
Concept Drift Adaptation for Real-time Prediction

*A thesis submitted in partial fulfilment of the requirements
for the degree of*

Doctor of Philosophy
in
Computer Science

by

Yiliao Song

to

School of Computer Science, Faculty of Engineering and
Information Technology
Australian Artificial Intelligence Institute

University of Technology Sydney
NSW - 2007, Australia

August 2020

© 2020 by Yiliao Song
All Rights Reserved

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Yiliao Song* declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy in the School of Computer Science at the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:

SIGNATURE: _____
Signature removed prior to publication.

DATE: August, 2020

ABSTRACT

Concept drift refers to the phenomenon of distribution changes in a data stream. Using concept drift adaptation techniques to predict the target variable(s) of real-time data streams has gained the ever-increasing attention of researchers in recent years.

This research aims to develop a set of concept drift adaptation methods for predicting the target variable of real-time data streams. The literature review reveals two issues in the area of concept drift: i) how the concept drift problem limits the learning capability; ii) how to make adaptation in more realistic scenarios that data streams have uncertainties other than concept drift.

To address the issue i), this research discovers three root causes of limited learning capability when concept drift occurs. It is found that when concept drift occurs in a data stream, the prediction accuracy is decreased because 1) the training set contains more than one patterns so that the predictor cannot be well-learned; 2) a newly arrived data instance may present old patterns but an old instance presents the new pattern; and 3) few data instances are available when a new concept is identified at its early stage. Three concept drift adaptation methods are designed to address the three situations separately. Situation 1) is solved by developing a *fuzzy clustering-based adaptive regression* (FUZZ-CARE) approach. FUZZ-CARE can learn how many patterns exist in the training set and the membership degree of each instance belonging to each pattern; To learn the predictor with the most relevant data rather than the newest arrived data, a *segment-based drift adaptation* (SEGA) method to sequentially pick out the best segments in the training data to update the predictors. This addresses the situation 2). An *adaptive fuzzy network* (AFN) is designed to address the situation 3) through generating samples of the new concept with the previous data instances.

To address the issue ii), this research discusses the concept drift phenomenon under two scenarios that are more realistic. One is to solve the concept drift problem when data is noisy. A *noise-tolerant drift adaptation* (NoA) method is designed for handling concept drift when the data stream contains signal

noise; the other is to solve the concept drift problem when data also contains temporal dependency. A theoretical study is conducted for the regression of data streams with concept drift and temporal dependency, and based on this study, a *drift-adapted regression* (DAR) framework is established.

To conclude, this thesis not only provides a set of effective drift adaptation methods for real-time prediction, but also contributes to the development of concept drift area.

DEDICATION

To my loving husband and parents.

ACKNOWLEDGMENTS

It is a memorial and exciting journey at University of Technology Sydney (UTS) for pursuing my Ph.D. degree in the past four years. I am sincerely grateful to the people who inspired and helped me in many ways.

I would like to express my foremost and deepest gratitude to my principal supervisor, Distinguished Professor Jie Lu. Without her patience and encouragement, I would not have been able to complete this Ph.D. program. She led me into a new academic research direction, and convincingly guided and encouraged me to think and work as a professional scientist. She placed considerable trust in my research ability and unconditionally support me in pursuing my own research interests. Her wisdom and immense knowledge always enlightened me to go further and deeper in my research. Her decisiveness and sharp insights continuously motivated me when I got lost or afraid about the future. Her confidence and enthusiasm inspired me to do the right thing even when the road got tough. I felt extremely honored to be guided by such a rigorous researcher as well as an enthusiastic mentor. What she taught me and what I learned from her in the past four years has benefited my Ph.D. study and will be a great treasure throughout my life.

Meanwhile, I am greatly indebted to my co-advisor, A./Professor Guangquan Zhang. He taught me step by step how to become a qualified researcher from its beginning. He always led me to the right research direction with his expert knowledge of theory and abundant research experience. Without his critical comments, I would waste my time on trivial research ideas. Discussion with him greatly improves the scientific aspect and quality of my research. He helped me to build my confidence in my research outcomes and to be hopeful when faced with any difficulty, from academic to living.

I would like to truly thank my co-advisor, A./Professor Haiyan Lu. Without her endless trust and help, I would never have this valuable opportunity to pursue my Ph.D. research in UTS. She encouraged me to participate in different research activities which have significantly broaden my research horizons. She gave me constant care over my conditions. Her kindness helped me to go through

the tough period in my Ph.D. program. It is always pleasant to share all kinds of opinions with her.

During my Ph.D. period, I am very fortunate to join the Department of Computer Science at Southern University of Science and Technology as a visiting scholar, working with Prof. Xin Yao and Dr. Changwu Huang for three months. I would like to express my thankfulness to Prof. Xin Yao and Dr. Changwu Huang that they led me to the evolutionary computation area. Discussion and cooperation with them helped me to fill my knowledge blank of evolutionary computation efficiently and enlightened me on my own research at the same time.

I would like to express my thankfulness to every member of the Decision Systems & e-Service Intelligence Lab (DeSI) in the Centre for Artificial Intelligence (CAI) for their careful participation in my presentation and valuable comments for my research. It was a wonderful experience to spend four years with these dedicated researchers. I especially thank Dr. Ning Lu, Dr. Anjin Liu, Dr. Fan Dong and Dr. Feng Gu who helped me greatly to deeply understand my research problem during my Ph.D. candidature; Dan Shang, Hang Yu, Bin Wang and Bin Zhang who have shared their opinions and comments with me, Dr. Hua Zuo, Dr. Qian Zhang, Dr. Shan Xue who shared my joys and sadness, Dr. Dianshuang Wu who helped me in the living.

I genuinely thank Sue Felix and Robyn Barden for polishing the language of my publications. They are always patient to all my emails of questioning revised sentences. I have learned much about academic writing from them.

Meanwhile, I must thank Dr. Xin Wang and Dr. Bo Han for the valuable suggestions at the beginning of my Ph.D. study. Although they are working in different research areas, their persistence and passion on research impressed and inspired me. I thank all my wonderful friends, classmates and colleagues for every enjoyable moment.

Last, I would like to express my heartfelt appreciation and gratitude to my husband, parents, and families for their love and support.

LIST OF PUBLICATIONS

1. Y. Song, G. Zhang, H. Lu, J. Lu, "Fuzzy Clustering-based Adaptive Regression for Drifting Data Streams", *IEEE Transactions on Fuzzy Systems* , (2019). [CORE A*]
2. Y. Song, G. Zhang, H. Lu and J. Lu, "A Fuzzy Drift Correlation Matrix for Multiple Data Stream Regression", *IEEE International Conference on Fuzzy Systems* , Glasgow, Scotland (2020). [CORE A]
3. Y. Song, G. Zhang, J. Lu, H. Lu, "A Noise-tolerant Fuzzy c-Means based Drift Adaptation Method for Data Stream Regression", *IEEE International Conference on Fuzzy Systems* , New Orleans, USA (2019). [CORE A]
4. Y. Song, G. Zhang, H. Lu, J. Lu, "A Self-adaptive Fuzzy Network for Prediction in Non-stationary Environments", *IEEE International Conference on Fuzzy Systems* , Rio de Janeiro, Brazil (2018). [CORE A]
5. Y. Song, G. Zhang, J. Lu, H. Lu, "A fuzzy kernel c-means clustering model for handling concept drift in regression", *IEEE International Conference on Fuzzy Systems* , Naples, Italy (2017). [CORE A]
6. A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation", *in the 26th International Joint*

Conference on Artificial Intelligence , Melbourne, Australia (2017). [CORE A*]

