5.4, are chosen to be the spline basis. Furthermore, these spline bases can have orders $2, 3, 4$ (degree $1, 2, 3$) in the preprocessing procedure of each dataset. Moreover, based on the extensive numerical experiments, the predefined parameter $\tau$, in Section 5.4.1, is suggested to be a number between $10^{-2}$ and $10^{-4}$. The value of $\tau$ should neither be too small or too big in case of overfitting and underfitting problems. Thereafter, $\tau = 10^{-4}$ is set in the examples in Section 5.5.2 and Section 5.5.3; $\tau = 10^{-2}$ is set the first and the last example in section 5.5.1 and 5.5.4. The domains of the input and output functions are all transformed to $[0, 1]$.

Note that the state-of-the-art function-on-function regression models applied for comparisons in this section are: “linmod” [118] in the “fda” package, “pffr” [160], “aff” (“affpc”, “affs”, “flm”) [176], “SigCompFofHighDim” [170], and “sigmod” [169] in the package “FRegSigCom”. These packages are all publicly available in statistical software R. Additionally, the R code of FFS is currently available upon request. The “-”s appear in the numerical results (i.e., Table 5.4 and Table 5.3) for some approaches stand for the program failures of those selected approaches.

5.5.1 Simulation Example: Functional Nonlinear Model

In this simulation example, a group of data are generated from a functional nonlinear model. The whole dataset is constructed in 4 parts, each of which has a unique behaviour. To be more specific, four different functional linear models: $M_1, M_2, M_3$ and $M_4$ presented by (5.19) stand for the data behaviour of these 4 parts, separately. Following this train of thought, the data in this example are simulated using these four models. There are 1000 trajectories of $(x(t), y(t))$ pairs generated in total: 200 trajectories from each of $M_1$ and $M_3$, 300 trajectories from each of $M_2$ and $M_4$.

To explain in more detail, assume the input-output pairs $\{x_i(s), y_i(t)\}_{i=1}^{n_i}$ are formed by $\{x^{(1)}_i(s), y^{(1)}_i(t)\}_{i=1}^{n_1}, \{x^{(2)}_i(s), y^{(2)}_i(t)\}_{i=1}^{n_2}, \{x^{(3)}_i(s), y^{(3)}_i(t)\}_{i=1}^{n_3}$ and $\{x^{(4)}_i(s), y^{(4)}_i(t)\}_{i=1}^{n_4}$, which are generated from models $M_1, M_2, M_3$ and $M_4$ presented in (5.19), respectively.

$$M_k: y^{(k)}(t) = \int I \beta^{(k)}(s,t)x^{(k)}(s)ds + \varepsilon^{(k)}(t),$$  \hspace{1cm} (5.19)

where $I = [0, 1], M_k$ with $k = 1, 2, 3, 4$ stand for the models, $\beta^{(1)}(s,t) = \cos(t\pi/3)\sin(s\pi/3)$,
\[ \beta^{(2)}(s,t) = \frac{\sqrt{ts}}{1+t}, \quad \beta^{(3)}(s,t) = \exp\{-t(s)\}, \quad \beta^{(4)}(s,t) = 1 + st. \]  

The inputs \( x(s) \) are generated according to \( x(s) = \sum_{p=1}^{P} x_p B_p^X(s) \), where \( \{B_p^X(s)\}_{p=1}^{P} \) is the spline basis of \( X \) and \( \{x_p\}_{p=1}^{P} \) are the simulated coefficients. These coefficients of the inputs from the above four models are presented as \( \{(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(P)})\}_{i=1}^{n_1}, \{(x_i^{(2)}, x_i^{(2)}, \ldots, x_i^{(P)})\}_{i=1}^{n_2}, \{(x_i^{(3)}, x_i^{(3)}, \ldots, x_i^{(P)})\}_{i=1}^{n_3} \) and \( \{(x_i^{(4)}, x_i^{(4)}, \ldots, x_i^{(P)})\}_{i=1}^{n_4} \), and are generated from four multivariate Gaussian distributions \( N(\mu_1, \Sigma_1) \), \( N(\mu_2, \Sigma_2) \), \( N(\mu_3, \Sigma_3) \) and \( N(\mu_4, \Sigma_4) \), respectively. The covariance matrices are \( \Sigma_1 = \text{diag}(0.05) \), \( \Sigma_2 = \text{diag}(0.2) \), \( \Sigma_3 = \text{diag}(0.5) \) and \( \Sigma_4 = \text{diag}(0.7) \), and \( \mu_1, \mu_2, \mu_3 \) and \( \mu_4 \) are randomly generated from \( N(0,0.1), N(0.4,0.2) \), \( N(0.5,0.5) \) and \( N(1,0.1) \), respectively. Besides, the observations of \( y_i^{(k)}(t) \) are computed by \( y_i^{(k)}(t) = \sum_{j=1}^{J} \beta(s_j,t_i)x_i^{(k)}(s_j) \triangle_j + \epsilon^{(k)}(t_i) \), where \( \epsilon^{(k)}(t_i) \) are i.i.d variables drawn from \( N(0,0.001) \), \( \triangle_j = 1/J \).

Based on the aforementioned procedure, the simulated input observations and their corresponding output observations are \( \{(x_i^{(k)}(s_1), \ldots, x_i^{(k)}(s_n))\}_{i=1}^{n_k} \) and \( \{(y_i^{(k)}(t_1), \ldots, y_i^{(k)}(t_L))\}_{i=1}^{n_k} \), where \( k = 1, 2, 3, 4 \), \( n_1 = n_3 = 200 \), \( n_2 = n_4 = 300 \), \( P = 20 \), \( J = 100 \), \( L = 50 \). The order of the basis splines are chosen to be 2, 3, 4. After shuffling the dataset, the first 500 input-output pairs are used for training, whilst the remaining 500 pairs are applied as the testing data.

![Coefficients of x(s).](image-url)

**Fig. 5.3:** Simulated coefficients of inputs data.

To make it more visible for the dataset, Fig. 5.3 shows the randomly generated coefficients of the splines in a 2-D point plot as well as the real trajectories of the input curves generated from \( M1, M2, M3, M4 \); To be more specific, Fig. 5.6 displays the real trajectories of \( x(s) \) generated from those four models with order 2, 3, 4. The three figures in the first row of Fig. 5.7 present the 2-D point plots of \( \{(x_i^{(k)}(s_1), \ldots, x_i^{(k)}(s_j), y_i^{(k)}(t_1), \ldots, y_i^{(k)}(t_L))\}_{i=1}^{n} \) with \( k = 1, 2, 3, 4 \), whereas the 3 figures at the bottom of Fig. 5.7 show the...
FFS is compared with various state-of-the-art approaches by 50 replications of the experiment. Accuracies obtained from the algorithms are listed in Table 5.4. It can be seen from this table that FFS can always provide more accurate results than other state-of-the-art algorithms no matter which order of the spline bases are used for generating $x(s)$.

![Insert Table 5.4 here!]

![Insert Fig. 5.6 here!]

![Insert Fig. 5.7 here!]

### 5.5.2 Diffusion Tensor Imaging (DTI) Data

Diffusion tensor imaging (DTI) data are introduced in Term 1 of Section B.2, Appendix B. CCA-FA curves are used as the inputs $x_i(s)$ of the model, and the RCST-FA curves are treated as the output curves $y_i(t)$. The first 100 CCA-FA and RCST-FA pairs are used for training, and the remaining 155 pairs are applied for testing.

**TABLE 5.1: Example 2: DTI data prediction.**

<table>
<thead>
<tr>
<th>Methods</th>
<th>MSPEs (train) $\times 10^{-2}$</th>
<th>MSPEs (test) $\times 10^{-2}$</th>
<th>Rule num.</th>
</tr>
</thead>
<tbody>
<tr>
<td>linmod [118]</td>
<td>0.1517</td>
<td>0.6863</td>
<td>-</td>
</tr>
<tr>
<td>pffr [160]</td>
<td>0.3172</td>
<td>0.4447</td>
<td>-</td>
</tr>
<tr>
<td>affpc [176]</td>
<td>0.3364</td>
<td>0.4523</td>
<td>-</td>
</tr>
<tr>
<td>affs [176]</td>
<td>0.3026</td>
<td>0.5048</td>
<td>-</td>
</tr>
<tr>
<td>flm [176]</td>
<td>0.3373</td>
<td>0.4532</td>
<td>-</td>
</tr>
<tr>
<td>SigCompFofHighDim [170]</td>
<td>0.2727</td>
<td>0.4476</td>
<td>-</td>
</tr>
<tr>
<td>sigmod [169]</td>
<td>0.2548</td>
<td>0.4613</td>
<td>-</td>
</tr>
<tr>
<td>FFS (order=2)</td>
<td>0.3068</td>
<td>0.4404</td>
<td>6</td>
</tr>
<tr>
<td>FFS (order=3)</td>
<td>0.3073</td>
<td>0.4404</td>
<td>6</td>
</tr>
<tr>
<td>FFS (order=4)</td>
<td>0.3311</td>
<td>0.4609</td>
<td>5</td>
</tr>
</tbody>
</table>

The comparison results for FFS and the state-of-the-art models and approaches are listed in Table 5.1. It is obvious that the results shown in Table 5.1 demonstrate that

---

4Table 5.4, Fig. 5.6 and Fig. 5.7, are presented at the end of this chapter.
FFS can make preferable prediction results. Especially, when using the spline basis with order 2 or 3 to fit the original observations, FFS can obtain more accurate results than all the other methods used as comparisons using a small number of rules.

Due to the space limitation for presenting the mathematical formulas of the EFFS (order=4) learned functional fuzzy rules, alternatively, the centre and consequent functions of the functional fuzzy rules are demonstrated in Fig. 5.4. The corresponding radii of these 6 rules are: 0.0017, 0.0017, 0.0021, 0.0022, 0.0018, 0.0006.

### 5.5.3 Capital Bike Share Data

The dataset used in this example is described in Term 2, Section B.2, Appendix B. The first 60 input-output curve pairs construct the training set, and the remaining 42 pairs form the testing set. It can be seen from Table 5.2 that FFS can make preferable prediction results. Especially, when using the spline basis with order 2 or 3 to fit the original observations, FFS uses small number of rules, and performs better than other state-of-the-art approaches.

In order to show the interpretability of FFS, the centres and consequents of the learned 6 functional fuzzy rules are presented in Fig. 5.5. The radii of these rules are: 6.8981, 3.9217, 4.7057, 6.8024, 4.1883, 4.5373.
TABLE 5.2: Example 3: Capital bike share data prediction.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MSPEs (train)</th>
<th>MSPEs (test)</th>
<th>Rule num.</th>
</tr>
</thead>
<tbody>
<tr>
<td>linmod [118]</td>
<td>0.3638</td>
<td>0.5499</td>
<td>-</td>
</tr>
<tr>
<td>pffr [160]</td>
<td>0.4209</td>
<td>0.5103</td>
<td>-</td>
</tr>
<tr>
<td>affpc [176]</td>
<td>0.4268</td>
<td>0.3929</td>
<td>-</td>
</tr>
<tr>
<td>affs [176]</td>
<td>0.4092</td>
<td>0.4018</td>
<td>-</td>
</tr>
<tr>
<td>f1m [176]</td>
<td>0.5102</td>
<td>0.5553</td>
<td>-</td>
</tr>
<tr>
<td>SigCompFofHighDim [170]</td>
<td>0.4035</td>
<td>0.5197</td>
<td>-</td>
</tr>
<tr>
<td>sigmod [169]</td>
<td>0.4153</td>
<td>0.5180</td>
<td>-</td>
</tr>
<tr>
<td>FFS (order=2)</td>
<td>0.3837</td>
<td>0.3928</td>
<td>6</td>
</tr>
<tr>
<td>FFS (order=3)</td>
<td>0.3837</td>
<td>0.3928</td>
<td>6</td>
</tr>
<tr>
<td>FFS (order=4)</td>
<td>0.3685</td>
<td>0.4046</td>
<td>6</td>
</tr>
</tbody>
</table>

Fig. 5.5: Centres and consequents of rules.
5.5.4 Electricity Consumption of France

This example is used to learn the relationship between the temperatures and the electricity usage of France. The dataset and background for this example is specifically introduced in the term 3 in Section B.2, Appendix B. The training data are 300 pairs of curves randomly selected from the 361 curves of temperature and the electricity consumption datasets, and the remaining 61 pairs are used for testing. The final numerical results reported in Table 5.3 are based on the mean of 50 Monte Carlo simulations. It can be observed that FFS provides a higher prediction accuracy, no matter which order of the splines (2, 3 or 4) are used for fitting the original samples. This example presents FFS has the same behaviour as it in the previous three examples. FFS can achieve better performance when using the lower order spline basis to represent the discrete observations as the functional data. The possible reason behind this is that the splines with higher order are more smooth than the splines with lower order. Thus, they are insufficient for fitting the roughness and the slight fluctuations of the curves.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MSPEs (train)×10^{-2}</th>
<th>MSPEs (test)×10^{-2}</th>
<th>Rule num.(ave.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>linmod [118]</td>
<td>1.1936</td>
<td>1.2901</td>
<td>-</td>
</tr>
<tr>
<td>pffr [160]</td>
<td>1.2759</td>
<td>1.1496</td>
<td>-</td>
</tr>
<tr>
<td>affpc [176]</td>
<td>1.0371</td>
<td>1.0625</td>
<td>-</td>
</tr>
<tr>
<td>affs [176]</td>
<td>0.9741</td>
<td>1.0245</td>
<td>-</td>
</tr>
<tr>
<td>flm [176]</td>
<td>1.3290</td>
<td>1.1975</td>
<td>-</td>
</tr>
<tr>
<td>SigCompFofHighDim [170]</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sigmod [169]</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FFSs (order=2)</td>
<td>0.9047</td>
<td>0.9508</td>
<td>14.56</td>
</tr>
<tr>
<td>FFSs (order=3)</td>
<td>0.8995</td>
<td>0.9518</td>
<td>14.36</td>
</tr>
<tr>
<td>FFSs (order=4)</td>
<td>0.8997</td>
<td>0.9543</td>
<td>14.52</td>
</tr>
</tbody>
</table>

5.6 Chapter Summary

This chapter has proposed a new type of fuzzy system — FFS. FFS is suitable to learn nonlinear function-on-function regression problems. The achievements made by FFS can be summarized as: (i) Taking the advantage of treating data samples as function values, instead of discrete data points, FFS expands the fuzzy systems from dealing with
data points in the finite dimensional real space $\mathbb{R}^n$ to the infinite dimensional function space $L^2$, and provide a new fuzzy system for handling big data using fuzzy system based models. (ii) FFS is, in essence, a nonlinear functional regression model which is able to learn the underlying nonlinear relationships between curves instead of regarding and assuming that they will fit the linear model (or a certain type of nonlinear model). (iii) Benefitting from the simple structure and the inference procedure of the “If-Then” fuzzy rules, FFS can provide a flexible and interpretive approach for solving function-on-function regression models.
### TABLE 5.4: Example 1: Simulation example.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Spline order of $x(s)$ is 2</th>
<th>Spline order of $x(s)$ is 3</th>
<th>Spline order of $x(s)$ is 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSPEs (train) $\times 10^{-2}$</td>
<td>MSPEs (test) $\times 10^{-2}$</td>
<td>Rule num</td>
</tr>
<tr>
<td>linmod [118]</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pffr [160]</td>
<td>2.2306</td>
<td>2.6093</td>
<td>-</td>
</tr>
<tr>
<td>affpc [176]</td>
<td>1.2049</td>
<td>1.2368</td>
<td>-</td>
</tr>
<tr>
<td>affs [176]</td>
<td>1.0738</td>
<td>1.2368</td>
<td>-</td>
</tr>
<tr>
<td>flm [176]</td>
<td>2.4575</td>
<td>2.8566</td>
<td>-</td>
</tr>
<tr>
<td>SigCompFof-HighDim [170]</td>
<td>1.9531</td>
<td>2.3251</td>
<td>-</td>
</tr>
<tr>
<td>sigmod [169]</td>
<td>1.9029</td>
<td>2.2924</td>
<td>-</td>
</tr>
<tr>
<td>FFS (order=2)</td>
<td>0.9165</td>
<td>1.0441</td>
<td>4.98</td>
</tr>
<tr>
<td>FFS (order=3)</td>
<td>0.9143</td>
<td>1.0427</td>
<td>4.98</td>
</tr>
<tr>
<td>FFS (order=4)</td>
<td>0.9105</td>
<td>1.0379</td>
<td>4.98</td>
</tr>
</tbody>
</table>
Fig. 5.6: Simulated inputs data.
Fig. 5.7: Simulated input-output pairs and outputs.
Chapter 6

Evolving Functional Fuzzy System — An Online Function-on-Function Regression Approach

6.1 Chapter Introduction

FFS has been developed in Chapter 5. However, the FFS is an offline model as the existing works which need to undertake extensive computational jobs when handling the big functional data or functional streaming data. In this chapter, further progress that has been made based on FFS is presented. Motivated by the idea of the online manner of the EFSs, an online function-on-function regression model known as EFFS has been developed. This allows big functional data coming as data streams and processing them online. In order to make EFFS applicable, an online learning approach is proposed for identifying EFFS dynamically. This learning approach is proposed based on the basic framework of EFS online learning approaches: Firstly, functional fuzzy rule generation and merging theory is developed; and Secondly, WRLS for learning consequent parameters of EFFS is induced. EFFS is tested on both classic benchmark examples and real-life datasets. The numerical results further demonstrate the effectiveness of EFFS.