7. J. Lu, A. Liu, Y. Song, G. Zhang, "Data-driven Decision Support under Concept Drift in Streamed Big Data", *Complex & Intelligent Systems* , (2019).
8. Y. Song, J. Lu, H. Lu, G. Zhang, "A Segment-based Drift Adaptation Method for Data Streams", *IEEE Transactions on Neural Networks and Learning Systems* , (submitted). [CORE A*]
9. Y. Song, J. Lu, H. Lu, G. Zhang, "A drift-adapted regression framework for data streams with temporal dependency", *Artificial Intelligence* , (submitted). [CORE A*]
10. F. Dong, J. Lu, Y. Song, F. Liu, G. Zhang, "A Drift Region-Based Data Sample Filtering Method", *IEEE Transactions on Cybernetics* , (submitted). [CORE A]

TABLE OF CONTENTS

List of Publications	ix
List of Figures	xvii
List of Tables	xxi
1 Introduction	1
1.1 Background and Motivations	1
1.2 Research Questions and Objectives	4
1.3 Research Contributions	10
1.4 Research Significance	12
1.5 Thesis Structure	14
2 Literature Review	17
2.1 Data Stream Mining	17
2.1.1 Evolving data streams	18
2.1.2 Real-time prediction	20
2.2 A General Introduction of Concept Drift	22
2.2.1 Problem description	22
2.2.2 Types of concept drift	24

TABLE OF CONTENTS

2.2.3	Concept drift applications	25
2.3	Concept Drift Adaptation	27
2.3.1	Classification of concept drift adaptation methods	28
2.3.2	Blind concept drift adaptation	30
2.3.3	Drift detection	32
2.3.4	Informed concept drift adaptation	33
2.3.5	Concept drift adaptation methods in regression cases	35
2.3.6	Drift adaptation using fuzzy techniques	37
2.3.7	Drift adaptation in a noisy environment	38
3	Drift Adaptation by Identifying Concepts by Fuzzy Clustering	
	Process	41
3.1	Introduction	42
3.2	Definitions and Notations	45
3.3	A Fuzzy Clustering-based Drift Adaptation Method—FUZZ-CARE	49
3.3.1	Embedding clustering in objective function	49
3.3.2	Finding an optimum predictor with the minimum loss	51
3.3.3	The kernel clustering-based version	53
3.3.4	The general procedure and pseudocode of FUZZ-CARE	56
3.4	Experimental Evaluations	59
3.4.1	Validation on synthetic data	62
3.4.2	Comparison on real-world data streams	70
3.4.3	Comparison on stationary datasets	75
3.4.4	Statistical test on data streams	78
3.4.5	Parameter analysis and computation complexity	81

3.5	Summary	83
4	Drift Adaptation by Sequentially Updated Data Segments	85
4.1	Introduction	86
4.2	Definitions and Notations	88
4.3	A Segment-based Drift Adaptation Method—SEGA	89
4.3.1	The segmented symmetric degree (SSD)	90
4.3.2	Drift-gradient and how to sequentially update it	94
4.3.3	The general procedure and pseudocode of SEGA	99
4.4	Experimental Evaluations	102
4.4.1	Evaluation on synthetic data	105
4.4.2	Evaluation on real-world data streams	112
4.4.3	Statistical test of real-world data streams	120
4.4.4	Parameter analysis and computation complexity	122
4.5	Summary	125
5	Drift Adaptation by Generating Samples of New Concept	127
5.1	Introduction	128
5.2	Definitions and Notations	129
5.2.1	Generative adversarial nets (GAN)	130
5.2.2	Adaptive neuro-fuzzy inference system	132
5.3	Drift Adaptation by an Adaptive Fuzzy Network—AFN	134
5.3.1	Detection by the adversarial model in GAN	135
5.3.2	Generating data by the generative model in GAN	136
5.3.3	The general procedure and pseudocode of AFN	137
5.4	Experimental Evaluations	138

TABLE OF CONTENTS

5.4.1	Experiment setup	140
5.4.2	Experimental results	142
5.5	Summary	146
6	Drift Adaptation with Noisy Data	147
6.1	Introduction	148
6.2	Definitions and Notations	150
6.3	A Noise-tolerant Drift Adaptation Method—NoA	153
6.4	Experimental Evaluations	155
6.4.1	Evaluation on synthetic data	157
6.4.2	A case study: solar radiation prediction	162
6.5	Summary	163
7	Drift Adaptation with Temporal Dependency	165
7.1	Introduction	166
7.2	Definitions and Notations	167
7.3	Drift Adaptation with Temporal Dependency	171
7.3.1	Analysis of testing error when the real drift exists	171
7.3.2	Drift adaptation procedure in DAR	179
7.4	Experimental Evaluations	186
7.4.1	Experiments on synthetic data	189
7.4.2	Experiments on real data streams	197
7.4.3	Statistical test on data Streams	200
7.5	Summary	202
8	Conclusion and Future Research	203

8.1	Conclusions	203
8.2	Future Study	206
A	Appendix	209
A.1	Derivation step from Symmetric Degree (SD) to Segmented Sym- metric Degree(SSD)	209
A.2	Estimation of model parameters for the mixed training set	211
A.3	The locally weighted regression	211
	Bibliography	213

LIST OF FIGURES

FIGURE	Page
1.1 Standard procedure of conventional machine learning methods. . . .	2
1.2 Learning with concept drift adaption.	4
1.3 Thesis structure	16
2.1 Three main challenges in evolving data streams	19
2.2 Four types of concept drift.	25
2.3 Learning components and their classification in concept drift adapta- tion methods.	29
3.1 The training set for a data stream.	42
3.2 Learn a predictor by the OLS method using the training set that contains two patterns.	43
3.3 The difference between FUZZ-CARE and other learning methods when the same input maps to two outputs. Black dots represent a 2-dimensional dataset, where two patterns exist in the same training set. The red line on the left is the fitted line of the linear regression method. The red and blue lines on the right are the fitted lines of FUZZ-CARE.	57

LIST OF FIGURES

3.4	FUZZ-CARE flowchart showing different updating strategies.	58
3.5	Generated synthetic data with different types of drift. We generate one no-drift data on which we base five examples of drifting data. The red and black dots represent two different patterns. In the sub-figures 1)–4), the dots are drawn in a 2-dimensional plane with axis of X_t and Y_t axis; while in 5) and 6) subplots, they are drawn in a 3-dimensional sphere with an additional axis of time T	65
3.6	Estimated parameters and errors of FUZZ-CARE ^{1,2} for Rec-Drift-Mix.	69
3.7	Parameter analysis of FUZZ-CARE.	82
4.1	Relationship between the definitions proposed in this chapter.	89
4.2	An example to show the drawback of a one-sided measurement.	92
4.3	Flowchart of SEGA.	102
4.4	Parameter analysis for real-world data streams of regression tasks.	123
4.5	Parameter analysis for real-world data streams of classification tasks. Usenet1 and Usenet2 are not included because they have few instances.	123
5.1	Generative Adversarial Net (GAN).	131
5.2	Adaptive neuro-fuzzy inference system (ANFIS).	132
5.3	The flowchat of AFN.	137
6.1	Synthetic noisy data stream with different types of drift.	159

- 7.1 Error decreasing process. Circles represent the scatter plots of inputs and outputs before drift occurs, while dots show the results after drift has occurred. Colors denote the different estimation results: blue is for real values, red is for LWR-LDD⁺, and black is for DAR-linear. The testing errors of 950th-1150th instances are given in the subplot E, where the gray shadow represents LWR-NoAdapt, the red dotted line represents LWR-LDD⁺ and the black line represents DAR-linear. . . 194
- 7.2 The improved percentage of MAE/MAPE of DAR-linear from LWR-LDD⁺(linear). Both methods have drift adaptation process. The difference is that DAR-linear is implemented on the reconstructed space. . 197