The novelties of EFFS can be summarized as: EFFS inherits all the merits from FFS, but makes an improvement by providing a swift way to process big functional data and functional streaming data. Furthermore, EFFS is a self-organized system and can
start from an empty rule base. It can expand and shrink the rule base, and tune the parameters dynamically depending on the knowledge learnt from the rapidly coming data. Moreover, to the best of our knowledge, EFFS is the first approach that considers solving functional regression problems based on an evolving fuzzy framework.

The rest of this chapter\(^1\) is organized as follows: Section 6.2 presents the exact problems that need to be solved in this chapter. Section 6.3 presents the general form of the EFFS and the corresponding identification problems. The online learning approaches including structure and parameters learning for EFFS are shown in Section 6.4. In Section 6.5, the EFFS is compared with many state-of-the-art approaches and used on real-world examples. Conclusions are made in Section 6.6.

### 6.2 Problem Statement

In order to give a clear presentation of the work in this chapter, the following list gives a brief summary of problems that are solved in the following chapters:

1. **How to construct the framework of EFFS?** To be more specific, the fuzzy rules used by EFFS should be given.

2. **How to learn the parameters that needed for identifying an EFFS?** In order to identify an EFFS, antecedent parameters and consequent parameters should be learnt. Taking the thought of EFS, approaches for rule generation and rule simplification should be proposed.

### 6.3 Evolving Functional Fuzzy Systems

An EFFS is a time-varying operator \( F(x(s)) : X \to Y \) that maps a functional space \( X \) to another functional space \( Y \). An EFFS is consisted of a set of “If-Then” functional fuzzy rules with inputs and outputs being curves from functional spaces \( X \subset L^2 \) and \( Y \subset L^2 \). Assume input \( x(s) \in \Omega_X \) and output \( y(t) \in \Omega_Y \) are functions \( x(s) = \sum_{p=1}^{P} \alpha_p B^X_p(s) \)

\(^{1}\)This chapter is written based on our submitted manuscript: D. Ge, & X. J. Zeng, “Learning functional data online: an evolving functional fuzzy system approach”, IEEE Transactions on Cybernetics, 2019.
and \( y(t) = \sum_{q=1}^{Q} \beta_q B^Y_q(t) \), where \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_P) \) and \( \beta = (\beta_1, \beta_2, \ldots, \beta_Q) \) are coefficients, \( B^X(s) = (B^X_1(s), B^X_2(s), \ldots, B^X_P(s))^T \) and \( B^Y(t) = (B^Y_1(t), B^Y_2(t), \ldots, B^Y_Q(t))^T \) are the basis functions of \( X \) and \( Y \), respectively. The form of a single functional fuzzy rule \( R_k \) can be presented by (6.1) below:

\[
R_k: \text{If } x(s) \text{ is } \Gamma_k, \text{ Then } y^{(k)}(t) = \sum_{q=1}^{Q} h_{kq} B^Y_q(t),
\]

in which \( \Gamma_k \) is the fuzzy set, \( h_{kq} \) are real consequent parameters, \( B^Y_1(t), B^Y_2(t), \ldots, B^Y_Q(t) \) are the basis functions of \( Y \subset L^2 \). The functional membership of \( x(s) \) in the fuzzy sets \( \Gamma_k \) (\( k = 1, 2, \ldots, K \)) are Gaussian type of mapping \( \mu_k(x(s)) \):

\[
\mu_k(x(s)) = \exp\left\{ -\frac{\|x(s) - c_k(s)\|_{L^2}^2}{2\sigma_k^2} \right\},
\]

where \( c_k(s) \in L^2 \) are clustering centres, which are functions, and \( \sigma_k \in \mathbb{R} \) are the radii, which are real numbers. The firing strengths of \( R_k \) are \( \gamma_k(x(s)) = \mu_k(x(s)) \), \( k = 1, 2, \ldots, K \). With these basic knowledge, the output of the evolving functional fuzzy system is:

\[
y(t) = F(x(s)) = \sum_{k=1}^{K} \theta_k(x(s)) y^{(k)}(t),
\]

in which \( \theta_k(x(s)) \) are the normalized firing strengths which can be computed by (6.4):

\[
\theta_k(x(s)) = \frac{\gamma_k(x(s))}{\sum_{j=1}^{K} \gamma_j(x(s)).}
\]

Until here, it is hard to find any difference between FFS and EFFS. The major difference between an FFS and an EFFS is: FFS has a fixed number of fuzzy rules, and fixed parameters \( (c_k(s), \sigma_k \text{ and } h_{kq}) \); however, EFFS has the rule number \( K \), every parameter \( c_k(s), \sigma_k \text{ or } h_{kq} \) varying time-to-time. Therefore, unlike FFS discussed in Chapter 5, the problem of identifying an EFFS should be an online learning problem. It means that there is a need to develop the methodology for learning the adaptive structure and parameters of EFFS in an online manner.
6.4 Online Learning Method for Evolving Functional Fuzzy Systems

In order to achieve the aim of learning the large functional datasets, one approach is the streaming processing. This makes the data come as data streams with each “data point” containing the full observations of a curve. The learning approach for the EFFS can be divided into two major modules: antecedent parameters and structure learning, and consequent parameters learning. Section 6.4.1 presents the approaches for identifying the antecedent parameters \( (c_k(s) \text{ and } \sigma_k) \) and the rule numbers \( K \). To be more specific, it focuses on how to update and evolve the EFFS on the fly. Section 6.4.2 shows the weighted recursive least square (WRLS) formulas induced from minimizing the local error functions for learning the consequent parameters \( h_{kq} \).

6.4.1 Learning The Antecedent Parameters and Structures

The rule base of an EFFS should grow from an empty set. The EFFS should evolve from two directions simultaneously. The first direction is expanding the rule base to enhance the model accuracy. When new information comes and can not be explained by the existing rules, the rule base should be expanded to enable to explain the new knowledge. The second direction is shrinking the rule base in order to lower down the computational burden and make the model more interpretable. As fuzzy rules are frequently calibrated and updated, there are likely to be some fuzzy rules growing very similar. In this case, these similar fuzzy rules should be merged in order to simplify the system, avoid rule conflict and relief the computational burden. Apart from the structure updating, the parameters should also be learnt and updated. Based on the knowledge of statistics, the cluster centres and radii are learnt by computing the mean and variance of the curves.

Assuming the new coming input-output pair is \( (x_N(s), y_N(t)) \). The very first step is to fit the discrete observations on the curves \( x_N(s) \) and \( y_N(t) \) into functional type with

\[
x_N(s) = \sum_{p=1}^{P} \alpha_{N,p} B_X^p(s) \quad \text{and} \quad y_N(t) = \sum_{q=1}^{Q} \beta_{N,q} B_Y^q(t),
\]

where \( B_X^p(s) \) and \( B_Y^q(t) \) are basis functions. Note that this process is nothing different from the preprocessing presented in Section 5.4.1, Chapter 5, but using a predefined number of basis for both \( B_X^p(s) \) and \( B_Y^q(t) \).
6.4.1.0.1 Rule Generation and Updating Approach

The criteria that is applied for deciding generate or update rules is shown in Criteria 6.1.

Criterion 6.1. If \( \exists k^* \in \{1, 2, \ldots, K\} \) s.t. \( \gamma_{k^*}(x_N(s)) \geq \epsilon \) \( \left( k^* = \arg \max_k \gamma_k(x_N(s)) \right) \), then \( x_N(s) \) should be used to update the parameters \( c_{k^*}(s) \) and \( \sigma_{k^*} \) by (6.5) and (6.6); otherwise, generate a new fuzzy rule \( R_{K+1} \) with centre \( c_{K+1}(s) \) and radius \( \sigma_{K+1} \) are computed by (6.7) and (6.8).

\[
\tilde{c}_{k^*}(s) = \frac{n_{k^*}c_{k^*}(s) + x_N(s)}{n_{k^*} + 1}, \tag{6.5}
\]
\[
\tilde{\sigma}_{k^*}^2 = \frac{n_{k^*}}{n_{k^*} + 1} \left\{ \sigma_{k^*}^2 + ||c_{k^*}(s) - \tilde{c}_{k^*}(s)||_2^2 + \xi ||x_N(s) - \tilde{c}_{k^*}(s)||_2^2 \right\}, \tag{6.6}
\]

where \( n_{k^*} \) is the number of data points located in the cluster formed by the rule \( R_{k^*} \), \( \xi \) is a weight for balancing the influence of the distances between the centres and radii.

\[
c_{K+1}(s) = x_K(s), \tag{6.7}
\]
\[
\sigma_{K+1} = \frac{1}{K} \sum_{k=1}^{K} \sigma_k, \tag{6.8}
\]

in which \( \sigma_{K+1} \) is initialized by the average of the radii of the existing rules. Besides, the consequent parameters of rule \( R_{K+1} \) can be initialized by \( h_{K+1} = \frac{\sum_{k=1}^{K} h_k}{K} \).

Note that in Criteria 6.1, the \( L^2 \) norm in (6.6) is calculated numerically applying the trapezoidal rule, and the detailed computation steps are shown in Section A.3.1, Appendix A.

6.4.1.0.2 Rule Merging Approach

Supposing that rule \( R_{i_1} \) and rule \( R_{i_2} \) grow to be similar, then these two rules should be merged depending on the following Criteria 6.2:

Criterion 6.2. If \( S(R_{i_1}, R_{i_2}) \geq \epsilon \) \( \left( m \right) \), then \( R_{i_1} \) and \( R_{i_2} \) should be merged as one fuzzy rule. \( S(R_{i_1}, R_{i_2}) \) is the similarity measure, and it can be computed by (6.9),

\[
S(R_{i_1}, R_{i_2}) = \exp\{ -||c_{i_1}(s) - c_{i_2}(s)||_2^2 - ||\sigma_{i_1} - \sigma_{i_2}||_2 \}. \tag{6.9}
\]

\(^2\)Similar similarity measure can also be found from [55, 56].
Rule merging only happens in the situation when a fuzzy rule is being updated. Furthermore, it merges $R_{k^*}$ ($k^* = \arg \max_{k} \gamma_k(x_N(s))$) with its nearest rule $R_{k'}$ ($k' = \arg \max_{k} S(R_{k^*}, R_k)$). Assuming the rule $R_{k''}$ is the rule merged by $R_{k^*}$ and $R_{k'}$, the centre and radius of $R_{k''}$ can be computed by the (6.10) and (6.11), respectively. A similar method can be found in e.g., [39].

\[
c_{k''}(s) = \frac{1}{n_{k^*} + n_{k'}} (n_{k^*} c_{k^*}(s) + n_{k'} c_{k'}(s))
\]

(6.10)

\[
\sigma_j^2 = \frac{1}{n_{k^*} + n_{k'}} (n_{k^*} \sigma_{j,k^*}^2 + n_{k'} \sigma_{j,k'}^2 + n_{k^*} \| c_{k^*}(s) - c_{k''}(s) \|^2_{L^2} + n_{k'} \| c_{k'}(s) - c_{k''}(s) \|^2_{L^2}),
\]

(6.11)

where $\| c_{k^*}(s) - c_{k''}(s) \|^2_{L^2}$ and $\| c_{k'}(s) - c_{k''}(s) \|^2_{L^2}$ can be computed using the same method as $\| c_k(s) - c_{k}(s) \|^2_{L^2}$ in (6.6). Furthermore, the consequent parameters of $R_{k''}$ should be initialized by $h_{k''} = (n_{k^*} h_{k^*} + n_{k'} h_{k'}) / (n_{k^*} + n_{k'})$. Once $R_{k^*}$ and $R_{k'}$ are merged to $R_{k''}$, the rule $R_{k'}$ will be removed and the rule $R_{k''}$ will be replaced by $R_{k''}$.

### 6.4.2 Learning Consequent Parameters

Consequent parameters are computed in order to minimize the local error functions $E_k$, $k = 1, 2, \ldots, K$, presented in (6.12):

\[
E_k = \sum_{i=1}^{N} \theta_k(x_i) \int_{\Omega} \left\{ y_i(t) - (B^T(t))^T h_k \right\}^2 dt.
\]

(6.12)

In order to ensure the EFFS suitable for online computing, there is a need to induce the recursive formulas to calculate $h_k$. In this case, the optimization procedure can go through swiftly without storing a huge amount of historical data. Before presenting the proposed recursive formulas, it should assume $\hat{h}_k(N)$ to be the estimated consequent parameters of the rule $R_k$ applying all the data pairs $(x_i(s), y_i(t))$ ($i = 1, 2, \ldots, N$). Minimizing (6.12), a group of recursive formulas to update and compute $h_k$ are shown in the formulas from (6.13) to (6.16):

\[
P(N) = \Lambda \{ \Lambda + 1 \}^{-1} P(N - 1),
\]

(6.13)

\[
\Lambda = \theta_k(x_N) P(N - 1) \int B^T(t)(B^T(t))^T dt,
\]

(6.14)

\[
\hat{h}_k(N) = \hat{h}_k(N - 1) + P(N) \theta_k(x_N) \int B^T(t)e_N(t) dt,
\]

(6.15)
\[ \varepsilon_N(t) = y_N(t) - (B^Y(t))^T \hat{h}_k(N - 1), \]  

(6.16)

in which \( I \) is an identity matrix, \( P \) is initialized by \( P(0) = \Omega I_{Q \times Q} \), where \( \Omega \) can be selected as a big number, e.g., 1000 and 100000. The mathematical induction and calculation for (6.13) — (6.16) can be found in Section A.3.2, Appendix A. Integrals \( \int B^Y(t)(B^Y(t))^T \, dt \) and \( \int B^Y(t) \varepsilon_N(t) \, dt \) are calculated numerically by the trapezoidal rule, details of which are shown in Section A.3.3, Appendix A.