LIST OF TABLES

TABLE	Page
3.1 Experimental design in this section.	60
3.2 A comprehensive comparison of different editions of FUZZ-CARE method on different types of drift.	66
3.3 Validation on real-world data streams. MAE and its corresponding rank are used as the evaluations	73
3.4 Comparison results on 13 stationary datasets.	77
3.5 Friedman test and its post-hoc test of all the methods over all eight data streams.	80
3.6 Friedman test and its post-hoc test of all the methods over House, Sensor20, Sensor46, SMEAR and Solar data streams.	80
4.1 Validation of the effectiveness of drift-gradient	100
4.2 Type I and Type II error of drift-gradient under three distributions .	100
4.3 Experimental design in this section.	104
4.4 Validation of Regression Tasks on Synthetic Data (MAE as the Evalu- ation Criterion).	109
4.5 Validation on Synthetic Data of Classification Tasks (Acc as the Eval- uation Criterion).	111

LIST OF TABLES

4.6	Validation on Real-world Data of Regression Tasks (MAE as the Evaluation Criterion).	115
4.7	Validation on Real-world Data of Classification Tasks (Acc as the Evaluation Criterion).	118
4.8	Friedman test and its post-hoc test of all the methods over real-world regression data streams (no CCPP), where "Friedman Test" is the result for Friedman test and "Friedman - post-hoc test after Conover" is for the pairwise comparison. "+", "*", "**", and "***" means this value is significant at the level of 0.1, 0.05, 0.01 and 0.001 respectively. "df" denotes the freedom degree.	121
4.9	Friedman test and its post-hoc test of all the methods over real-world classification data streams, where "Friedman Test" is the result for the Friedman test and "Friedman - post-hoc test after Conover" shows the pairwise comparison. "+", "*", "**", "***" and "df" have the same meaning as they are in Table 4.8	121
4.10	Run-time on real-world data streams (s CPU-time)	126
5.1	Experimental design in this section.	139
5.2	Main results table.	144
5.3	Robustness test of AFN with different parameters.	145
6.1	The uniqueness of NFA.	150
6.2	Experimental design in this section.	156
6.3	Evaluation results on synthetic data.	161
6.4	Evaluation on the SMEAR data	163

7.1	Experimental Design	188
7.2	Parameters for generating multi-dimensional non-linear data in (7.32)	190
7.3	Comparisons between NoAdapt and adaptation by LDD ⁺ approaches on different spaces, and DAR with different predictors.	192
7.4	Comparisons of MAE between NoAdapt and adaptation by LDD ⁺ approaches on different spaces, and DAR with different predictors (multi-dimensional non-linear cases).	195
7.5	Comparisons of MAPE between NoAdapt and adaptation by LDD ⁺ approaches on different spaces, and DAR with different predictors (multi-dimensional non-linear cases).	196
7.6	MAE comparisons between different methods on real-world data streams. The DAR methods outperform the compared methods on data streams where concept drift occurs.	199
7.7	Friedman test and its post-hoc test of all the methods over all seven data streams, where "Friedman Test" is the result for Friedman test and "Friedman - post-hoc test after Conover" is for the pairwise com- parison. "*", "**", and "***" means this value is significant at the level of 0.05, 0.01 and 0.001 respectively. "df" denotes the freedom degree.	201

INTRODUCTION

1.1 Background and Motivations

In the context of the Internet of Things and Big data, the access to potentially infinite amounts of data from a variety of sources such as economics, industrial monitoring, ecosystems, and so on, generates the data stream, which is a huge volume of data that arrives in a sequential way (dos Reis et al., 2016; Haque et al., 2016b). The sequential characteristic of data streams makes the real-time prediction for a data stream different from prediction for stationary data. Each instance in a data stream is first to test the model, and then to train the model (Bifet et al., 2015).

New challenges have appeared in data stream analysis, one of which relates to unpredictable data distribution changes (Gama et al., 2013). Conventional batch-based machine learning systems are built on a static assumption of inde-

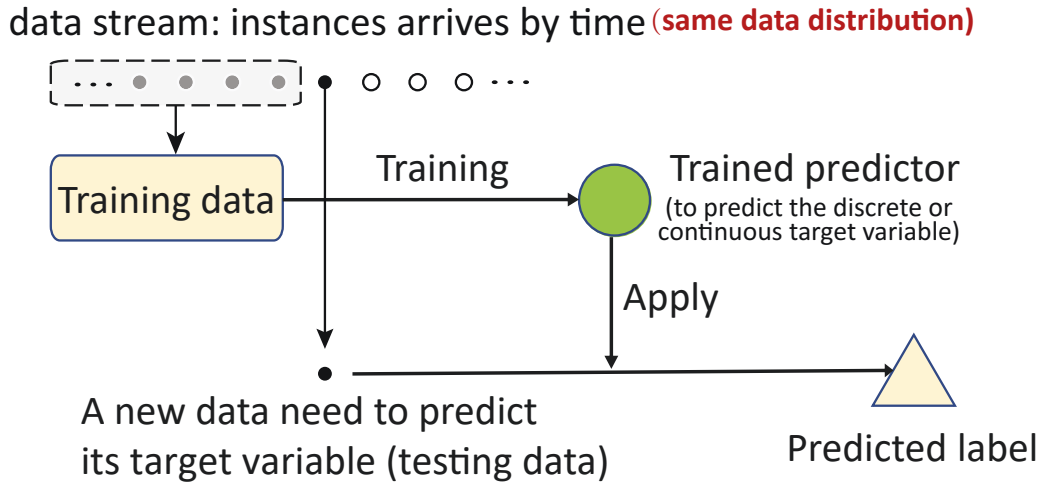


Figure 1.1: Standard procedure of conventional machine learning methods.

pendent and identically distributed (i.i.d) data and therefore are not suitable to make real-time prediction for data streams (Haque et al., 2016b). A standard training/learning and testing/prediction procedure of conventional machine learning methods is shown in Figure 1.1 which is only suitable when training data and testing data have the same distribution.

Concept drift refers to these unpredictable distribution changes over time (Schlimmer and Granger, 1986a,b) and has gained the ever-increasing attention of researchers in recent years (Lu et al., 2018). In a very recent technical report from Berkeley (Stoica et al., 2017), acting in dynamic environments and continual learning has been considered as one of nine research opportunities that can help address current AI research challenges. Unlike noise series, concept drift is influenced by hidden changes in the context (Widmer and Kubat, 1996; Kubat, 1992; Schlimmer and Granger, 1986a,b). The influence of hidden context changes results in biased estimation of the target output, and the error can be depicted

by hidden context. For example, in rain forecast, the prediction of rainfall based on atmospheric factors may change radically in different periods.

Studies on concept drift attempt, as far as possible, to learn data without these hidden variables, because hidden variables are difficult to observe and in many situations are unknown based on the current knowledge (Žliobaitė, 2010a). For example, cellphone usage was limited to communication about ten years ago, but now usage has been changed to include taking photos, searching the Internet, conducting business, learning online, and more. Customer cellphone usage prediction must consequently relinquish their models from the past. If a cellphone company continues to use an old usage pattern, it can possibly fail. In this example, one hidden context is the development of cellphone techniques. This development certainly exists but is difficult to quantify. Many subsequent studies have proved that the concept drift detection and adaptation technique is an effective way to solve the problem of distribution changes (Parker and Khan, 2015; Žliobaitė et al., 2014a). In the task of prediction for data streams, the trained predictor will be not applicable if concept drift occurs; the trained predictor will be updated by a concept drift adaptation method so that it can be used for new data that has different distribution (Figure 1.2).