---

**Algorithm 3: EFFS learning approach.**

| Input: | \( F_{N-1}(x(s)) \), \( \{(s_j, x_N(s_j))\}_{j=1}^{J_N} \) and \( \{(t_l, y_N(t_l))\}_{l=1}^{L_N} \) |
| Output: | \( F_N(x(s)) \), \( y_{\text{train}}^N(t) \) and \( y_{\text{test}}^N(t) \) |

Represent data by: \( x_N(s) = \sum_{p=1}^{P} \alpha_{N,p} B_p^Y(s) \) and \( y_N(t) = \sum_{q=1}^{Q} \beta_{N,q} B_q^Y(t) \).

Compute online testing result: \( \hat{y}_N^\text{test}(t) = F_{N-1}(x_N(s)). \)

for \( k = 1 \) to \( K \) do

Compute \( \gamma_k(x_N(s)) \).

Select \( k^* = \arg \max \gamma_k(x_N(s)). \)

if \( \gamma_k(x_N(s)) \geq \varepsilon^{(a)} \) then

Update rule \( R_k \) using (6.5) and (6.6).

if \( \exists k' \neq k^* \) s.t. \( S(R_k, R_{k'}) > \varepsilon^{(m)} \) then

Merge \( R_k^* \) and \( R_{k'} \) to \( R_{k''} \).

end if

else

Add a new rule \( R_{K+1} \) following from Criterion 6.1.

end if

Update consequent parameters using (6.13)–(6.16), and obtain \( F_N(x(s)) \).

Compute online training result: \( y_{\text{train}}^N(t) = F_N(x_N(s)) \).

---

To summarize, the learning framework of EFFS can be further demonstrated by Fig. 6.1. In Fig. 6.1, \( F(N-1) \) and \( F(N) \) stand for the functional fuzzy systems learnt from \( \{(x_i(s), y_i(t))\}_{i=1}^{N-1} \) and \( \{(x_i(s), y_i(t))\}_{i=1}^{N} \), respectively. The learning process of EFFS is shown inside the red dashed rectangle in Fig. 6.1.

Based on the rule generation and simplification steps and the framework in Fig. 6.1, the sudo code of the EFFS learning algorithm is given in Algorithm 3. Assume \( F_{N-1}(x(s)) \) is the EFFS obtained from updating using the \( N-1 \)-th input-output function pairs \( (x_{N-1}(s), y_{N-1}(t)) \), the new coming raw observations of \( (x_N(s), y_N(t)) \) are \( \{(s_j, x_N(s_j))\}_{j=1}^{J_N} \) and \( \{(t_l, y_N(t_l))\}_{l=1}^{L_N} \), \( y_N^\text{test}(t) \) is the online testing result, and \( y_N^\text{train}(t) \)
CHAPTER 6. EVOLVING FUNCTIONAL FUZZY SYSTEM — AN ONLINE
FUNCTION-ON-FUNCTION REGRESSION APPROACH

Read \((x_N(s), y_N(t))\). Testing results: \(\hat{y}_N(t)\).

The learnt \(F(N - 1)\).

\[ \exists k \text{ s.t. } \gamma_k(x_N(s)) > \epsilon(a) \]

Generate new rule by Criterion 6.1.

Update existing rule by Criterion 6.1.

Merge rules using Criterion 6.2.

Update consequent parameters and obtain \(F(N)\).

The framework of EFFS.

Fig. 6.1: The framework of EFFS.

is the online training result, \(P\) and \(Q\) are the predefined number of bases.

6.5 Numerical Experiments

EFFS is tested on the benchmark examples and compared with the state-of-the-art approaches. In real applications, the data are collected as observations on each curve of the input \(x(s)\) and output \(y(t)\). Assume that the observations of discrete data points on the input curves are \(\{(x_i(s_{i1}), x_i(s_{i2}), \ldots, x_i(s_{iL_i}))\}_{i=1}^N\) and \(L_i\) may different from each other; and observations for the output curves are \(\{(y_i(t_{i1}), y_i(t_{i2}), \ldots, y_i(t_{iT_i}))\}_{i=1}^N\) and \(T_i\) may be different from each other. Furthermore, these observations are very likely to have missing data, and the time stamps \(s\) and \(t\) are mapped into \([0, 1]\). Basis splines are applied to fit the original observations, in order to obtain the functional form of the \(x(s)\) and \(y(t)\). As there is no enough information about \(\xi\) in similarity measure, \(\xi = 1\) is set in the numerical examples. The evaluation criteria based on the original observations are
the MSPEs shown in (5.18) in Section 5.5, Chapter 5.

In the following examples, the number of basis splines is set to be 10 or 20 for the input and the output. The number of spline basis is kept the same for all the approaches used for comparison. The spline order used by EFFS is chosen from 2, 3, 4; the thresholds $\varepsilon_a$ and $\varepsilon_m$ are varying from (0.1, 0.2, 0.3, 0.4, 0.5, 0.6) and (0.95, 0.96, 0.97, 0.98), respectively. Whereas, in Example 1 and 2, only the best results are presented in this section, the full details of the results with regard to different combinations of the thresholds will be presented in Section A.4.2, Appendix A. In example 3, all the results obtained from these thresholds are given. Furthermore, there are no big gaps between the results based on a different combination of the parameters. Also, EFFS is running in an online mode across all these three examples.

6.5.1 Example 1: Electricity Usage Daily Curves of France

| TABLE 6.1: Example 1: Result comparison of different methods. |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Method                          | No. of Basis = 10 |                | No. of Basis = 20 |                |                |
|                                 | MSPE (train)     | MSPE (test)    | Rule Num.       | MSPE (train)   | MSPE (test)    |
| linmod [118]                    | 0.0046           | 0.0150         | –               | 0.0042         | 0.0163         |
| pffr [160]                      | 0.0133           | 0.0134         | –               | 0.0134         | 0.0131         |
| affpc [176]                     | 0.0106           | 0.0111         | –               | 0.0104         | 0.0114         |
| affs [176]                      | 0.0091           | 0.0117         | –               | 0.0094         | 0.0115         |
| flm [176]                       | 0.0141           | 0.0130         | –               | 0.0130         | 0.0135         |
| FFS (order=2)                   | 0.0086           | 0.0119         | 15.16           | 0.0085         | 0.0120         | 14.1           |
| FFS (order=3)                   | 0.0083           | 0.0112         | 15.9            | 0.0081         | 0.0115         | 15.9           |
| FFS (order=4)                   | 0.0088           | 0.0112         | 14.54           | 0.0080         | 0.0119         | 14.06          |
| EFFS(order=2)                   | 0.0100           | 0.0111         | 17.25           | 0.0097         | 0.0112         | 37.03          |
| EFFS(order=3)                   | 0.0096           | 0.0110         | 25.21           | 0.0097         | 0.0112         | 37.69          |
| EFFS(order=4)                   | 0.0094           | 0.0111         | 36.43           | 0.0096         | 0.0112         | 37.87          |

The dataset used in this example is the electricity usage data described in term 3 in Section B.2, Appendix B. The data are randomly reordered for 50 times, and every time the first 100 pairs are chosen as the training data, and the remaining 261 pairs are set as the testing data. The comparisons with other state-of-the-art models and algorithms on this Monte Carlo simulation example are listed in Table 6.1. EFFS is running on the online mode, and the testing results are computed only based on the testing data. The online training and prediction MSPEs for real observations are presented in Table 6.2. Note that “SigCompFofHighDim” and “sigmod” which appeared in example 2 in
Table 6.2: Example 1: Online learning MSPEs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSPE (train)</td>
<td>MSPE (test)</td>
</tr>
<tr>
<td>EFFS(order=2)</td>
<td>0.0112</td>
<td>0.0132</td>
</tr>
<tr>
<td>EFFS(order=3)</td>
<td>0.0105</td>
<td>0.0132</td>
</tr>
<tr>
<td>EFFS(order=4)</td>
<td>0.0098</td>
<td>0.0127</td>
</tr>
</tbody>
</table>

Section 6.5.2 lack the ability to deal with missing values; therefore they are not listed in Table 6.1. The results of EFFS presented in Table 6.1, 6.2 are the best average results that can be obtained ($\varepsilon_a = 0.6, \varepsilon_m = 0.98$).

The comparisons in Table 6.1 reflect that EFFS can give satisfactory results no matter which order of the spline basis is used. Table 6.2 demonstrates that EFFS can also obtain very accurate online prediction results. It can be observed from Table 6.1 and 6.2, that the results of EFFS are stable no matter what number of bases and which spline order is applied.

6.5.2 Example 2: Diffusion Tensor Imaging (DTI) Data

The DTI dataset used in this example is specified in term 1 of Section B.2, Appendix B. The first 50 data pairs are used for training, and the remaining 205 pairs are applied for testing.

Table 6.3, 6.4 reflect, that no matter which order or how many splines are used, EFFS would always be a better choice compared with other approaches. Note that “X” indicates that the method “sigmoid” failed to obtain the results.

The online leaning characteristics of EFFS makes that the learned functional fuzzy rules are updated over time. The rules, 17 in total, obtained by EFFS at the last learning step are shown in Fig. 6.2. These are the results obtained when spline order equals 2, the number of basis is 10, and $\varepsilon^{(a)} = 0.6$ and $\varepsilon^{(m)} = 0.98$. Furthermore, the radii of these 17 rules are: 0.0142, 0.0177, 0.0136, 0.0645, 0.0224, 0.0118, 0.0236, 0.0363, 0.0321, 0.0180, 0.0184, 0.0491, 0.0491, 0.0184, 0.0184, 0.0378, 0.0184.
TABLE 6.3: Example 2: Result comparison of different methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>No. of Basis = 10</th>
<th>No. of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSPE (train)</td>
<td>MSPE (test)</td>
</tr>
<tr>
<td>linmod [118]</td>
<td>0.0035 0.0060</td>
<td>–</td>
</tr>
<tr>
<td>pffr [160]</td>
<td>0.0032 0.0052</td>
<td>–</td>
</tr>
<tr>
<td>affpc [176]</td>
<td>0.0036 0.0048</td>
<td>–</td>
</tr>
<tr>
<td>affs [176]</td>
<td>0.0028 0.0058</td>
<td>–</td>
</tr>
<tr>
<td>flm [176]</td>
<td>0.0035 0.0050</td>
<td>–</td>
</tr>
<tr>
<td>SigCompFofHighDim [170]</td>
<td>0.0038 0.0054</td>
<td>–</td>
</tr>
<tr>
<td>sigmod [169]</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FFS (order=2)</td>
<td>0.0034 0.0052</td>
<td>5</td>
</tr>
<tr>
<td>FFS (order=3)</td>
<td>0.0036 0.0053</td>
<td>5</td>
</tr>
<tr>
<td>FFS (order=4)</td>
<td>0.0038 0.0058</td>
<td>6</td>
</tr>
<tr>
<td>EFFS(order=2)</td>
<td>0.0038 0.0045</td>
<td>11.35</td>
</tr>
<tr>
<td>EFFS(order=3)</td>
<td>0.0040 0.0047</td>
<td>17.15</td>
</tr>
<tr>
<td>EFFS(order=4)</td>
<td>0.0042 0.0050</td>
<td>24.20</td>
</tr>
</tbody>
</table>

TABLE 6.4: Example 2: Online learning MSPEs.

<table>
<thead>
<tr>
<th>Method</th>
<th>No. of Basis = 10</th>
<th>No. of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSPE (train)</td>
<td>MSPE (test)</td>
</tr>
<tr>
<td>EFFS(order=2)</td>
<td>0.0036 0.0043</td>
<td>11.35</td>
</tr>
<tr>
<td>EFFS(order=3)</td>
<td>0.0037 0.0046</td>
<td>17.15</td>
</tr>
<tr>
<td>EFFS(order=4)</td>
<td>0.0039 0.0049</td>
<td>24.20</td>
</tr>
</tbody>
</table>

Fig. 6.2: Centres and consequents of rules.
6.5.3 Example 3: ETFs Intra-Daily Volumes

This example focuses on making online predictions for the daily curves of the equity transaction volumes with the daily volume curves used as inputs. The intra-daily predictions for transaction volumes are very crucial ingredients for many of the algorithmic trading strategies (e.g., Volume Weighted Average Price (VWAP) [216]). ETFs are chosen to be studied in this example. As most ETFs track a stock or bond index, data from two equity index ETFs, DIA (Dow Jones ETF) and QQQ (Nasdaq ETF), from Jan. 2014 to Dec. 2015 are used. The datasets contain the trading data in milliseconds collected from the NYSE Trade and Quote (TAQ) in WRDS. By the method shown in [217], these raw observations are cleaned, with wrong observations being detected and removed. The daily volume curve used in this example is the 15-min time series, each point of which is calculated by the sum of all the transaction volumes in the corresponding 15-min time interval. Therefore, there should be 26 observations on each daily curve. Note that the opening time for the market is 9:30 am, while the closing time is 16:00 pm. Supposing the volume curve for the $N$-th day is $x_N(s)$, $x_N(s)$ is applied to predict $x_{N+1}(s)$. The spline orders are chosen from 2, 3, 4. Both 10 and 20 are used as the number of bases. After using logarithm on the data, the DIA observations are varying in $[9.2744, 14.9090]$, the mean is 11.8975, and only 0.4711% data are smaller than 10. Whereas the QQQ observations are within $[2.1972, 16.4678]$, the mean is 13.6061, and only 10 points (0.077%) are smaller than 10 including 1 point ($7.69 \times 10^{-3}$% smaller than 5). The prediction MSPEs of EFFS presented in Table 6.5, 6.6, 6.7 and 6.8 indicate that using the basis number as 10 can help to get more accurate results than choosing basis number as 20, for both DIA and QQQ volumes no matter which threshold values are used. The best result of DIA volume prediction results is MSPE=0.3730 (3.1351% of the mean of the data set) with 5.43 rules (on average) are used; the worst MSPE obtained is 0.4451 (3.7411% of the mean value) along with 2.78 rules are used on average. It can be seen from Table 6.7, that the best result for QQQ data is MSPE=0.3124 (2.2960% of the mean of the) with 43.74 rules being used on average, while the worst result can be found from Table 6.8 with MSPE=0.3486 (6.34 rules are used on average) is about 2.5621%.
TABLE 6.5: Example 3: Online learning MSPEs (Rule Num.) for DIA with Basis Num. = 10.