Recent research of concept drift poses several unsolved and challenging problems in this area, i.e., 1) how to effectively understand concept drift to help improve adaption (Lu et al., 2018); 2) how to effectively react to drift by adapting related knowledge (Gomes et al., 2017b); 3) the lack of theoretical frameworks for learning in non-stationary environment (Ditzler et al., 2015); 4) how to solve the transient concept drift and limited data problem (Ditzler et al., 2015); and how to solve the concept drift problem if data has other uncertainties such as 5) noise

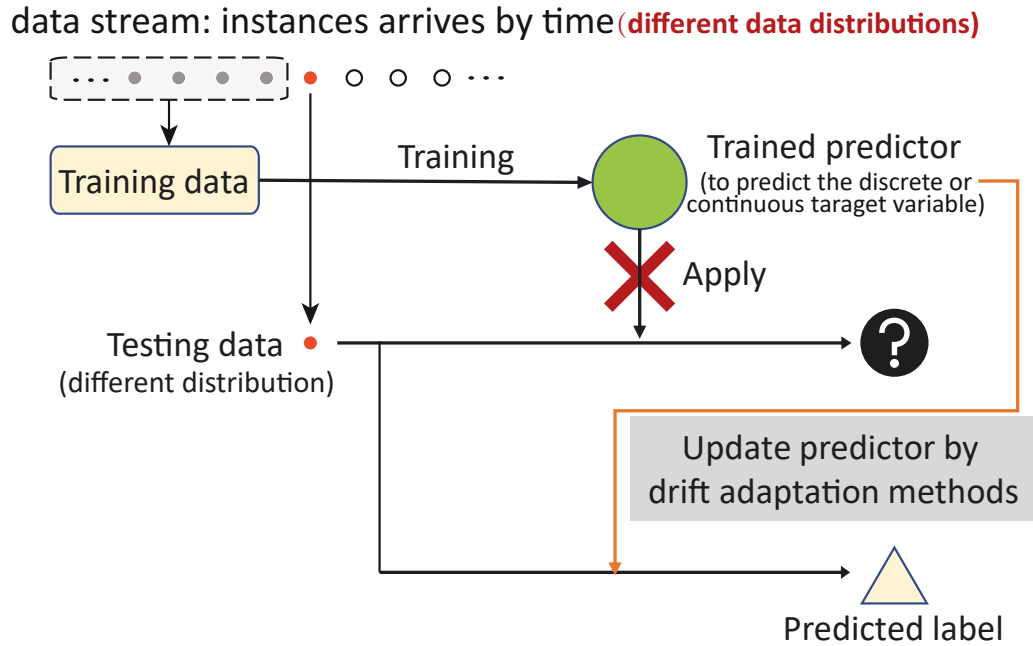


Figure 1.2: Learning with concept drift adaptation.

(Liu et al., 2018a) and 6) temporal dependencies (Gomes et al., 2017a). These problems have either impaired the drift adaptation effectiveness and limited the usage of drift adaptation methods.

This study aims to give comprehensive analysis and solutions to all the above-mentioned challenges. Chapter 3, Chapter 4 and Chapter 5 target challenge 1) and 2); Chapter 5 also provides a solution for challenge 4); Chapter 6 aims to solve challenge 5), and Chapter 7 provide solutions for challenges 3) and 6).

1.2 Research Questions and Objectives

This research aims to develop a set of concept drift adaptation methods to update the learned predictors for real-time predictions. The predictor, if not specified, is

to predict a continuous target variable. The data pattern, refers to the underlying distribution of data. After concept drift occurs, data presents a different pattern. The following research questions will be answered:

RESEARCH QUESTION 1 (RQ1): *How to learn the predictor with a training set that may contain more than one patterns when concept drift occurs?*

As it is unknown when concept drift occurs (Fan, 2004), the training set used to train the predictor probably contains several patterns, leading to the situation that the same input value maps to different outputs. If the drift occurs suddenly, a data instance will certainly belong to one pattern (Schlimmer and Granger, 1986a). However, if the drift occurs incrementally, some of the data instances will simultaneously belong to two patterns to some extent (Lu et al., 2018), and it will be inappropriate to use crisp logic to present the belonging relationship between the data instances and patterns. The prediction tasks for data streams will be much complicated if different types of drift occur in the data stream.

RESEARCH QUESTION 2 (RQ2): *How to identify the most relevant data from the training set to update the predictor?*

When a data stream contains concept drift, a drift adaptation method needs to update predictors so that the updated predictor can always present the newest data pattern (Gama et al., 2014). Using "reliable" data to learn a data-driven predictor is critical to guarantee the performance of the predictor (Liu et al., 2017a, 2018a). Once the drift is detected, most informed drift adaptation methods will abandon the previous predictor and learn a new predictor with the newly arrived data instances (Lu et al., 2014). However, even a data instance is lately obtained and recorded in the training set, this instance may present an obsolete pattern. Similarly, old data instances may present the newest pattern.

RESEARCH QUESTION 3 (RQ3): *How to solve the data insufficiency problem when a new concept is identified at its early stage?*

Using drift detection method to identify drift requires at least one full window of new instances (Lu et al., 2014, 2016); therefore, the length of the window determines the detection lag. Identifying a new concept in its early stages means there are few available instances of the new concept (Ditzler et al., 2015). It is difficult to learn a precise predictor for the new concept if there are not enough instances. Reducing the detection lag means reducing the number of instances, and the accuracy of drift detection may suffer. Conversely, increasing the number of instances to a level sufficient for learning accurate predictors will result in greater detection lag. Due to the problem that the data of the new concept is insufficient, drift adaptation methods are faced with a choice between insufficient learning and adaptation delay.

RESEARCH QUESTION 4 (RQ4): *How to deal with concept drift problem for noisy data streams?*

Concept drift problem will be more challenging if problems found in an off-line setting also present in a data stream (Ramírez-Gallego et al., 2017). One of the problems is to learn in a noisy environment (Gomes et al., 2017a). Noise is described as “irrelevant or meaningless data” (Xiong et al., 2006). Specifically, Xiong et al. summarized noise from a task-based aspect as data that significantly hinders the aimed data analysis. Data cleansing or noise removal techniques are designed in an off-line setting to remove the irrelevant noise and enhance the data analysis process (Hernández and Stolfo, 1998). However, this description of noise does not cover the noise in signal processing or time series analysis. Compared to the stationary data, a data stream contains time stamps. Therefore,

each variable in the data stream is a signal or a time series (Cavalcante et al., 2016). In signal processing and time series analysis, the noise is not presented by several data instances. Instead, the noisy signal or series mixes with the designed signal or time series all the time such as background music in voice audio signals and Gaussian noise in time series analysis (Xie et al., 2018).

RESEARCH QUESTION 5 (RQ5): *How to deal with concept drift problem for data streams that have temporal dependency?*

Standard machine learning approaches are built on a static assumption of independent and identically distributed (i.i.d) data. For data stream with concept drift, the distribution is changing over time and therefore breaks the assumption of identical distribution. So far, many concept drift adaptation methods have been designed to overcome the non-identical distribution problem. However, in regression data streams, the target variable is a time series that is probably autocorrelated (Hamilton, 1994), which leads to the temporal dependency problem. How to deal with the problems of temporal dependency and concept drift simultaneously is a big challenge and very few studies could be found that can solve these two problems at the same time.

This research aims to achieve the following objectives, which are expected to answer the above research questions:

RESEARCH OBJECTIVE 1 (RO1): *To develop a drift adaptation method that can identify concepts during the learning process. (aims to answer RQ1)*

For the training set containing more than one patterns, the learning tasks consist of four sub-tasks: 1) recognizing patterns, 2) learning predictors for each pattern, 3) classifying the current instance as a specific pattern, and 4) designing an adaptation method to solve tasks 1)-3) quickly and repeatedly. To solve the

above four tasks, this research introduces fuzzy logic into the learning process and develops a drift adaptation method based on fuzzy clustering process to learn how many patterns exist in the observed data instances, i.e., the training set, and the membership degree of each instance belonging to each pattern during the process of learning the parameters for the predictor.