<table>
<thead>
<tr>
<th>$\varepsilon^{(m)}$</th>
<th>spline order = 2</th>
<th>spline order = 3</th>
<th>spline order = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon^{(a)}$</td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
</tr>
<tr>
<td>0.1</td>
<td>0.3965(5.00)</td>
<td>0.3730(5.43)</td>
<td>0.3753(6.34)</td>
</tr>
<tr>
<td>0.2</td>
<td>0.3811(8.31)</td>
<td>0.3771(8.80)</td>
<td>0.3771(8.79)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3841(26.71)</td>
<td>0.3830(28.04)</td>
<td>0.3797(28.67)</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4090(45.62)</td>
<td>0.3834(43.99)</td>
<td>0.3834(44.94)</td>
</tr>
<tr>
<td>0.5</td>
<td>0.3886(46.88)</td>
<td>0.3771(46.98)</td>
<td>0.3891(47.14)</td>
</tr>
<tr>
<td>0.6</td>
<td>0.3954(47.26)</td>
<td>0.3891(47.15)</td>
<td>0.3891(47.14)</td>
</tr>
</tbody>
</table>

TABLE 6.6: Example 3: Online learning MSPEs (Rule Num.) for DIA with Basis Num. = 20.

<table>
<thead>
<tr>
<th>$\varepsilon^{(m)}$</th>
<th>spline order = 2</th>
<th>spline order = 3</th>
<th>spline order = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon^{(a)}$</td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
</tr>
<tr>
<td>0.1</td>
<td>0.4377(2.78)</td>
<td>0.4395(2.78)</td>
<td>0.4451(2.78)</td>
</tr>
<tr>
<td>0.2</td>
<td>0.4039(10.57)</td>
<td>0.4387(10.06)</td>
<td>0.4036(10.71)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3986(19.96)</td>
<td>0.4016(20.55)</td>
<td>0.3966(17.92)</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4170(46.95)</td>
<td>0.4130(46.52)</td>
<td>0.4099(47.13)</td>
</tr>
<tr>
<td>0.5</td>
<td>0.3969(47.35)</td>
<td>0.4072(47.33)</td>
<td>0.4128(47.35)</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4235(47.42)</td>
<td>0.4001(47.35)</td>
<td>0.4263(47.42)</td>
</tr>
</tbody>
</table>

6.6 Chapter Summary

In this chapter, EFFS, an online learning approach for solving the function-on-function regression problems, has been proposed. Within the fuzzy systems framework, EFFS can effectively learn the nonlinear relationship between functions (i.e., learn or identify the nonlinear operator $f(\cdot)$ in $y(t) = f(x(s)) + \varepsilon(t)$) in a dynamic and interpretable way. It has been widely acknowledged that online/real-time learning is an efficient way to cope with large datasets. Big functional datasets and functional streaming data can be learnt effectively online, due to its adaptive feature and low requirement for historical information storage. Furthermore, an online learning approach of EFFS has been proposed. It provides the exact methods that enable the structure and parameters evolving of EFFS. The comparison of EFFS with the state-of-the-art function-on-function regression models have been made on benchmark examples. These have demonstrated EFFS can achieve accurate prediction results. Online learning examples shown in this chapter have
proven the effectiveness of EFFS on solving function-on-function regression problems online.

### TABLE 6.7: Example 3: Online learning MSPEs (Rule Num.) for QQQ with Basis Num. = 10.

<table>
<thead>
<tr>
<th>$\varepsilon^{(m)}$</th>
<th>spline order = 2</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>0.1</td>
<td>0.3433(4.60)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.3393(6.45)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.3266(19.29)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.3254(34.91)</td>
<td>0.3255(34.95)</td>
<td>0.3256(34.96)</td>
<td>0.3255(34.99)</td>
</tr>
<tr>
<td>0.5</td>
<td>0.3285(45.28)</td>
<td></td>
<td>0.3275(45.28)</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.3324(47.15)</td>
<td></td>
<td>0.3320(47.15)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>spline order = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.2</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.4</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>spline order = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.2</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.4</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.6</td>
</tr>
</tbody>
</table>

### TABLE 6.8: Example 3: Online learning MSPEs (Rule Num.) for QQQ with Basis Num. = 20.

<table>
<thead>
<tr>
<th>$\varepsilon^{(m)}$</th>
<th>spline order = 2</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
<td>0.98</td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
<td>0.98</td>
<td>0.95</td>
</tr>
<tr>
<td>0.1</td>
<td>0.3637(4.88)</td>
<td></td>
<td></td>
<td></td>
<td>0.3399(3.62)</td>
<td></td>
<td></td>
<td></td>
<td>0.3427(3.62)</td>
</tr>
<tr>
<td>0.2</td>
<td>0.3474(6.47)</td>
<td></td>
<td></td>
<td></td>
<td>0.3459(6.34)</td>
<td></td>
<td></td>
<td></td>
<td>0.3486(6.34)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3303(18.21)</td>
<td></td>
<td></td>
<td></td>
<td>0.3276(16.77)</td>
<td></td>
<td></td>
<td></td>
<td>0.3463(13.37)</td>
</tr>
<tr>
<td>0.4</td>
<td>0.3191(42.41)</td>
<td></td>
<td></td>
<td></td>
<td>0.3208(44.37)</td>
<td></td>
<td></td>
<td></td>
<td>0.3306(36.61)</td>
</tr>
<tr>
<td>0.5</td>
<td>0.3471(47.41)</td>
<td></td>
<td></td>
<td></td>
<td>0.3310(47.38)</td>
<td></td>
<td></td>
<td></td>
<td>0.3311(47.28)</td>
</tr>
<tr>
<td>0.6</td>
<td>0.3327(47.54)</td>
<td></td>
<td></td>
<td></td>
<td>0.3358(47.52)</td>
<td></td>
<td></td>
<td></td>
<td>0.3349(47.47)</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusions and Future Work

This thesis concentrates on the fuzzy system approaches to regression problems for the data either in real space $\mathbb{R}^n$ or functional spaces. Thus, this thesis can be divided into two parts. The first part, which contains the research works in Chapters 3 and 4, is devoted to studying and answering some unsolved questions for the data streams online learning and predicting problems based on the EFS framework. These questions are:

- How to ensure the optimality of EFSs?
- How to identify the thresholds that are used to control the system evolving?

The second part, which includes the research in Chapters 5 and 6, concerns the functional data regression problems of those regression models whose inputs and outputs are functions. Two questions need to be answered in this part are:

- Is it possible to propose a function-on-function regression approach which has an interpretable and simple structure, and can deal with the nonlinear or linear regression problems? If yes, how?
- How to deal with the functional regression problem in real-time, when dataset is big?

In Section 7.1 of this chapter, the major contributions made by this thesis are summarized. The possible future works related to this thesis are discussed in Section 7.2.
7.1 Thesis Contributions

Overall, the research work of this thesis is settled on the regression problems of data streams and functional (streaming) data. These are very important research topics in big data research, and also very crucial topics in the areas of application, e.g., finance, medicine and energy.

The following subsections summarize the work that have been carried out and presented in Chapters 3, 4, 5 and 6.

7.1.1 Evolving Fuzzy Systems: Optimum Approaches

The traditional approaches for identifying the EFSs suffer the problem of being suboptimal. Hence this thesis starts from here in order to explore and propose EFSs identification approaches that can ensure optimality. Most of the traditional EFSs learning approaches are proposed in a heuristic way, i.e., thinking about how to evolve the rule base, and then implementing the existing consequent learning approaches, directly. For example, this process is just like making a model. Firstly, one needs to design and make each component individually and then splice them together as the final equipment. In this way, it is hard to ensure that each part can cooperate well. Unlike this process, the identification approaches are proposed thinking from the final optimization objective, and then propose the approaches for structure and parameters evolving backwards for severing the ultimate aim. To give a more visible picture, firstly, it should be considered what the final equipment should look like; Thereafter, it should be investigated how each component should be designed to make the equipment satisfy certain characteristics.

Returning to the final optimization problem of EFSs with regard to the data streams, the error functions should be minimized finally. One most widely used approach to solve this problem is the WRLS, however, this approach can not obtain an local optimal estimator, but a local suboptimal one. The starting point is the original form of WRLS. Through considering every change of the structure or the antecedent parameters of the fuzzy system, the corresponding formulas extended form of WRLS (known as EWRLS for short), which can guarantee the local optimality, are induced. The methods proposed in Chapter 3 known as EMFSPO and LEOA depend on two different fuzzy rules, Mamdani and T-S, respectively. Comparison results with many state-of-the-art approaches on the benchmark examples have demonstrated the proposed approaches can provide
favourable results. Furthermore, an additional discovery is that the LEOA has a global optimal behaviour.

However, there is never a “free lunch”. In order to fulfill the aim of optimality, EMFSPO and LEOA need to remember additional information and even keep more fuzzy rules. This indicates that the EMFSPO and LEOA need to sacrifice the computational speed in order to pay for the optimality. In general, EMFSPO and LEOA provide a choice for people to select when they are focusing more on the optimality than the processing speed.

7.1.2 Evolving Fuzzy Systems: Self-Learning Approaches for Dynamic Thresholds

Because fast approaches are more applicable for handling the data streams, suboptimal approaches are concentrated on. Another important problem, which has not been properly solved by most of the existing EFSs learning approaches, is studied: the thresholds selection problem. Heuristic values of the thresholds chosen from the extensive experiments are frequently used in the existing works. Firstly, SEFS has been proposed. SEFS can be regarded as the first trail for proposing an EFSs learning approach that provides a dynamic speed of rule generation. SEFS realizes the dynamic threshold relying on the risk of overfitting and underfitting contained in the online training errors, and then enabling the threshold which fluctuates in an appropriate interval. However, SEFS still needs to select the predefined threshold that controls rule simplification. In order to solve this problem, an improved approach known as EFS-SLAT has been proposed. EFS-SLAT has the dynamic thresholds for controlling the rule generation and simplification, allowing the two thresholds to cooperate with, but to restrict, each other.

Furthermore, a geometric similarity measure has been proposed and embedded in SEFS. This measure helps to judge the similarity between two fuzzy rules by computing the $L^2$ distance between firing strengths. Instead of computing the numerical integral directly, the formula has been inducted to compute the similarity swiftly and accurately. However, this similarity measure may misidentify the similarity between two tiny clusters, due to the fact that the small value of the centres and radius would disturb the judgement of the similarity. In order to make up this shortcoming, Mahalanobis distance has been used in the EFS-SLAT to assist and enhance the proposed geometric similarity measure. Moreover, a local version of the VFF-RLS, which is known to be robust to the
noise, has been carried out for updating consequent parameters.

### 7.1.3 Functional Fuzzy System — An Offline Function-on-Function Regression Approach

As many of the data are actually samples of functions, in Chapters 3 and 4, the regression problem is about learning the function/map through its samples. However, the problem which is investigated in Chapters 5 and 6 is about learning the map between the curves in two functional spaces. With each point to be a curve, the functional data analysis enables to deal with more information and high dimensional data. There are not many research works which are able to solve the nonlinear regression problems using an understandable model. Unlike many statistical models, fuzzy systems requires no assumptions for data distributions or linear (or certain types of nonlinear) relationships. The fuzzy systems are known as universal approximators, and equipped with flexible and interpretable structures with the help of “If-Then” fuzzy reasoning. Inspired by these properties that fuzzy systems have, an FFS, which can carry out the function-on-function nonlinear regression problems, has been proposed. From the fuzzy systems aspect, by extending the input and output space from finite dimensional spaces to infinite dimensional spaces, FFS can be regarded as a new type of fuzzy systems.

To identify the FFS, the k-mean clustering method provides a great help in clustering and constructing the fuzzy rules. Furthermore, the antecedent parameters can also be estimated from the grouped data. The consequent parameters of the FFS are identified though the induced least square approach (global optimal) from minimizing the error function.

The FFS has been compared with many recent and classic function-on-function regression approaches to either artificial or real datasets. The results indicated that FFS is a competitive approach.

### 7.1.4 Evolving Functional Fuzzy System — An Online Function-on-Function Regression Approach

The existing functional data analysis tools can assist in handling the high dimensional data to some degree. However, when a big functional dataset (or a functional streaming
dataset) need to be processed, the existing approaches, most of which are offline methods, would take a long time for training. Utilizing the knowledge of the data streams prediction approaches, a possible way is treating the large functional datasets based on the streaming processing, which makes the data arrive in streams. Then, these data can be processed using online learning approaches. In order to achieve the online learning goal of functional data, an EFFS has been proposed. The EFFS is a one-pass approach, which discards the data point once it has been used, and which stores no historical data. EFFS borrows the idea from EFSs, but enables it to deal with large functional datasets.

In order to identify an EFFS online, the rule number and antecedent parameters are identified dynamically under the framework of EFSs. In the meantime, the WRLS formulas for updating the consequent parameters are induced from minimizing a group of local error functions. Not only benchmark examples are used for comparison with the existing approaches, but an online learning example is also provided. The numerical results can support the effectiveness of the EFFS. However, it can also be found that EFFS involves a relatively large number of rules. This phenomenon may be caused by the fact that merging similar rules can contribute little to rule simplification. Further work needs to be carried out in order to deal with this problem.

7.2 Future Work

Even though some progress on both data streams and functional data regression approaches have been made, these achievements are only the tip of the iceberg. There still remain a lot of problems which need to be investigated in the future. The future work plans are presented using the following bullet points:

- LEOA and EMFSPO are two EFS learning approaches that are proposed in order to obtain local optimal estimations. However, these two methods need to store additional information, which can be observed from the EWRLS formulas used to estimate the consequent parameters. Furthermore, this additional information may increase significantly when the rule numbers are increased. This phenomenon results in a slower computational speed of the LEOA and EMFSPO, especially when the rule numbers become large. Moreover, LEOA and EMFSPO also try to keep more fuzzy rules than some of the other state-of-the-art approaches. This indicate that the LEOA and the EMFSPO may be at risk of overfitting when they
realize the aim of getting optimal estimation. As a result, further works should be done in order to deal with this problem and find the best tradeoff between the optimal and the computation speed.

- SEFS and EFS-SLAT are two approaches proposed for allowing the thresholds (or control parameters) learned from the data automatically by the algorithms themselves. These two approaches enable the thresholds to vary in certain intervals applying the information about the risk of overfitting and underfitting contained in the training errors. However, this information can only reflect the risk of the system that suffers the overfitting and underfitting problem, and cannot accurately demonstrate whether the underfitting or overfitting actually happens. In this sense, more accurately judgement approaches should be explored. Furthermore, it is uncertain that the threshold only relies on the underfitting or overfitting information, more indicators should be studied. Moreover there are two boundaries, lower and upper bounds, which need to be determined. For this reason, the ideal approach, which has the thresholds completely learned from the data without (or with fewer) human controls, should be a direction of the future research.

- Considering the four EFS learning approaches (i.e., LEOA, EMFSPO, SEFS, EFS-SLAT), two kinds of fuzzy rules, T-S and Mamdani, are used. More types of fuzzy system frameworks are worth investigating, for example, the system with non-axis-paralleled rules (or the generalized evolving fuzzy systems introduced in Section 2.2.2.2.1.C in Chapter 2) used in [55] (PANFIS) and [58] (Gen-Smart-EFS), etc. Besides, reviewing the proposed approaches and the existing approaches, the criteria for testing this series of data stream learning approaches are the traditional benchmark examples. Strictly speaking, these datasets are not data streams. More work should be done on the application topics. Additionally, the online feature selection problem is seldom discussed. As selecting the appropriate features may have a great influence for the prediction accuracies, the feature selection is therefore an important topic that should be studied in depth.