RESEARCH OBJECTIVE 2 (RO2): *To develop a drift adaptation method that can sequentially select the most relevant data for training.* (aims to answer RQ2)

The informed drift adaptation methods propose to update the predictor only if drift is detected to assure the predictor is updated with the data belonging to the new concept (Lu et al., 2014, 2016). However, existing informed drift adaptation methods need to wait for an entire batch (time window) of data to detect drift and then update the predictor (if drift is detected), which causes adaptation delay (Boracchi et al., 2018; Lu et al., 2018). To overcome the adaptation delay and select the most relevant data to the new pattern, this research proposes a sequentially updated statistic, and based on it develop a drift adaptation method to update the predictor with the most reliable data when every new instance arrives.

RESEARCH OBJECTIVE 3 (RO3): *To develop a drift adaptation method that can generate samples of new concept.* (aims to answer RQ3)

The lack of data or imbalances in the data can be alleviated with data resampling techniques (Wang et al., 2015; Lu et al., 2014). The core idea in these methods is to use one sample for several times. None of these methods, however, are able to generate instances that represent the new concept. Recent research on the generative adversarial nets (GAN) shows great potentials to

generate samples from the same distribution of the target data (Goodfellow et al., 2014; Zheng et al., 2017). A big challenge for GAN is that the generated samples may not improve the accuracy of classifiers because GAN's inputs are essentially noise. To overcome this drawback and generate useful samples to improve the learning accuracy, this research develops a drift adaptation method to generate samples of the new concept through previous data.

RESEARCH OBJECTIVE 4 (RO4): *To develop a drift adaptation method for noisy data.* (aims to answer RQ4)

Drift adaptation for noisy data is a complex problem because concept drift and noise are not always independent in a data stream, and they may have some overlaps sometimes (Gomes et al., 2017a). Simple combination of concept drift methods and noise removal methods is less capable to deal with this kind of data streams (Xu and Wang, 2017; Sun et al., 2018). Several methods for handling concept drift problem for noisy data only discuss the noisy label problem (Schlimmer and Granger, 1986b; Bakker et al., 2009; Lu et al., 2014). However, data streams also contain signal noise because each variable is now a time series. This research develops a drift adaptation method for data streams with signal noise.

RESEARCH OBJECTIVE 5 (RO5): *To develop a drift adaptation framework for data streams with temporal dependency.* (aims to answer RQ5)

The continuous variable in data streams is a time series process that is probably autocorrelated, which leads to the temporal dependency problem. How to deal with the problems of temporal dependency and concept drift simultaneously is a big challenge and very few studies could be found that can solve these two problems at the same time. This research discusses the concept drift problem in

the data stream that have temporal dependency. A drift adaptation framework is developed for data streams with concept drift and temporal dependency in regression cases.

1.3 Research Contributions

This thesis aims to present a comprehensive analysis of concept drift adaptation problems from different aspects and in two new scenarios. The main contributions of this study are concisely summarised as follows:

1) *An expansive concept drift definition.*

An expansive definition of concept drift is proposed to present the characteristics of drift as well as the characteristics of a concept so that the concept drift problem is distinct from stochastic disturbances. In addition, the new definition of concept drift guarantees the validity for the drift adaptation methods in principle.

2) *A new objective function and a novel optimization strategy, and based on them, a new concept drift adaptation method.*

This research designs a new objective function to measure the difference between the estimated output and its true value over different data patterns. A novel optimization strategy is proposed to minimize the objective function to obtain the solution to the problem. Based on the proposed objective function and optimization strategy, a new concept drift adaptation method, named FUZZ-CARE is proposed. FUZZ-CARE is able to precisely learn and update predictors when there are several patterns in the training set.

3) *A new drift measurement and a new sequentially updated statistic, and*

based on them, a new concept drift adaptation method.

This thesis proposes a new drift measurement, segmented symmetric degree (SSD) to measure distributional discrepancy between two concepts. It removes the limitation of a one-sided measurement when two distributions have different variances. Based on SSD, drift-gradient is proposed to sequentially quantify the increase of SSD when every new instance arrives without computing the value of SSD before and after the new instance arrives. Using drift-gradient, a new drift adaptation method, named SEGA is proposed to automatically select the most relevant data for updating the predictor.

4) A new drift detection and sample generation network, and based on the network, a new drift adaptation method.

A new network-structured drift adaptation method, called AFN is proposed. AFN does not rely on the learner error nor a statistic, and thus can overcome the problems of insufficient learning and adaptation delay. In addition, the network structure facilitates embedding into other neural network based methods.

5) A new noise-tolerant drift adaptation method.

This adaptation method, NoA is able to deal with data streams consisting of any continuous variable that has signal noise; There has been no previous study of handling concept drift as well as signal noise.

6) A definition of the data stream with temporal dependency, and a new drift adaptation framework for data streams with temporal dependency.

This research defines the data stream from the aspect of mixed time series. A new definition formally presents the temporal dependency in a data stream. According to the new definition, this research reconstructs the feature space by a time delay embedding. This research also theoretically proves that estimation

error converges faster in a linear model built on the reconstructed space when data streams face concept drift and the temporal dependency problem. In addition, linear cases are generalized to non-linear cases by introducing locally weighted regression. Based on the proposed definition, this research develops a new framework, named DAR, for data streams with concept drift and temporal dependency. The suggested framework not only provides a way of utilizing generalized drift adaptation methods, but also broadens the application of a drift adaptation method in a more realistic scenario.

1.4 Research Significance

The theoretical and practical significance of this research is summarised as follows:

Theoretical significance: This research investigates the natural properties of concept drift and data streams: an expansive definition of concept drift is proposed. The expansive definition supplements the characteristics of a concept. Thus, it guarantees the validity for drift adaptation methods in principle; a formal definition of a data stream with temporal dependency is proposed, which lists key characteristics of a data stream that contains temporal dependency. This facilitates further studies on data streams. The concept drift problem is divided and conquered by a set of drift adaptation problems. More specifically, as the core of the concept drift adaptation is to update predictors, the proposed ideas could enrich the approach for powerful and meaningful manipulations on updating a predictor. This research also contributes to the theoretical analysis of drift detection by density-based statistics by enriching statistical properties of

its variance.

Meanwhile, this research explores the concept drift adaptation for more complex scenarios, such as adaptation under noisy data, and adaptation under temporal dependency. The extended scenarios could not improve the impact of this area, but also motivate its continuing development.

In addition, this research contributes to the field of learning in non-stationary environments by theoretically proving the estimation error converges faster in a linear model built on the reconstructed space by a time delay embedding. Meanwhile, this also extends the application scenario of time series theory.

Practical significance: The findings of this research contribute to the benefit of society given the increasing demand of real-time prediction in modern life. This study develops a set of drift adaptation methods to improve adaptation efficiency and prediction accuracy. The first adaptation method can identify different patterns and the degree that each instance belonging to each pattern, which can also be applied for pattern recognition. The second adaptation method is to pick out the most relevant data for learning predictors, can also be used as a dissimilarity measurement and multivariate two-sample test. The third adaptation generates new samples from the previous samples which can also be used for transfer learning. Meanwhile, the network structure of the third method makes it easy to be embedded into other adaptive neural networks. Meanwhile, two methods are developed in more realistic scenarios of data with noise and data with temporal dependency. The findings help resolve the real-world problems of online decision making. There is the potential of many other online applications that could benefit from this study.

1.5 Thesis Structure

The structure of the thesis is shown in Figure 1.3 and the chapters are organised as follows:

- CHAPTER 2 studies the literature of data stream mining and concept drift adaptation methods, thereby revealing the current research gap. In this chapter, the concept drift problem and basic procedure of concept drift adaptation are introduced. Then, categorization of the existing algorithms based on their implementation details are given. At last, the limitations of the reviewed algorithms are discussed, which inspires the following chapters and solutions.
- CHAPTER 3 proposes a fuzzy clustering based adaptation method, called FUZZ-CARE to embed the clustering loss in a regression loss function. The core idea is to train separate predictors for different patterns and use fuzzy membership matrix to identify the membership degree of each instance belonging to each pattern. This chapter addresses RQ1 to achieve RO1.
- CHAPTER 4 proposes a sequentially update statistic called drift-gradient, and based on drift-gradient, a segment based adaptation method, called SEGA is proposed to identify and select the most relevant data from the training to update the predictor. This chapter addresses RQ2 to achieve RO2.
- CHAPTER 5 proposes a network that can detect drift and generate samples of new concept after drift is detected. Based on that, a new drift adaptation method, called AFN, is proposed to update the predictors with the gener-

ated data, which overcome the data insufficiency when a new concept is identified at its early stage. This chapter addresses RQ3 to achieve RO3.

- CHAPTER 6 analyses the concept drift in a new scenario that data streams also contain signal noise. A noise-tolerant drift adaptation (NoA) method is proposed to deal with concept drift problem for noisy data streams. This chapter addresses RQ4 to achieve RO4.
- CHAPTER 7 analyses the concept drift in a new scenario that data streams also contain temporal dependency. Few theorems have been developed to compare the estimation error converge on the original feature space and that on a reconstructed space by a time delay embedding. Based on the theoretical conclusions, a drift adaptation framework, called DAR is proposed for data streams with concept drift and temporal dependency in regression cases. This chapter addresses RQ5 to achieve RO5.
- CHAPTER 8 summarises the findings of this thesis and points to directions for future work.

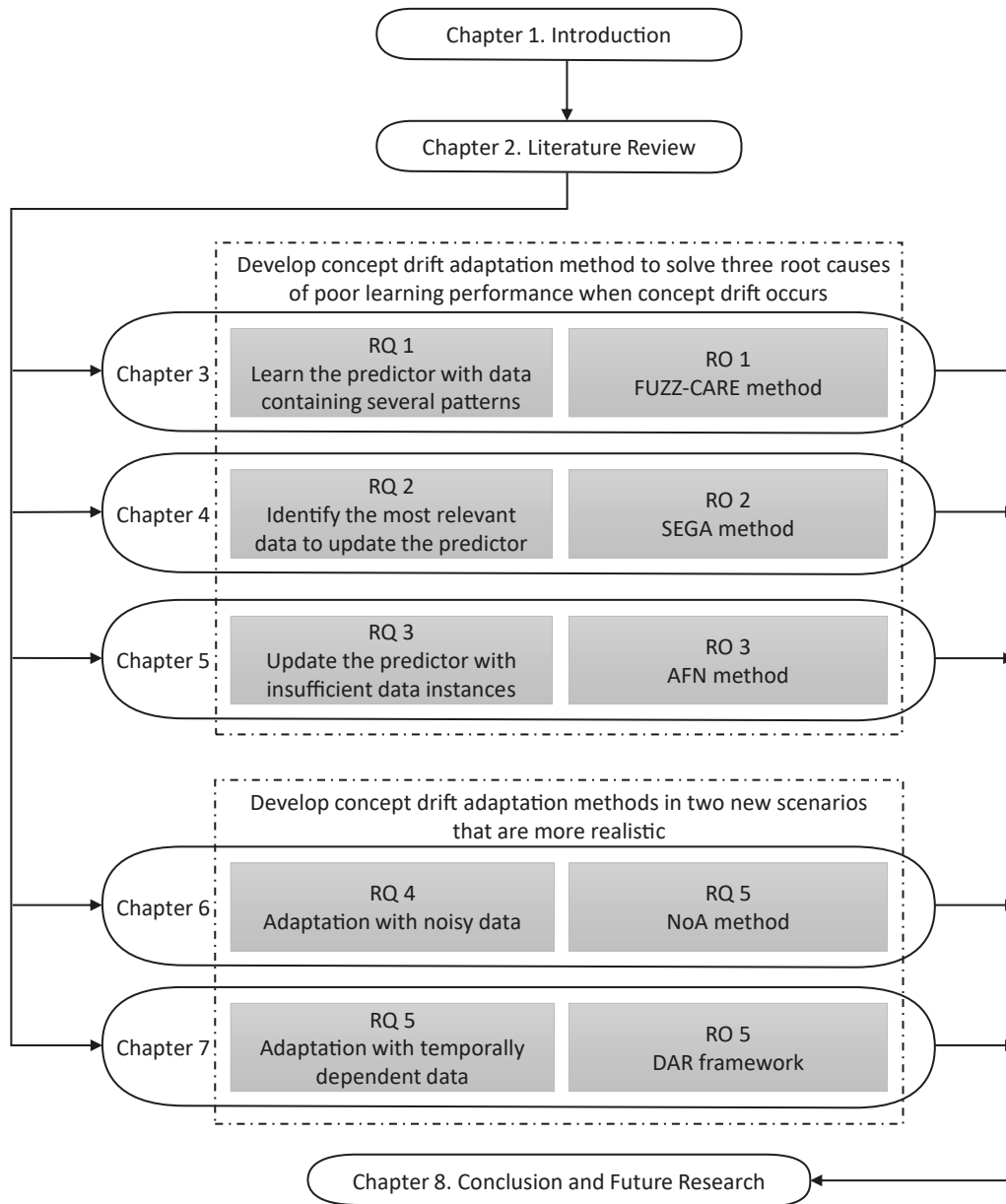


Figure 1.3: Thesis structure

LITERATURE REVIEW

This study focuses on using concept drift adaptation methods to predict labels for real-time data streams. This chapter reviews related research in this area that has recognized the significance of this problem or provided solutions to this problem. Section 2.1 introduces the evolving data stream and real-time prediction in data stream mining area. Then, the concept drift problem is introduced including its definition, types, and applications in Section 2.2. Concept drift adaptation techniques are comprehensively reviewed in Section 2.3.

2.1 Data Stream Mining

This section gives a general review of data stream mining, which is the basis to profoundly understand the concept drift problem.

2.1.1 Evolving data streams

Big data is an outcome of the current information explosion that is relevant to a diverse range of fields in the natural, life, social, and applied science, including physics, biology, medicine, economics and management (Maass et al., 2017). Big data has been widely characterized by the three Vs (Janssen et al., 2017): a hugely increased Volume of data, a Variety of data sources and quality, and the high Velocity at which data is generated or obtained (Elgendy and Elragal, 2014). Big data technology holds incredible promise for improving people's lives, accelerating scientific discovery and innovation, and instigating positive societal change (Drosou et al., 2017). Meanwhile, new challenges accompanying the heterogeneity, incompleteness, scale, timeliness, privacy and process complexity of big data, including aspects of data acquisition, data storage, information extraction, and big data analysis, need to be overcome (Gama, 2012). (Labrinidis and Jagadish, 2012). Further three Vs are now recognized as the development of big data analysis: Veracity, which focuses on the unreliability inherent in data sources; Variability, which refers to variations in data flow rates; and Value, which refers to the issue of low value density (Gandomi and Haider, 2015; Fan and Bifet, 2013; Elgendy and Elragal, 2014).