- In the functional data regression aspect, the proposed approaches, FFS and EFFS, are constructed on a single-input-single-output (SISO) rules. More frameworks for the FFS should be studied. For example, the rule base can be generalized to be multi-input-single(or multi)-output (MISO or MIMO). But one should be reminded that the high dimensional functional data are very likely to bring a huge
computational burden. It is obvious that either the existing approaches or the proposed approaches contain a crucial step, this being the data preprocessing. This procedure is the function approximation of the data, which enables the infinite dimensional data to be processed in the finite dimensional space. The problem of determining the number of bases has not been properly solved, and more research should be done. Especially, when the aim is proposing an online approach, it usually arises the question of whether the basis numbers can be learned online and how? Furthermore, because rule pruning approaches are effective tools to simplify the rule base online. Therefore, rule pruning approaches should be investigated in EFFS related research.

- Despite of the fuzzy systems based functional data regression approaches, more research should be done in investigating interpretable nonlinear regression approaches. Furthermore, in functional data research area, there are many other research topics that worth to be studied further. For examples, feature selection problem is another important topic in functional data regression, classification, clustering, etc.; Approaches for functional data classification and clustering should be investigated; and real applications for functional data analysis tools still has a long way to go.
Appendix A

Mathematical Inductions, Proofs and Additional Numerical Results

A.1 Mathematical Proof in Chapter 3

A.1.1 Proof of Theorem 3.1

Proof. $\gamma^t_{i}(x(t+1)) < \varepsilon_0$ implies $\exp \left\{ - \frac{1}{\sigma_{k(t+1), j}} \right\} \leq \frac{\varepsilon_0}{T}$. As each $\gamma^t_{i}(x(t+1))$ is bounded, so $\exists \varepsilon > 0$ such that $\sum_{k=1}^{t} A_i(x(k)) < \varepsilon$. $\square$

A.1.2 Proof of Theorem 3.2

Proof. As

$$
\frac{\gamma^t_{i}(x(k))}{\left( \sum_{l=1}^{K(t)} \gamma^t_{l}(x(k)) \right) \left( \sum_{l=1, l \neq i}^{K(t)} \gamma^t_{l}(x(k)) \right)}
$$

$i = 1, 2, \ldots, K(t)$, are bounded and $\mathcal{M}^t_{i} < \varepsilon_p$, then $\exists \varepsilon > 0$ such that $\sum_{k=1}^{t} B_{i,k} < \varepsilon$. $\square$
APPENDIX A. MATHEMATICAL INDUCTIONS, PROOFS AND ADDITIONAL NUMERICAL RESULTS

A.1.3 Proof of Theorem 3.3

**Proof.** Once \( \| c_{i_1} - c_{i_2} \| < \varepsilon_c \) and \( \| \sigma_{i_1}^2 - \sigma_{i_2}^2 \| < \varepsilon_{\sigma} \) hold, then \( \gamma_i^{(t)}(x(k)) \approx \gamma_{i_2}^{(t)}(x(k)) \approx \gamma_{i_0}^{(t)}(x(k)) \). Use \( \gamma_i^{(t)}(x(k)) \) to replace \( \gamma_{i_0}^{(t)}(x(k)) \) in (3.17) and (3.19), we can get (A.2),

\[
\overline{R}'(t + 1) = \overline{R}'(t) + \sum_{k=1}^t \tilde{C}_{i,k} + \theta_i^{(t+1)}(x(t + 1)),
\]

(A.2)

\[
f'(t + 1) = f'(t) + \sum_{k=1}^t \tilde{C}_{i,k} \gamma(k) + \theta_i^{(t+1)}(x(t + 1)) y(t + 1).
\]

in which

\[
\tilde{C}_{i,k} = \frac{\gamma_i^{(t)}(x(k))}{\left| \sum_{l=1}^{K(i)} \gamma_i^{(t)}(x(k)) - \gamma_{i_2}^{(t)}(x(k)) \right|}.
\]

Let \( \tilde{R}'_{i}(t + 1) = \sum_{k=1}^t \{ \theta_i^{(t)}(x(k)) + \tilde{C}_{i,k} \}, \tilde{f}_{i}(t + 1) = \sum_{k=1}^t \{ \theta_i^{(t)}(x(k)) + \tilde{C}_{i,k} \} y(k), \) thus (3.23) holds. Further, from \( \overline{R}'(t + 1) = \tilde{R}'_{i}(t) + \theta_i^{(t+1)}(x(k)) \) and \( \tilde{f}'(t + 1) = \tilde{f}_{i}(t) + \theta_i^{(t+1)}(x(k)) y(k) \), it could be obtained that (3.22) holds. \qed

A.1.4 Proof of Theorem 3.4

**Proof.** If Condition 3.5 holds, then \( \sigma_{K(i+1),j}^2 \leq \frac{n(\gamma_{K(i+1),j})^2}{-2\log(\frac{\eta}{\psi})} \) is satisfied, which further implies

\[
\exp\{- n \sum_{j=1}^{K(i+1),j} \left( \frac{(\gamma_{K(i+1),j})^2}{-2\log(\frac{\eta}{\psi})} \right) \} \leq \frac{\eta}{\psi}. \]

Because each \( \frac{\gamma_{K(i+1),j}}{\gamma_{K(i),j}(x(k))} \) is bounded, so \( \exists \varepsilon > 0 \) such that \( \sum_{k=1}^{t} A_{i,k} x(k)^T < \varepsilon \) and \( \sum_{k=1}^{t} A_{i,k} y(k) < \varepsilon \). Then, \( \psi_i \) could be estimated recursively by (3.59). Besides, it is easy to verify that \( \varepsilon \)-completeness is satisfied. \qed

A.1.5 Proof of Theorem 3.5

**Proof.** As there exists a finite number \( N_1 > 0 \) such that \( \frac{\gamma_i^{(t+1)}(x(k))}{\left( \sum_{l=1}^{K(i+1)} \gamma_i^{(t)}(x(k)) \right) \left( \sum_{l=1}^{K(i+1)} \gamma_i^{(t+1)}(x(k)) \right)} \) \( i = 1, 2, \ldots, K', \) \( i \neq i^* \), and \( \sum_{p=i^*+1}^k \gamma_i^{(p)}(x(p)) < \varepsilon_p \), so there exists an \( \varepsilon > 0 \) such that \( \sum_{k=1}^{t} B_{i,k} x(k)^T < \varepsilon \) and \( \sum_{k=1}^{t} B_{i,k} y(k) < \varepsilon \). \qed
A.1.6 Proof of Theorem 3.6

Proof. As condition 3.7 is satisfied, so \( r_i^{(l)}(x(k)) \approx r_i^{(l)}(x(k)) \approx r_i^{(l)}(x(k)) \), thus we can use \( \gamma_i^{(l)}(x(k)) \) to replace \( r_i^{(l)}(x(k)) \) in (3.64) and (3.65), then these two equations could be approximated by (A.4) and (A.5),

\[
\bar{R}(t + 1) = \bar{R}(t) + \sum_{k=1}^{l} C_{i,k}^{d_{j_2}} x(k)^{T} x(k) + \sum_{k=1}^{l} \gamma_i^{(l)}(x(t + 1)) x(t) x(t)^{T}, \quad \text{(A.4)}
\]

\[
f_i(t + 1) = f_i(t) + \sum_{k=1}^{l} C_{i,k}^{d_{j_2}} x(k)^{T} y(k) + \sum_{k=1}^{l} \gamma_i^{(l)}(x(t + 1)) x(t) y(t + 1), \quad \text{(A.5)}
\]

in which

\[
C_{i,k}^{d_{j_2}} = \frac{\gamma_i^{(l)}(x(k)) \gamma_i^{(l)}(x(k))}{\sum_{l=1}^{K^{(l)}} \gamma_i^{(l)}(x(k)) - \gamma_i^{(l)}(x(k))}. \quad \text{(A.6)}
\]

Let

\[
\tilde{R}_i^{(l_2)}(t + 1) = \bar{R}(t) + \sum_{k=1}^{l} C_{i,k}^{d_{j_2}} x(k)^{T} x(k) = \sum_{k=1}^{l} \{\theta_i^{(l)}(x(k)) + C_{i,k}^{d_{j_2}}\} x(k)^{T} x(k), \quad \text{(A.7)}
\]

\[
f_{i}^{(l_2)}(t + 1) = f_i(t) + \sum_{k=1}^{l} C_{i,k}^{d_{j_2}} x(k)^{T} y(k) = \sum_{k=1}^{l} \{\theta_i^{(l)}(x(k)) + C_{i,k}^{d_{j_2}}\} x(k)^{T} y(k), \quad \text{(A.8)}
\]

thus we can get that (3.73), (3.74) and (3.75) hold. Further, use (A.9) and (A.10),

\[
\bar{R}(t + 1) = \tilde{R}_i^{(l_2)}(t) + \sum_{k=1}^{l} \gamma_i^{(l)}(x(t + 1)) x(t) x(t)^{T}, \quad \text{(A.9)}
\]

\[
f_i(t + 1) = \tilde{f}_i^{(l_2)}(t) + \sum_{k=1}^{l} \gamma_i^{(l)}(x(t + 1)) x(t) y(t + 1). \quad \text{(A.10)}
\]

it is easy to get that (3.72) holds. \( \square \)

A.1.7 Deduction of Formula (3.68)

Once \( l_1 \)-th and \( l_2 \)-th fuzzy rule are merged into the \( l_0 \)-th as shown in (3.68), the sample size
APPENDIX A. MATHEMATICAL INDUCTIONS, PROOFS AND ADDITIONAL NUMERICAL RESULTS

205

of the $l_0$-th cluster becomes $N_l + N_{l_2}$. Based on (3.71), the radius $\sigma_{l_0,j}$ could be estimated using (A.11),

$$
(\sigma_{l_0,j})^2 = \frac{N_l + N_{l_2}}{N_l + N_{l_2} - 1} \sum_{k=1}^{N_l} (x_{k,j} - c_{l_0,j})^2
\quad \text{(A.11)}
$$

Similarly, we have

$$
\sum_{i=1}^{N_l} (x_{i,j} - c_{l_0,j})^2 = \sum_{i=1}^{N_l} (x_{i,j} - c_{l_1,j} + c_{l_1,j} - c_{l_0,j})^2
\quad \text{(A.12)}
$$

$$
= (N_l - 1) (\sigma_{l_1,j})^2 + N_l (c_{l_1,j} - c_{l_0,j})^2.
$$

Similar formula could be used to present $\sum_{i_2=1}^{N_l} (x_{i_2,j} - c_{l_0,j})^2$. Therefore, based on (A.11) and (A.12), (3.68) is easy to be obtained.

### A.2 Mathematical Proof in Chapter 4

#### A.2.1 The Proof of Proposition 4.1

**Proof.** Let $A = \sum_{k=1}^{t} \lambda^{t-k} e_k$; then, $\varepsilon_t = \varepsilon_{\max} - (\varepsilon_{\max} - \varepsilon_{\min}) \exp\{-A\}$. Assume $A' \leq A''$; then,

$$
\varepsilon'_t - \varepsilon''_t = \varepsilon_{\max} - (\varepsilon_{\max} - \varepsilon_{\min}) \exp\{-A'\} - \varepsilon_{\max} - (\varepsilon_{\max} - \varepsilon_{\min}) \exp\{-A''\}
\quad \text{(A.13)}
$$

$\therefore A' - A'' \leq 0$, thus, $\varepsilon'_t - \varepsilon''_t \leq 0$. Therefore, $\varepsilon_t$ is monotonically increasing against $\sum_{k=1}^{t} \lambda^{t-k} e_k$. $\square$

#### A.2.2 Induction of (4.8) and (4.9) in rule updating method

Suppose $x(t)$ is the new input, and the cluster center and radius for fuzzy rule $R_r$ is $c_{r}(t-1) = (c_{r,1}(t-1), c_{r,2}(t-1), \ldots, c_{r,n}(t-1))$ and $\sigma_{r}(t-1) = (\sigma_{r,1}(t-1), \sigma_{r,2}(t-1), \ldots, \sigma_{r,n}(t-1))$, respectively. Assume that the input points that satisfy the *rule updating method* before $x(t)$
comes are \( \{x'(k)\}_{k=1}^{N_e(t-1)} \). Input \( x(t) \) satisfies the condition of the rule updating method; then, the updated center (\( c_p(t) \)) and radius (\( \sigma_p(t) \)) of rule \( R_p \) are inducted by (A.14) and (A.15), respectively:

\[
c_p(t) = \left( \sum_{k=1}^{N_e(t-1)} x'(k) + x(t) \right) / (N_e(t-1) + 1)
\]

\[
= (N_e(t-1) \cdot c_p(t-1) + x(t)) / (N_e(t-1) + 1)
\]

\[
c_p(t) = c_p(t) + (x(t) - c_p(t-1)) / (N_e(t-1) + 1)
\]

(A.14)

\[
(\sigma_p(t))^2 = \sum_{k=1}^{N_e(t-1)} (x'_j(k) - c_p(t))^2 + (x(t) - c_p(t))^2
\]

\[
= \frac{N_e(t-1)}{(N_e(t-1) + 1)} + \frac{x(t) - c_p(t)}{(N_e(t-1) + 1)}
\]

(A.15)

where \( j = 1, 2, \ldots, n \).

### A.2.3 Induction of (4.20) and (4.21) in rule merging method

Assume that all the historical inputs within the cluster of rule \( R_k \) are \( \{x_{i_k}(l_{k})\}_{l_{k}=1}^{N_{i_k}} \). Then, \( c_p \) and \( \sigma_p \) could be computed by (A.16) and (A.17), respectively, as follows:

\[
c_p = \sum_{l_{k}=1}^{N_e} x_l(I_{p})(l_{k}) + \sum_{l_{k}=1}^{N_e} x_l(I_{k})(l_{k})
\]

\[
= \frac{N_{i_k} c_p + N_{i_k} c_{i_k}}{N_e + N_{i_k}}
\]

(A.16)

\[
(\sigma_p)^2 = \sum_{l_{k}=1}^{N_e} (x_l(I_{p}) - c_p)^2 + \sum_{l_{k}=1}^{N_e} (x_l(I_{k}) - c_{i_k})^2
\]

\[
= \frac{N_{i_k} c_p + N_{i_k} c_{i_k}}{N_e + N_{i_k}}
\]

(A.17)

The induction of \( \psi_p \) is almost the same as \( c_p \).
A.2.4 The Proof of Theorem 4.1

Proof. Without loose of generality, we prove the theorem from the following three cases.

1. Rule adding: Assume $\Omega' \subseteq \Omega$ is a closed subset and $\Omega' \subseteq \bigcup_{i=1}^{n_0} B_i(c_i, k_i \sigma_i)$, where $B_i(c_i, k_i \sigma_i)$ are the open balls with centers $c_i$ and radii $k_i \sigma_i$.

   (i) Assume there exist countable number of points $\{x(k)\}_{k=1}^{\infty} = \Omega \setminus \Omega'$. Based on the rule adding method, $\forall x(k)$, there would be a new fuzzy rule added with center $x(k)$ and radius $\frac{1}{k_i} |x(k) - c_i|$, where $k_i = \sqrt{-2 \log \varepsilon_i}$, $\varepsilon_i = \arg \max_{i=1, \ldots, n_0} \|x(k) - c_i\|$. Then, $x(k) \in B'_i(x(k), k_i \sigma_i)$, where $B'_i(x(k), k_i \sigma_i)$ is an open ball with center $x(k)$. Besides, as $\varepsilon_i \geq \varepsilon_{\text{min}}$, $\therefore \exp\left\{-\sum_{j=1}^{n} \frac{(x_j(x(k)) - c_{i,j})^2}{2 \sigma^2_{i,j}} \right\} \geq \varepsilon_{\text{min}}$. Therefore, $\bigcup_{i=1}^{n_0} B_i(c_i, k_i \sigma_i) \cup \bigcup_{k=1}^{\infty} B'_i(x(k), k_i \sigma_i)$ is an open cover of $\Omega$. $\therefore$ is compact, $\exists \bigcup_{k=1}^{\infty} B'_i(x(l), k_i \sigma_i) \subseteq \Omega \subseteq \bigcup_{i=1}^{n_0} B_i(c_i, k_i \sigma_i) \cup \{\bigcup_{i=1}^{n_0} B'_i(x(l), k_i \sigma_i)\}$. Let $N_0 = n_0 + N'_0$, the result is proved.

(ii) Assume there exists a closed subset $\Omega'' = \Omega \setminus \Omega'$. Construct a bounded countable subset $W = \mathbb{Q}^n \cap \Omega''$. It is obvious that $W$ is dense in $\Omega''$. Based on the rule adding method, $\forall x'(k) \in W$, $\exists B'_i(x'(k), k_i \sigma_i)$ with $c_i$ and $\sigma_i$ are the same as they are in (i).

As $W$ is countable, so $\{B'_i(x'(k), k_i \sigma_i)\}_{k=1}^{\infty}$ is a countable open cover of $W$. Because $W$ is dense in $\Omega''$, $\forall x(k) \in W$ s.t. $x(k) \in B'_i(x'(k), k_i \sigma_i)$. Therefore, $\Omega'' \subseteq \bigcup_{i=1}^{n_0} B'_i(x'(l), k_i \sigma_i)$. Furthermore, similar to (i), $\therefore \Omega''$ is compact, $\exists N'_0 < \infty$ s.t. $\Omega \subseteq \bigcup_{i=1}^{n_0} B_i(c_i, k_i \sigma_i) \cup \{\bigcup_{i=1}^{n_0} B'_i(x'(l), k_i \sigma_i)\}$.

2. Rule updating:

As $\lim_{N_i \to \infty} c_{i,r}(t) = c_{i,r}$, $\lim_{N_i \to \infty} \sigma_{i,r}^2(t) = \sigma_{i,r}^2$, and from (4.8) and (4.9), then $\lim_{N_i \to \infty} c_{i,r}(t + 1) = c_{i,r}$, $\lim_{N_i \to \infty} \sigma_{i,r}^2(t + 1) = \sigma_{i,r}^2$. Based on the convergence of the cluster centers and radiiuses, the conclusion could be get directly.

3. Rule merging:

As $\exists i_1, i_2 \in \{1, 2, \ldots, K\}$ s.t. $\forall \varepsilon > 0$, $\| \gamma_{i_1}(x) - \gamma_{i_2}(x) \|_2 < \varepsilon$, then, $\forall x \in \Omega$, $| \sum_{j=1}^{n} \mu_{i,j} \mu_{i,j}(x) - \sum_{j=1}^{n} \mu_{i,j}(x) | = | \exp\left\{-\sum_{j=1}^{n} \frac{(x_j(x) - c_{i,j})^2}{2 \sigma_{i,j}^2} \right\} - \exp\left\{-\sum_{j=1}^{n} \frac{(x_j(x) - c_{i,j})^2}{2 \sigma_{i,j}^2} \right\} | < \varepsilon$ holds. So $| \sum_{j=1}^{n} \{(x_j - c_{i_1,j})^2 \sigma_{i_1,j}^2 - (x_j - c_{i_2,j})^2 \sigma_{i_2,j}^2 \} | < \varepsilon$; besides, $x_j = c_{i_1,j} \implies |c_{i_1,j} - c_{i_2,j}| < \varepsilon$, $x_j = c_{i_1,j} + \sigma_{i_1,j} \implies |\sigma_{i_1,j} - \sigma_{i_2,j}| < \varepsilon$. Furthermore, $\lim_{N_i \to \infty} c_{i_1}(t) = c_{i_1}$, $\lim_{N_i \to \infty} c_{i_2}(t) = c_{i_2}$, $\lim_{N_i \to \infty} \sigma_{i_1}^2(t) = \sigma_{i_1}^2$, $\lim_{N_i \to \infty} \sigma_{i_2}^2(t) = \sigma_{i_2}^2$, (21), (22). $\therefore \lim_{N_1, N_2 \to \infty} c_{i_0}(t + 1) = c_{i_0}$, $\lim_{N_1, N_2 \to \infty} \sigma_{i_0}^2(t + 1) = \sigma_{i_0}^2$, (21), (22). Then, the conclusion is obtained.

$\square$
A.2.5 Induction of VFF-WRLS

For each rule $R_i$, suppose

$$R_i(t) = \sum_{k=1}^{t} \lambda_i^{t-k} x(k) x^T(k) \quad \text{and} \quad (A.18)$$

$$Q_i(t) = \sum_{k=1}^{t} \lambda_i^{t-k} \theta_i(k) y(k). \quad (A.19)$$

Then, similar to [203], the conventional RLS (CRLS) updating formulas for $\psi_i$ could be presented as follows:

$$P_i(t) = \frac{1}{\lambda_i} (P_i(t-1) - \frac{\theta_i(t) P_i(t-1) x(t) x^T(t)}{\lambda_i + \tau_i(t)}), \quad (A.20)$$

$$\psi_i(t) = \psi_i(t-1) + \frac{q \theta_i(t) P_i(t-1) x(t) x^T(t) P_i(t-1)}{\lambda_i + \tau_i(t)}, \quad (A.21)$$

where $\tau_i(t) = \theta_i(t) x^T(t) P_i(t-1) x(t)$, and $q$ is the convergence factor. So, the posterior error signal $e_i(t)$ can be expressed as

$$e_i(t) = \theta_i(t) [y(t) - x^T(t) \psi_i(t)]$$

$$= (1 - q \tau_i(t)) e_i(t) - q \tau_i(t) \psi_i(t), \quad (A.22)$$

where $\tau_i(t) = \frac{\tau_i(t)}{1 + \tau_i(t)}$, $e_i(t) = e_i'(t) + \psi_i(t) = \theta_i(t) [x^T(t) h(t-1) - x^T(t) \psi_i(t-1)]$. The optimal $\lambda_i$ are obtained through the same way as [203] by minimizing $E[(e_i'(t))^2]$ and setting $q = 1$.

A.3 Mathematical Proof in Chapter 6

A.3.1 Induction and Numerical Computation of $L^2$ Norms in (6.6)

In (6.6), $\|c_{k,j} - \tilde{c}_{k,j}\|_2^2$ and $\|x_{N,j} - \tilde{c}_{k,j}\|_2^2$ should be calculated numerically by trapezoidal rule. Firstly, in order to compute $\|c_{k,j} - \tilde{c}_{k,j}\|_2^2$, we assume that the domain of $s_j$ is $\Omega_{s_j}$ which has an evenly partition with knots $s_{j,1}, s_{j,2}, \ldots, s_{j,L_j}$, then the $\|c_{k,j} - \tilde{c}_{k,j}\|_2^2$ can be computed by (A.23),

$$\|c_{k,j} - \tilde{c}_{k,j}\|_2^2 = \int_{\Omega_{s_j}} [c_{k,j}(s_j) - \tilde{c}_{k,j}(s_j)]^2 ds_j \quad (A.23)$$

$$= \sum_{l=1}^{L_j-1} \frac{1}{2} (s_{j,l+1} - s_{j,l}) \{[c_{k,j}(s_{j,l+1}) - \tilde{c}_{k,j}(s_{j,l+1})]^2 + [c_{k,j}(s_{j,l}) - \tilde{c}_{k,j}(s_{j,l})]^2\}.$$
APPENDIX A. MATHEMATICAL INDUCTIONS, PROOFS AND ADDITIONAL NUMERICAL RESULTS

209

Whereas, as we know the exact values of \( x_{N,j}(s_j) \) on its own trajectory, then we use \( s_{j,1}, s_{j,2}, \ldots, s_{j,L_{N,j}} \), which ensures that \( x_{j,1}, x_{j,2}, \ldots, x_{j,L_{N,j}} \) are known, as a partition of the \( \Omega_{N,j} \). Remark that \( s_{j,1}, s_{j,2}, \ldots, s_{j,L_{N,j}} \) need not to be evenly spaced. The \( L^2 \) norm \( \|x_{N,j} - \bar{c}_{k,j}\|_{L^2}^2 \) can be calculated by \( (A.24) \),

\[
\|x_{N,j} - \bar{c}_{k,j}\|_{L^2}^2 = \int_{\Omega_{N,j}} [x_{N,j}(s_j) - \bar{c}_{k,j}(s_j)]^2 ds_j
\]

\[
= \sum_{i=1}^{L_{N,j} - 1} \frac{1}{2} (s_{j,i+1} - s_{j,i}) \{ [x_{N,j}(s_{j,i+1}) - \bar{c}_{k,j}(s_{j,i+1})]^2 + [x_{N,j}(s_{j,i}) - \bar{c}_{k,j}(s_{j,i})]^2 \}
\]

\[\text{A.3.2 Induction of (6.13), (6.14) and (6.15)}\]

The recursive updating formulas (6.13), (6.14) and (6.15) are inducted from minimizing (6.12): \( \min \sum_{i=1}^{N} \theta_k(x_i) \int_{\Omega} [y_i(t) - (B^Y(t))^T h_k]^2 dt \). Compute the derivative about \( h_k \), we can get the equation \( (A.25) \):

\[
\frac{\partial L_k}{\partial h_k} = 2 \sum_{i=1}^{N} \theta_k(x_i) \int_{\Omega} B^Y(t) [y_i(t) - (B^Y(t))^T h_k] dt = 0.
\]

Therefore, \( (A.26) \) could be obtained:

\[
\hat{h}_k = \tilde{R}(N) f(N),
\]

where \( \tilde{R}(N) = \sum_{i=1}^{N} \theta_k(x_i) \int_{\Omega} B^Y(t) (B^Y(t))^T dt \) and \( f(N) = \sum_{i=1}^{N} \theta_k(x_i) \int_{\Omega} B^Y(t) y_i(t) dt \). We can get recursive formulas for \( \tilde{R}(N) \) and \( f(N) \) shown in \( (A.27) \) and \( (A.28) \):

\[
\tilde{R}(N) = \tilde{R}(N-1) + \theta_k(x_N) \int_{\Omega} B^Y(t) (B^Y(t))^T dt,
\]

\[
f(N) = f(N-1) + \theta_k(x_N) \int_{\Omega} B^Y(t) y_N(t) dt.
\]

Then, \( (6.15) \) could be obtained by compute \( \hat{h}_k^{(N)} = \tilde{R}(N)^{-1} f(N) \) using \( (A.29) \):

\[
\hat{h}_k^{(N)} = \tilde{R}(N)^{-1} \{ f(N-1) + \theta_k(x_N) \int_{\Omega} B^Y(t) y_N(t) dt \}
\]

\[
= \tilde{R}(N)^{-1} \{ R(N-1) \hat{h}_k^{(N-1)} + \theta_k(x_N) \int_{\Omega} B^Y(t) y_N(t) dt \}
\]

\[
= \tilde{R}(N)^{-1} \{ ([\tilde{R}(N) - \theta_k(x_N) \int_{\Omega} B^Y(t) (B^Y(t))^T dt] \hat{h}_k^{(N-1)} + \theta_k(x_N) \int_{\Omega} B^Y(t) y_N(t) dt \}
\]

\[
= \hat{h}_k^{(N-1)} + \tilde{R}(N)^{-1} \theta_k(x_N) \int_{\Omega} B^Y(t) e_N(t) dt.
\]
where $\varepsilon_N(t) = y_N(t) - (B^Y(t))^T \hat{h}_k^{(N-1)}$. We use formula $\{A + BCD\}^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}$ on (A.27) to further compute $R(N)^{-1}$. Let $A = R(N-1)$, $B = \int\Omega B^Y(t)(B^Y(t))^T dt$, $C = \theta_k(x_N)I, D = I$, then, we can get (A.30):

$$R(N)^{-1} = \hat{R}(N-1)^{-1} - \theta_k(x_N)\hat{R}(N-1)^{-1} \int\Omega B^Y(t)(B^Y(t))^T dt$$

$$\{\theta_k(x_N)\hat{R}(N-1)^{-1} \int\Omega B^Y(t)(B^Y(t))^T dt + I\}^{-1}\hat{R}(N-1)^{-1}. \tag{A.30}$$

Write $\hat{R}(N)^{-1}$ to be $P(N)$, (A.30) is turned out to be (6.13). Use (A.29) and (A.30) we can obtain (6.15).

**A.3.3 Calculate The Integrals in (6.13), (6.14) and (6.15)**

We need to induct the formulas for $\int\Omega B^Y(t)(B^Y(t))^T dt$ and $\int B^Y(t)[y_N(t) - (B^Y(t))^T \hat{h}_k^{(N-1)}] dt$.

To calculate $\int\Omega B^Y(t)(B^Y(t))^T dt$, we make an evenly partition of $\Omega$, and compute the integral by (A.31):

$$\int\Omega B^Y(t)(B^Y(t))^T dt = \sum_{j=1}^{M} B^Y(t_j)^T B^Y(t_j) \delta,$$  \tag{A.31}

where $B^Y(t_j) = (B^Y_1(t_j), B^Y_2(t_j), \ldots, B^Y_Q(t_j))$ and $\delta = t_{j+1} - t_j$ with $j = 1, 2, \ldots, M - 1$.

As $y_N(t)$ are transformed into functional data type i.e. basis spline $(B_1^Y(t), B_2^Y(t), \ldots, B_Q^Y(t))^T$ with coefficients $h_N = (h_{N1}, h_{N2}, \ldots, h_{NQ})$. Thus, $\int B^Y(t)[y_N(t) - (B^Y(t))^T \hat{h}_k^{(N-1)}] dt$ can be computed using (A.31), the evidence of which can be found from (A.32):

$$\int B^Y(t)[y_N(t) - (B^Y(t))^T \hat{h}_k^{(N-1)}] dt \tag{A.32}$$

$$= \int B^Y(t)[(B^Y(t))^T \hat{h}_k^{(N)} - (B^Y(t))^T \hat{h}_k^{(N-1)}] dt$$

$$= \int B^Y(t)(B^Y(t))^T dt[\hat{h}_k^{(N)} - \hat{h}_k^{(N-1)}].$$

**A.4 Additional Numerical Results in Chapter 5**

**A.4.1 Complexity Analysis of LEOA**

In this section, the computational burden of the LEOA is discussed. In a similar way to many existing evolving fuzzy approaches, the complexity of LEOA depends on the
number of fuzzy rules \( K \) and the dimension of input \( n \). From the structure of LEOA and its detailed steps, it can be seen that the complexity of the whole process, which includes rule generation, merging and pruning process, is \( O(Kn) \). However, the new parameters updating procedure has the complexity \( O(K^2) \), which would increase the complexity when the rule number \( K \) is larger than \( n \). This is because the proposed EWRLS is required to remember more information than the widely used WRLS. This is in order to meet the requirement of optimality. Compared to state-of-the-art methods such as eTS [27], DENFIS [26], FLEXFIS [28], DEC [35], PANFIS [55] and Gen-Smart-EFS [58], LEOA is required to store more historical information and has a more complex structure when \( K > n \). Nonetheless, LEOA takes into account the changes of antecedent structure for convergency of consequent parameters, and ensures the optimality of consequent parameters instead of the sub-optimality presented in most of the existing approaches. However, the high computational complexity is a price that must be paid for it.