Recent developments of the Internet of Things brought serious challenges to big data that data arrives in as continuous streams (Losing et al., 2016), namely the data stream which consists of multiple infinite and fast evolving data series (Jaworski et al., 2018; dos Reis et al., 2016; Haque et al., 2016b). Eight challenges of data stream mining were discussed in (Krempl et al., 2014), covering the cycle of knowledge discovery from data. These challenges are summarized from

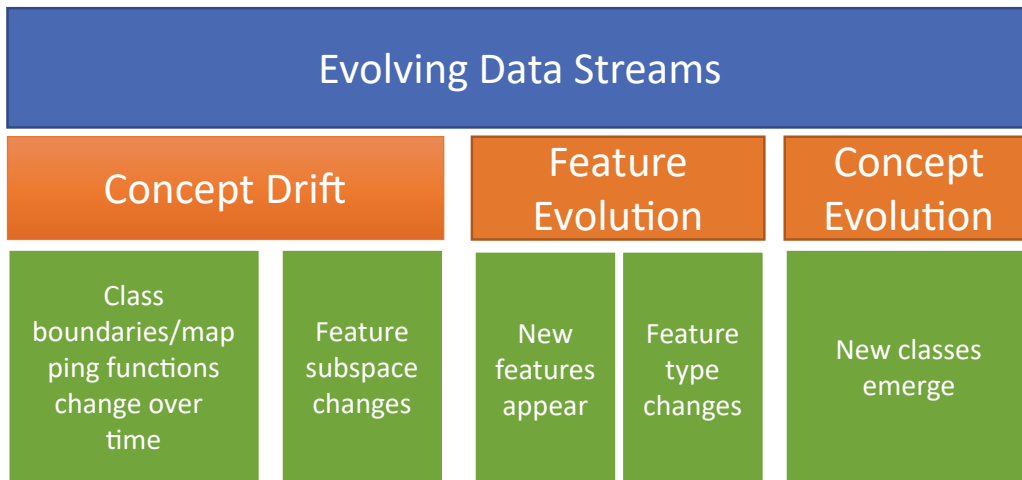


Figure 2.1: Three main challenges in evolving data streams

three aspects (Lu et al., 2019): 1) the development of new data mining skills for data streams; 2) the development of simpler, self-adaptive machine learning algorithms; and 3) the requirements of privacy and confidentiality for gaining trusts of the users and society in the system.

In recent years, data stream mining has been extensively studied in growing fields of multidisciplinary research including data bases, artificial intelligence, machine learning, automated scientific discovery, statistics, decision making and so on (Gurjar and Chhabria, 2015). The main challenges in learning the evolving data stream have been categorized to three kinds (Figure 2.1) (Parker and Khan, 2015): 1) concept drift, refers to class boundaries change over time or the distribution of a feature changes (Zhao et al., 2018). For example, in dressing recommender systems, the model is trained to recognize whether the dressing in a given photo is fashion or not. The characteristics of fashion change fast, and therefore the classification boundaries between “fashion” and “not fashion” changes; 2) feature evolution, which denotes that new features appear or feature

type changes (Hou et al., 2017). For example, the assessment model may have new features when the bank credit assessment system updates to include more details of the customers; 3) concept evolution, represents the situation that novel classes emerge (Zhu et al., 2018). For example, new topics appear and old topics disappear or reappear in the bibliometric field, such as the novel topic of "block chain" in 2013 and the recurrent novel class of "neural network". Let d_τ be a multivariate random variable defined on $\mathcal{X}_\tau \times \mathcal{Y}_\tau$ with respect to $p_\tau \in \mathcal{P}(\mathcal{X}_\tau \times \mathcal{Y}_\tau)$, where $\tau \in \mathbb{Z}^+$, $\mathcal{X}_\tau \subseteq \mathbb{R}^{s_\tau}$ is a topological space, $\mathcal{Y}_\tau = \{1, \dots, C_\tau\}$ is a label space and $\mathcal{P}(\mathcal{X}_\tau \times \mathcal{Y}_\tau)$ consists of all Borel probability measures on $\mathcal{X}_\tau \times \mathcal{Y}_\tau$. A data stream is a number of observations $\mathbf{D}_\tau = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\tau, y_\tau)\}$ from p_1, \dots, p_τ , and the three kinds of changes in a data stream is defined as follows:

- **Concept drift:** $\exists \tau_0 : p_{\tau_0} \neq p_{\tau_0+1}$.
- **Feature evolution:** $\exists \tau_0 : \mathcal{X}_{\tau_0} \neq \mathcal{X}_{\tau_0+1}$.
- **Concept evolution:** $\exists \tau_0 : \mathcal{Y}_{\tau_0} \neq \mathcal{Y}_{\tau_0+1}$.

2.1.2 Real-time prediction

As data evolves over time, the validity and reliability of the historical data are questionable. Data stream mining has to consider these issues to perform accurate, up-to-date, real-time analysis. For example, the detection of highway flooding (Puthal et al., 2017). Real-time prediction is one of the most important applications of data stream mining (Gaber et al., 2005). Real-time prediction is to make prediction in real time (Ikonovska et al., 2015). It is clearly different from the prediction in a stationary setting because each instance in a data stream is first used to test the learned predictor, and then to train/update the predictor.

Real-time prediction for a non-stationary data stream arises in many scenarios, such as online transactions in the financial market, weather forecasting, air quality prediction and so on (Pratama et al., 2017; Li et al., 2018). For example, 20 years ago, TV was the main source of weather forecast information for most people. Additionally, weather forecasts would simply indicate the weather for tomorrow or the day after tomorrow at most. Today, people expect hourly weather reports and weather forecasts for a week in advance. Historically, a numerical weather prediction model was used to compute long-term weather variables. However, such models are not flexible enough to make hourly weather predictions. Nor can they be applied to a short-term, high-frequency online forecast system, as it takes a significant amount of time for the models to compute the necessary partial differential equations.

Conventional machine learning methods are not applicable to make real-time prediction for data streams. For a data stream, future data may exhibit different patterns from those of the previous data used to learn the predictor. Therefore, the prediction accuracy of the learner predictor is deteriorated due to this evolving nature of data streams. Although data stream mining has become important research topics during the last decade, a truly autonomous, self-maintaining, adaptive data mining system is still lacking (Krempel et al., 2014). The short lifespan of data restricts us from storing and accessing all historical data during each processing cycle; however, processing accuracy has been strictly limited by the fact that the data can be accessed only once (one-pass setting) (Aggarwal et al., 2003).

2.2 A General Introduction of Concept Drift

Concept drift is a particularly important factor in data stream mining. In this section, the concept drift problem and corresponding state-of-the-art solutions will be reviewed.

2.2.1 Problem description

Concept drift refers to the phenomenon of the changing data distribution (Gama et al., 2014). The problem of concept drift is discussed in a data stream that the newly arrived data instances may exhibit a different pattern from the previous data (Wang et al., 2018). Standard machine learning approaches are built on a static assumption of independent and identically distributed (i.i.d) data and therefore are not suitable for learning data streams once the data distribution has experienced unpredictable changes (Haque et al., 2016b; Gama et al., 2013).

A widely accepted definition of concept drift is $p_{t+1}(\mathbf{X}, y) \neq p_t(\mathbf{X}, y)$ (Gama et al., 2014). According to property of probability, concept drift can be decomposed into two parts whereby $p(\mathbf{X}, y) = p(y|\mathbf{X}) \times p(\mathbf{X})$. *Real drift* refers to the changes in the posterior probabilities, i.e., changes in $p(y|\mathbf{X})$, and *virtual drift* denotes the changes in $p(\mathbf{X})$ (Žliobaitė, 2010a). Existing research mainly focuses on the real concept drift problem. (Wang et al., 2018) studied concept drift on three decomposed parts from an aspect of class imbalance which included changes in prior probability $p(y)$, class-conditional *pdf*, i.e., $p(\mathbf{X}|y)$, and the posterior probability.

However, limited research explains the term “concept”. The only study on this problem is by (Webb et al., 2016) that a concept is formally defined as

a many-to-one mapping function $X \mapsto y$. Many machine learning algorithms require a many-to-many mapping X to y , which leads to the preferred definition of a concept as the distribution of a classification task, i.e., $Concept = P(X, y)$. Generally, a concept can be considered as a data pattern.