### A.4.2 Numerical Results for EFFS

The following tables are detailed results for EFFS when setting different threshold parameters. The results listed in Table A.1 – A.9 are the results of Example 1. Table A.10 – A.18 are the results of Example 2. Table A.1–A.3 and Table A.10–A.12 are the testing results with spline basis order equal to 2, Table A.4–A.6 and Table A.13–A.15 are the results with spline order equal to 3, and Table A.7–A.9 and Table A.16–A.18 are with spline order equal to 4. The results which are presented in “red” are those picked as best results and listed in section 6.5.1 and section 6.5.2.

<table>
<thead>
<tr>
<th>( \varepsilon_{\text{add}} )</th>
<th>( \varepsilon_{\text{merge}} )</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.0134</td>
<td>0.0135</td>
<td>0.0135</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0128</td>
<td>0.0123</td>
<td>0.0125</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0129</td>
<td>0.0123</td>
<td>0.0120</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0120</td>
<td>0.0115</td>
<td>0.0113</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0118</td>
<td>0.0117</td>
<td>0.0115</td>
</tr>
<tr>
<td>0.6</td>
<td>0.0118</td>
<td>0.0113</td>
<td>0.0115</td>
</tr>
</tbody>
</table>
### APPENDIX A. MATHEMATICAL INDUCTIONS, PROOFS AND ADDITIONAL NUMERICAL RESULTS

#### TABLE A.2: Example 1: Online testing MSPEs (spline order = 2).

<table>
<thead>
<tr>
<th>$\varepsilon_{\text{add}}$</th>
<th>$\varepsilon_{\text{merge}}$</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0151</td>
<td>0.0152</td>
<td>0.0152</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0147</td>
<td>0.0143</td>
<td>0.0143</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0148</td>
<td>0.0142</td>
<td>0.0139</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0140</td>
<td>0.0135</td>
<td>0.0133</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0135</td>
<td>0.0134</td>
<td>0.0131</td>
</tr>
<tr>
<td>0.6</td>
<td>0.0134</td>
<td>0.0129</td>
<td>0.0129</td>
</tr>
</tbody>
</table>

#### TABLE A.3: Example 1: Average number of rules (spline order = 2).

<table>
<thead>
<tr>
<th>$\varepsilon_{\text{add}}$</th>
<th>$\varepsilon_{\text{merge}}$</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>0.1</td>
<td>5.0608</td>
<td>5.5374</td>
<td>5.5413</td>
</tr>
<tr>
<td>0.2</td>
<td>6.0057</td>
<td>6.7089</td>
<td>6.9409</td>
</tr>
<tr>
<td>0.4</td>
<td>10.5824</td>
<td>12.4307</td>
<td>14.9327</td>
</tr>
<tr>
<td>0.6</td>
<td>20.5363</td>
<td>25.9353</td>
<td>31.5342</td>
</tr>
</tbody>
</table>

#### TABLE A.4: Example 1: Testing error with 50 pairs for training (spline order = 3).

<table>
<thead>
<tr>
<th>$\varepsilon_{\text{add}}$</th>
<th>$\varepsilon_{\text{merge}}$</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.95</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0133</td>
<td>0.0134</td>
<td>0.0134</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0129</td>
<td>0.0122</td>
<td>0.0123</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0128</td>
<td>0.0125</td>
<td>0.0122</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0120</td>
<td>0.0114</td>
<td>0.0112</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0117</td>
<td>0.0116</td>
<td>0.0114</td>
</tr>
<tr>
<td>0.6</td>
<td>0.0116</td>
<td>0.0112</td>
<td>0.0111</td>
</tr>
</tbody>
</table>
### TABLE A.5: Example 1: Online testing MSPEs (spline order = 3).

| $\varepsilon_{\text{add}}$ | $\varepsilon_{\text{merge}}$ | Number of Basis = 10 |  | Number of Basis = 20 |  |
|-----------------------------|-----------------------------|----------------------|-----------------------------|-----------------------------|
|                             |                             | 0.95                | 0.96                        | 0.97                        | 0.98                        | 0.95                | 0.96                        | 0.97                        | 0.98                        |
| 0.1                         |                             | 0.0150              | 0.0150                      | 0.0150                      | 0.0150                      | 0.0161              | 0.0160                      | 0.0160                      | 0.0160                      |
| 0.2                         |                             | 0.0147              | 0.0141                      | 0.0142                      | 0.0141                      | 0.0157              | 0.0156                      | 0.0149                      | 0.0148                      |
| 0.3                         |                             | 0.0146              | 0.0142                      | 0.0140                      | 0.0138                      | 0.0154              | 0.0147                      | 0.0142                      | 0.0142                      |
| 0.4                         |                             | 0.0139              | 0.0134                      | 0.0132                      | 0.0131                      | 0.0148              | 0.0139                      | 0.0141                      | 0.0138                      |
| 0.5                         |                             | 0.0134              | 0.0133                      | 0.0131                      | **0.0128**                  | 0.0137              | 0.0138                      | 0.0134                      | 0.0131                      |
| 0.6                         |                             | 0.0131              | 0.0128                      | 0.0127                      | 0.0128                      | 0.0134              | 0.0129                      | 0.0132                      | **0.0128**                  |

### TABLE A.6: Example 1: Average number of rules(spline order = 3).

| $\varepsilon_{\text{add}}$ | $\varepsilon_{\text{merge}}$ | Number of Basis = 10 |  | Number of Basis = 20 |  |
|-----------------------------|-----------------------------|----------------------|-----------------------------|-----------------------------|
|                             |                             | 0.95                | 0.96                        | 0.97                        | 0.98                        | 0.95                | 0.96                        | 0.97                        | 0.98                        |
| 0.1                         |                             | 5.0608              | 5.5374                      | 5.5413                      | 5.5430                      | 5.0788              | 5.1053                      | 5.1113                      | 5.1113                      |
| 0.2                         |                             | 5.8649              | 6.7089                      | 6.9409                      | 7.3798                      | 6.5119              | 6.5520                      | 7.9285                      | 8.0709                      |
| 0.3                         |                             | 7.1730              | 7.8146                      | 8.8521                      | 9.5355                      | 7.6315              | 8.4742                      | 10.4111                     | 10.6912                     |
| 0.4                         |                             | 10.6022             | 12.3447                     | 14.7611                     | 17.2671                     | 8.8053              | 10.7429                     | 13.4615                     | 15.9278                     |
| 0.5                         |                             | 12.6913             | 15.2733                     | 20.7786                     | **25.2081**                | 12.4598             | 14.8174                     | 20.5476                     | 25.4859                     |
| 0.6                         |                             | 21.7778             | 25.7377                     | 30.3243                     | 36.7996                     | 21.1623             | 28.3458                     | 34.9418                     | **37.6943**                 |

### TABLE A.7: Example 1: Testing errors with 50 pairs for training (spline order = 4).

| $\varepsilon_{\text{add}}$ | $\varepsilon_{\text{merge}}$ | Number of Basis = 10 |  | Number of Basis = 20 |  |
|-----------------------------|-----------------------------|----------------------|-----------------------------|-----------------------------|
|                             |                             | 0.95                | 0.96                        | 0.97                        | 0.98                        | 0.95                | 0.96                        | 0.97                        | 0.98                        |
| 0.1                         |                             | 0.0134              | 0.0135                      | 0.0135                      | 0.0135                      | 0.0139              | 0.0139                      | 0.0139                      | 0.0139                      |
| 0.2                         |                             | 0.0130              | 0.0123                      | 0.0124                      | 0.0122                      | 0.0137              | 0.0136                      | 0.0126                      | 0.0125                      |
| 0.3                         |                             | 0.0129              | 0.0123                      | 0.0120                      | 0.0119                      | 0.0134              | 0.0127                      | 0.0119                      | 0.0119                      |
| 0.4                         |                             | 0.0123              | 0.0115                      | 0.0113                      | 0.0111                      | 0.0127              | 0.0116                      | 0.0119                      | 0.0114                      |
| 0.5                         |                             | 0.0123              | 0.0120                      | 0.0119                      | 0.0113                      | 0.0119              | 0.0120                      | 0.0117                      | 0.0113                      |
| 0.6                         |                             | **0.0116**          | 0.0113                      | **0.0116**                  | **0.0111**                  | **0.0117**          | 0.01145                     | 0.0114                      | 0.0112                      |
### APPENDIX A. MATHEMATICAL INDUCTIONS, PROOFS AND ADDITIONAL NUMERICAL RESULTS

214

<table>
<thead>
<tr>
<th>Table A.8: Example 1: Online testing MSPEs (spline order = 4).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of Basis</strong></td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td><strong>ε</strong></td>
</tr>
<tr>
<td><strong>merge</strong></td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.2</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.4</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table A.9: Example 1: Average number of rules (spline order = 4).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of Basis</strong></td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td><strong>ε</strong></td>
</tr>
<tr>
<td><strong>merge</strong></td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.2</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table A.10: Example 2: Testing errors with 50 pairs for training (spline order = 2).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of Basis</strong></td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td><strong>ε</strong></td>
</tr>
<tr>
<td><strong>merge</strong></td>
</tr>
<tr>
<td>0.1</td>
</tr>
<tr>
<td>0.2</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.4</td>
</tr>
<tr>
<td>0.5</td>
</tr>
<tr>
<td>0.6</td>
</tr>
</tbody>
</table>
**APPENDIX A. MATHEMATICAL INDUCTIONS, PROOFS AND ADDITIONAL NUMERICAL RESULTS**

### TABLE A.11: Example 2: Online testing MSPEs (spline order = 2).

<table>
<thead>
<tr>
<th>$\epsilon_{\text{merge}}$</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon_{\text{add}}$</td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0050 0.0050 0.0049 0.0048</td>
<td>0.0052 0.0052 0.0051 0.0048</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0052 0.0052 0.0051 0.0047</td>
<td>0.0055 0.0055 0.0051 0.0048</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0052 0.0053 0.0050 0.0046</td>
<td>0.0054 0.0053 0.0049 0.0049</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0052 0.0051 0.0046 0.0045</td>
<td>0.0054 0.0052 0.0046 0.0045</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0051 0.0048 0.0047 0.0043</td>
<td>0.0052 0.0049 0.0046 0.0045</td>
</tr>
<tr>
<td>0.6</td>
<td>0.0049 0.0048 0.0045 0.0043</td>
<td>0.0049 0.0048 0.0045 0.0042</td>
</tr>
</tbody>
</table>

### TABLE A.12: Example 2: Average number of rules (spline order = 2).

<table>
<thead>
<tr>
<th>$\epsilon_{\text{merge}}$</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon_{\text{add}}$</td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
</tr>
<tr>
<td>0.1</td>
<td>1.0314 1.0314 1.4314 2.3451</td>
<td>1.0314 1.0314 1.4863 2.3804</td>
</tr>
<tr>
<td>0.2</td>
<td>1.0431 1.0431 1.4431 2.4784</td>
<td>1.0431 1.0431 1.4902 2.3882</td>
</tr>
<tr>
<td>0.3</td>
<td>1.1490 1.3176 1.8745 4.1412</td>
<td>1.1451 1.7490 1.9216 3.9843</td>
</tr>
<tr>
<td>0.4</td>
<td>1.4549 2.1373 2.6902 3.5412</td>
<td>1.6314 2.6863 3.2196 4.1137</td>
</tr>
<tr>
<td>0.5</td>
<td>3.6588 4.1608 5.3137 7.9490</td>
<td>2.5490 3.5333 5.4510 9.6824</td>
</tr>
<tr>
<td>0.6</td>
<td>4.2431 4.8235 11.7725 11.3451</td>
<td>6.8157 9.2549 17.3020 24.7765</td>
</tr>
</tbody>
</table>

### TABLE A.13: Example 2: Testing errors with 50 pairs for training (spline order = 3).

<table>
<thead>
<tr>
<th>$\epsilon_{\text{merge}}$</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon_{\text{add}}$</td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0053 0.0053 0.0052 0.0050</td>
<td>0.0053 0.0053 0.0051 0.0049</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0056 0.0056 0.0055 0.0050</td>
<td>0.0057 0.0057 0.0051 0.0048</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0055 0.0057 0.0050 0.0049</td>
<td>0.0055 0.0055 0.0050 0.0048</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0056 0.0055 0.0050 0.0048</td>
<td>0.0056 0.0053 0.0046 0.0049</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0055 0.0052 0.0049 0.0048</td>
<td>0.0053 0.0050 0.0044 0.0043</td>
</tr>
<tr>
<td>0.6</td>
<td>0.0051 0.0049 0.0047 0.0047</td>
<td>0.0047 0.0045 0.0042 0.0041</td>
</tr>
</tbody>
</table>
### APPENDIX A. MATHEMATICAL INDUCTIONS, PROOFS AND ADDITIONAL NUMERICAL RESULTS

#### TABLE A.14: Example 2: Online testing MSPEs (spline order = 3).

<table>
<thead>
<tr>
<th>$\varepsilon_{\text{add}}$</th>
<th>$\varepsilon_{\text{merge}}$</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>0.0053 0.0054 0.0052 0.0049</td>
<td>0.0053 0.0053 0.0051 0.0050</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.0055 0.0055 0.0054 0.0049</td>
<td>0.0056 0.0056 0.0051 0.0048</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.0055 0.0056 0.0050 0.0049</td>
<td>0.0054 0.0054 0.0050 0.0049</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.0055 0.0054 0.0049 0.0047</td>
<td>0.0055 0.0052 0.0046 0.0048</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.0054 0.0050 0.0048 0.0047</td>
<td>0.0052 0.0049 0.0045 0.0044</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.0051 0.0048 $\textbf{0.0046}$ 0.0046</td>
<td>0.0047 0.0045 0.0043 $\textbf{0.0042}$</td>
<td></td>
</tr>
</tbody>
</table>

#### TABLE A.15: Example 2: Average number of rules (spline order = 3).

<table>
<thead>
<tr>
<th>$\varepsilon_{\text{add}}$</th>
<th>$\varepsilon_{\text{merge}}$</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>1.0314 1.0314 1.4314 2.4275</td>
<td>1.0314 1.0314 1.4863 2.3412</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1.0431 1.0431 1.4431 2.4314</td>
<td>1.0431 1.0431 1.4902 2.4000</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1.1490 1.3294 1.8824 4.1451</td>
<td>1.1451 1.7490 2.3333 3.9843</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>1.3529 1.6902 2.2706 5.5137</td>
<td>1.6314 2.6824 3.2196 4.4706</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>2.2274 2.6510 3.6078 7.5098</td>
<td>2.6235 4.9529 6.2588 10.9294</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>5.7451 9.9020 $\textbf{17.1529}$ 19.3137</td>
<td>14.7725 18.7020 23.9961 $\textbf{36.7020}$</td>
<td></td>
</tr>
</tbody>
</table>

#### TABLE A.16: Example 2: Testing errors with 50 pairs for training (spline order = 4).

<table>
<thead>
<tr>
<th>$\varepsilon_{\text{add}}$</th>
<th>$\varepsilon_{\text{merge}}$</th>
<th>Number of Basis = 10</th>
<th>Number of Basis = 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.95 0.96 0.97 0.98</td>
<td>0.95 0.96 0.97 0.98</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>0.0055 0.0055 0.0054 0.0052</td>
<td>0.0054 0.0054 0.0052 0.0050</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.0057 0.0057 0.0056 0.0051</td>
<td>0.0057 0.0057 0.0052 0.0048</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.0057 0.0057 0.0052 0.0052</td>
<td>0.0056 0.0055 0.0049 0.0049</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.0057 0.0055 0.0052 0.0050</td>
<td>0.0056 0.0054 0.0047 0.0049</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.0056 0.0052 0.0050 0.0050</td>
<td>0.0054 0.0050 0.0044 0.0044</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.0055 0.0051 0.0049 $\textbf{0.0049}$</td>
<td>0.0048 0.0045 0.0042 $\textbf{0.0041}$</td>
<td></td>
</tr>
</tbody>
</table>
**TABLE A.17: Example 2: Online testing MSPEs (spline order = 4).**

<table>
<thead>
<tr>
<th>ε</th>
<th>ε_add</th>
<th>0.0055</th>
<th>0.0055</th>
<th>0.0054</th>
<th>0.0052</th>
<th>0.0053</th>
<th>0.0053</th>
<th>0.0052</th>
<th>0.0050</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.0055</td>
<td>0.0055</td>
<td>0.0054</td>
<td>0.0052</td>
<td>0.0053</td>
<td>0.0053</td>
<td>0.0052</td>
<td>0.0050</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.0058</td>
<td>0.0058</td>
<td>0.0057</td>
<td>0.0052</td>
<td>0.0056</td>
<td>0.0056</td>
<td>0.0052</td>
<td>0.0049</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.0058</td>
<td>0.0058</td>
<td>0.0053</td>
<td>0.0052</td>
<td>0.0054</td>
<td>0.0054</td>
<td>0.0049</td>
<td>0.0050</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.0058</td>
<td>0.0057</td>
<td>0.0053</td>
<td>0.0052</td>
<td>0.0055</td>
<td>0.0052</td>
<td>0.0047</td>
<td>0.0048</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.0057</td>
<td>0.0054</td>
<td>0.0051</td>
<td>0.0050</td>
<td>0.0052</td>
<td>0.0049</td>
<td>0.0045</td>
<td>0.0045</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.0056</td>
<td>0.0052</td>
<td>0.0051</td>
<td>0.0050</td>
<td>0.0047</td>
<td>0.0046</td>
<td>0.0043</td>
<td>0.0042</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE A.18: Example 2: Average number of rules (spline order = 4).**

<table>
<thead>
<tr>
<th>ε</th>
<th>ε_add</th>
<th>0.95</th>
<th>0.96</th>
<th>0.97</th>
<th>0.98</th>
<th>0.95</th>
<th>0.96</th>
<th>0.97</th>
<th>0.98</th>
<th>0.95</th>
<th>0.96</th>
<th>0.97</th>
<th>0.98</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1.0314</td>
<td>1.3014</td>
<td>1.4314</td>
<td>2.4549</td>
<td>1.0314</td>
<td>1.3014</td>
<td>1.4863</td>
<td>2.3412</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1.0431</td>
<td>1.0431</td>
<td>1.4431</td>
<td>2.4588</td>
<td>1.0431</td>
<td>1.0431</td>
<td>1.4902</td>
<td>2.4039</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1.1608</td>
<td>1.3843</td>
<td>1.8941</td>
<td>5.0941</td>
<td>1.1569</td>
<td>1.7686</td>
<td>1.9333</td>
<td>3.9843</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>1.1588</td>
<td>2.1569</td>
<td>2.4157</td>
<td>5.7216</td>
<td>1.6314</td>
<td>2.6863</td>
<td>3.2196</td>
<td>4.4706</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>1.9529</td>
<td>2.6745</td>
<td>3.8706</td>
<td>6.2863</td>
<td>2.6235</td>
<td>4.1529</td>
<td>6.3373</td>
<td>8.9765</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>5.9373</td>
<td>14.5451</td>
<td>18.6706</td>
<td>24.1961</td>
<td>15.9451</td>
<td>17.4784</td>
<td>24.0196</td>
<td>34.8196</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

Benchmark Examples & Datasets

Benchmark examples and datasets used from Chapter 3 to Chapter 6 are presented in this chapter. Section B.1 gives the detailed descriptions of the benchmark examples and datasets for evolving fuzzy systems used in Chapter 3 and Chapter 4. While examples and datasets used for evaluate the functional fuzzy systems in Chapter 5 and Chapter 6 are shown in Section B.2.

B.1 Benchmark Examples & Datasets for Evolving Fuzzy Systems

The accuracy for EFS is judged by RMSE (B.1), NDEI (B.2) and MAE (B.3),

\[ RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i^{est})^2}, \]  

\[ NDEI = \frac{RMSE}{std(y)}, \]  

\[ MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i^{est}|. \]

Benchmark examples used for comparison in this thesis are shown as follows.

1. **Nonlinear Dynamic System Identification**

   The system is used as a benchmark example in previous research works such as [27, 61,
It is defined by the following equation:

\[ y(t + 1) = \frac{y(t)}{1 + y^2(t)} + u^3(t), \]  

(B.4)

where \( u(t) = \sin(2\pi t/100) \). The inputs and corresponding outputs have the form \((y(t), u(t))\) and \(y(t + 1)\), respectively. In this example, the initial values of \((y(t), u(t))\) are \((0, 0)\) with \(u(t) \in [-1.0, 1.0]\) and \(y(t) \in [-1.5, 1.5]\), besides, there are totally 50200 samples \(\{y(t)\}_{t=1}^{50200}\) are generated, in which first 50000 data points are used for training and the remaining 200 data points are for the purpose of testing.

2. Nonlinear Dynamic System with Time-varying Characteristics

The same as [69], the model is:

\[ y(t + 1) = \frac{y(t)}{1 + y^2(t)} + u^3(t) + n(t), \]  

(B.5)

where \( u(t) = \sin(2\pi t/100) \), and \(n(t)\) is a time-varying factor, which could be presented as:

\[
    n(t) = \begin{cases} 
    0, & 1 \leq t \leq 1000 \text{ and } t \geq 2001 \\
    0.5, & 1001 \leq t \leq 1500 \\
    1, & 1501 \leq t \leq 2000 
    \end{cases} 
\]  

(B.6)

There are 3000 data points with \(t \in [1, 3000]\) generated, and all of them are used for online learning and testing. \((u(t), y(t))\) is the input, and \(y(t + 1)\) is the output.

3. High-dimensional System Identification Problem

The system is defined by the following equation (B.7):

\[
    y(t) = \frac{\sum_{i=1}^{m} y(t-i)}{1 + \sum_{i=1}^{m} y^2(t-i)} + u(t-1),
\]

(B.7)

where \( u(t) = \sin(2\pi t/20), m = 10, y(j) = 0, \) for \( j = 1, 2, \ldots, m \). The prediction model is of the form shown in (B.8):

\[
    \hat{y}(t) = f(y(t-1), y(t-2), \ldots, y(t-10), u(t-1)).
\]

(B.8)

There are 3300 data points produced with \(t \in [1, 3300]\). Only the example in Section 3.2.4.2 uses first 1650 as training data and the remaining 1650 as the testing data. Example in Section 3.3.4.4 uses the first 3000 for training, and only the last 300 data points are used
to evaluate the performance of the algorithms. In this example, \(y(t-1), y(t-2), \ldots, y(t-10), u(t-1)\) are used to forecast \(y(t)\).

4. Nonlinear Dynamic Plant

The model form of nonlinear dynamic plant is shown in (B.9):

\[
y(t + 1) = \frac{y(t)y(t-1)[y(t) - 0.5]}{1 + y^2(t) + y^2(t-1)} + u(t), \quad \text{(B.9)}
\]

where \(u(t) = \sin(2t/25), t \in [1, 5200]\). The first 5000 pairs of \((y(t-1), y(t), u(t))\) are used for training and the remaining 200 pairs are applied for forecasting.

5. Mackey-Glass Chaotic Time Series

Mackey-Glass Chaotic time series prediction example is a widely used benchmark example in existing works such as [27, 35, 56, 69, 89] and [96]. Data is generated from the following system:

\[
\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t), \quad \text{(B.10)}
\]

where \(\tau = 17\) and \(x(0) = 1.2\). There are totally 6000 observations generated. Three thousand data points from \(201 \leq t \leq 3200\) are used for online training, and 500 data points ranging from \(5001 \leq t \leq 5500\) are used for testing. The prediction model is of the form:

\[
\hat{x}(t + 85) = f(x(t-18), x(t-12), x(t-6), x(t)). \quad \text{(B.11)}
\]

6. Box-Jenkins gas furnace dataset

The Box-Jenkins dataset consists of 290 input-output pairs. Methane flow rate \(u(t)\), and \(CO_2\) concentration in off gas \(y(t)\) comprise the input of each data sample. The output of the process is \(y(t)\). From previous studies, it can be seen that the best model of this process is (B.12):

\[
y(t) = f(y(t-1), u(t-4)). \quad \text{(B.12)}
\]

7. S&P 500 Daily Closing Price

This dataset contains 60 years daily closing price of S&P500 collected from Yahoo! Finance website ranging from 03.01.1950 to 12.03.2009. There are totally 14893 data points. We use the SEFS to make online predictions for the original time series and the flipped time series with 29786 data points. The prediction model is of the form:

\[
\hat{x}(t + 1) = f(x(t-4), x(t-3), x(t-2), x(t-1), x(t)). \quad \text{(B.13)}
\]

The original dataset is used for training, and the flipped time series is applied for testing.
APPENDIX B. BENCHMARK EXAMPLES & DATASETS

8. **Delta Ailerons Data**
   Delta Ailerons data from KEEL-dataset are applied in this example. They are obtained from the task of controlling the ailerons of a F16 aircraft. Five inputs named as “RollRate”, “PitchRate”, “currPitch”, “currRoll”, “diffRollRate” are applied to predict the output “Sa”. The same as [81], we use the first 3000 points for training and the remaining 4129 points for evaluation.

9. **Helicopter unmanned aerial vehicle streaming data**
   The Rotary Unmanned Aerial Vehicles (RUAV) dataset is collected from an Align Trex450 Pro Direct Flight Control (DFC), fly bar-less, helicopter made in Taiwan. We use the same dataset conducted by the UAV laboratory of the UNSW Canberra campus, as [210]. The dataset is formed by 6000 samples, which are highly nonlinear and nonstationary. The first 3600 samples are used for training, and the remainder are used for testing. The input and output are set the same as [210].

10. **Data Sets from UCI Repository**
    MPG and Boston Housing data taken from UCI repository are used. Follow from [58], the evaluation procedure follows a 10-fold cross-validation.

### B.2 Benchmark Examples & Datasets for Functional Fuzzy Systems

Several datasets, which have been applied to compare with other approaches in this thesis, are specifically described as follows.

1. **Diffusion Tensor Imaging (DTI) Data**
   Diffusion tensor imaging (DTI) data are widely used as benchmark in functional data analysis. DTI data could be obtained from the R package ‘refund’. DTI data set contains 382 DTI scans of 142 subjects, 100 of which have multiple sclerosis (MS) and 42 are healthy. DTI scans contain the Fractional anisotropy (FA) tract profiles for the corpus callosum (CCA-FA) and the right corticospinal tract (RCST-FA). Fig.B.1 presents the curves of the CCA-FA and RCST-FA curves from the DTI data set. Examples using the DTI dataset focus on investigating the relationship between the CCA-FA and RCST-FA curves. Follow from [160], the curves that contain incomplete data are removed. Then, there remains 255 pairs of CCA and RCST curves. CCA-FA curves are used as the input $x_i(s)$ of the model, and the RCST-FA curves are treated as output curves $y_i(t)$. 
2. Capital Bike Share Data

Bike renting is an attractive area that attracts a lot of attention from people and organizations from all over the world in recent years; thus, it is important and meaningful to study the bike supply problem, on the one hand, to guarantee a sufficient number of bikes available to share; on the other hand, to prevent loads of redundant bikes and avoid the waste of resources. The relationship between the temperature and the bike rentals need to be investigated. The capital bike share data are collected by the Capital Bike Share system in Washington, DC. This system offers bike rental services on an hourly basis. The same as [176], we are focusing on the casual bike rentals instead of those with membership in the Capital Bike share program. The casual bike rentals are collected hourly from 1.1.2011 to 31.12.2012. The weather information (temperature (°C)) is also recorded hourly. There are 105 curves (each curve contains 24 hourly observations) for both bike rentals and the temperature data, respectively. In Fig. B.2, the figure on the left presents the curves of the temperature data, and the picture on the right side reflects the correspondence renting condition of the casual bikes. Removing the incomplete data from the data set, there remains 102 pairs of temperature and bike rentals curves. Temperature curves are used as predictors, and the hourly bike rental amount curves are performed as responses. In order to remove the skewness, logarithm transform \((x \rightarrow \log(x + 1))\) is operated on the responses.

3. Electricity Consumption of France with Missing Values

Electricity usage and energy consumption prediction is a crucial topic in environment protection and energy saving. The quality of the predictions can help to balance the supply and demand of electricity. With the accurate forecasts of the everyday curve of energy consumption, the energy suppliers would know how much energy (e.g. electricity, gas) would be consumed in every hour. This helps the energy suppliers to use the natural resources more economically and environmental friendly. RTE is the electricity transmission system operator of France. It provides the predictions of everyday electricity usage of metropolitan France (except Corsica). The forecasts provided by RTE are relying on meteorological data (temperature and cloudiness) and the consumption histories over France. In this example, we attempt to learn whether fewer information (compared with RTE) could also achieve preferable predictions of the electricity consumptions. As the political and economic center of France, Paris has the largest population over all the French metropolises. The electricity consumption of Paris has an extremely important influence to the whole consumption of the country. Therefore, we aim at learning the relationship between the everyday temperature of Paris and the electricity consumption of France, and using these daily temperature curves to make predictions of daily electricity consumption curves.
APPENDIX B. BENCHMARK EXAMPLES & DATASETS

Fig. B.1: DTI data.

Fig. B.2: Casual bike users and hourly temperature.

Fig. B.3: The daily temperature and electricity consumption of France in 2017.

The temperature data are collected from the website of ‘wunderground’\(^1\). The temperature data of Charles de Gaulle airport, Paris, from 2017-01-01 to 2017-12-31 are collected. The historical electricity consumption data of 2017 are obtained from the archives of RTE.

\(^1\)https://www.wunderground.com
Both temperature and electricity consumption data are collected every 30 minutes, so there should be 48 observations everyday. For the temperature observations, which share the same time stamp in a day, we reserve only the first observations. Besides, the temperature data set contains missing values, in which case we only remove the daily curves that have the first observation (with time stamp 12:00 AM) or the last observation (with time stamp 11:30 PM) empty. After this preprocess, we get 361 daily curves for each temperature and electricity usage. Furthermore, logarithm transformations ($x \rightarrow \log(x+10)$ and $x \rightarrow \log x$) are applied to the temperature and electricity consumption data, respectively, for centralization. The figures of these pairs of data are shown in Fig.B.3.
Bibliography


BIBLIOGRAPHY