Concept drift significantly hinders off-line predictive performance (Harries et al., 1998). Unlike noise series, concept drift is influenced by hidden changes in the context (Kubat, 1992; Schlimmer and Granger, 1986a,b). The influence of hidden context changes results in biased estimation of the target output (Widmer and Kubat, 1996), and the error can be depicted by hidden context. For example, in rain forecast, the prediction of rainfalls based on atmospheric factors may change radically in different periods. Studies on concept drift attempt, as far as possible, to learn data without these hidden variables, because hidden variables are difficult to observe and in many situations are unknown based on the current knowledge. For example, cellphone usage was limited to communication about ten years ago, but now the usage has been changed to include taking photos, searching the Internet, conducting business, learning online and more. Customer cellphone usage prediction must consequently relinquish their models from the past. If a cellphone company continues to use an old usage pattern, it will possibly fail. In this example, one hidden context is the development of cellphone techniques. This development certainly exists but is difficult to quantify.

Online prediction tasks become particularly difficult if real concept drift occurs, i.e., the changes in $P(y|\mathbf{X})$ (Yeon et al., 2010). The feature variables can be immediately observed and easily collected. They are used as the input of the predictor to estimate the target output, which is difficult to collect. The virtual drift (i.e., $P(x)$ changes) can be omitted if \mathbf{X} is independent of the model

parameters θ (Song et al., 2019a). One example is a weather forecast of “rain/no rain”. If today is rainy, it will probably rain tomorrow in the wet season, but will not rain in the dry season given the same condition. Clearly, applying a rain forecast model trained in the wet season to predict dry season weather will result in significant errors.

2.2.2 Types of concept drift

The occurrence of concept drift can be divided into four types (Figure 2.2): *sudden drift*, *incremental drift*, *gradual drift* and *reoccurring drift*. Lu et al. have reviewed the peculiarities of these four types of drift (Lu et al., 2018). Sudden drift refers to the case that one pattern suddenly switches to another. For example, Kate is reading the news. A sudden interest in meat prices in New Zealand when she got an assignment to write an article, is a sudden drift (Žliobaitė, 2010b). Incremental drift consists of many intermediate patterns between two patterns when one is changed to another. For example, a sensor slowly wears off and becomes less accurate (Gama et al., 2014). Gradual drift means the new pattern goes back to the previous one sometimes, but with decreasing frequency, such as the change in walking pattern for toddlers (Abdallah et al., 2018). Reoccurring concept means that previously seen patterns that are seen again, e.g., most economics, transportation and weather data present this characteristic (Lu et al., 2018). It should be clarified that the incremental drift and gradual drift are different. Incremental drift occurs when data instances gradually change their values over time, and gradual drift occurs when the change in data instances includes the class distribution of previous data (Wadewale and Desai, 2015).

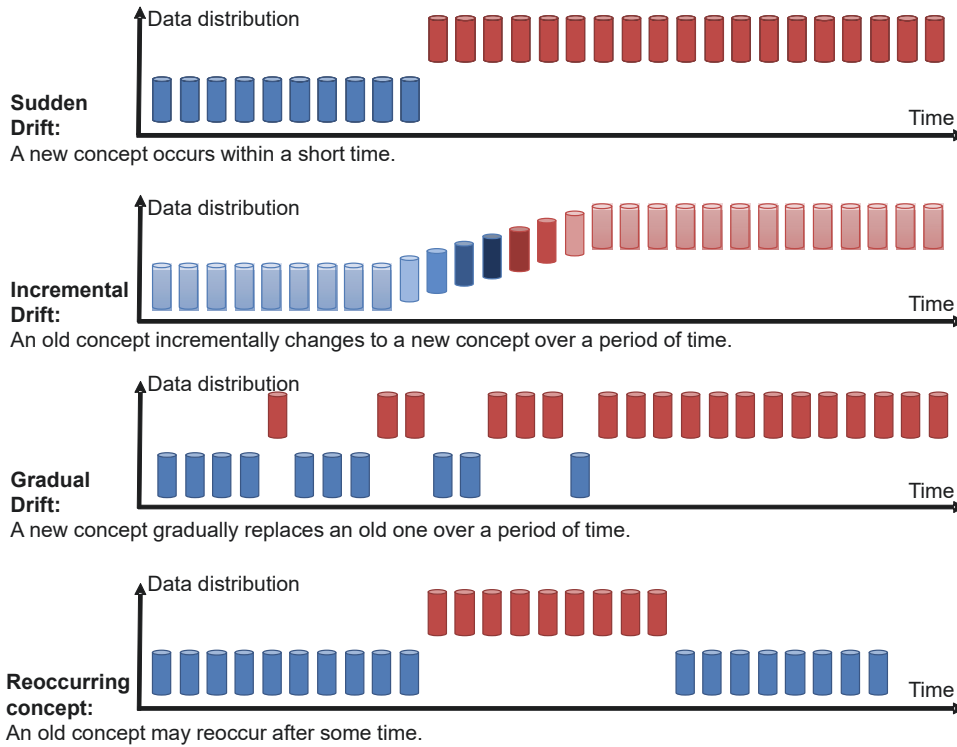


Figure 2.2: Four types of concept drift.

Existing concept drift-related studies handle concept drift problem in a general way by assuming the existence of sudden non-reoccurring drift, regardless of the type of drift. Especially, (Song et al., 2019a) summarizes four types of drift into two categories: 1) *permanent drift* includes sudden or gradual drift that once drift occurs the concept will change to another concept and never turn back (e.g., replacement of sensors); 2) *alternate drift* includes gradual drift and reoccurring concept that the concept switches between several alternate patterns.

2.2.3 Concept drift applications

Handling concept drift is particularly important in real world practice because stream data are ubiquitous in many applications. Examples include network

traffic, telecommunications, and financial transactions, to name just three. Data mining tasks in these systems will inevitably encounter the concept drift problem. In some cases, the ability to handle concept drift becomes the key factor in improving system performance. A comprehensive review of concept drift industrial applications can be found in (Žliobaitė et al., 2016; Žliobaitė, 2010b), in which the authors list many industrial examples of different types of application, including monitoring and control, information management, analytics and diagnostics.

Concept drift detection applications. Drift detection applications fulfill the industrial requirement of diagnosing significant changes in the internal and external environment of industry trends or customer preferences: for example, using drift detection technology to diagnose changes in user preferences on news (Harel et al., 2014). Similar tasks include fraud detection in finance, intrusion detection in computer security, mobile masquerade detection in telecommunications, topic changes in information document organization, and clinical studies in the biomedical area.

Concept drift adaptation applications. Concept drift adaptation applications concern the maintenance of continuously effective evaluation and prediction system for industry. These applications sometimes also involve drift detection technologies for better accuracy. A real case represented in (Sousa et al., 2016) is the design of a credit risk assessment framework for dynamic credit scoring. Other real-world drift adaptation applications can be found in customer churn prediction in telecommunication, traffic management in transportation, production and service monitoring, recommendations for customers, and bankruptcy prediction in finance.

With the rapid development of technology, data streams are becoming more

highly dimensional with larger sizes and faster speeds. The new challenges presented by big data streams require more advanced concept drift applications. One concern is how to handle concept drift problems in the Internet of Things (IoT) (De Francisci Morales et al., 2016), where the huge quantity of big data streams require deeper insight and a better understanding of concept drift.

2.3 Concept Drift Adaptation

Concept drift adaptation aims to design an effective strategy to update the learned predictors by time, to make sure that the predictor can be self-adaptive to the data that may change over time Gama et al. (2014). There are three basic requirements in drift adaptation: 1) **Adaptation should be fast** Bifet et al. (2017). As the data arrives as a stream, the adapted predictor should be updated and used to predict the label value of future instances before their true value is available. For example, for a drift adaptation method, the adaptation process should be done in half an hour if the frequency of data is half an hour. 2) **Adaptation should be robust** dos Reis et al. (2016). As the data stream is assumed to have infinite upcoming data instances, the adaptation strategy is required to have no accumulation of error. 3) **Adaptation should also be applicable in a non-drift data stream** Song et al. (2019a). Although concept drift is a characteristic of the data stream, it is not necessary for every data stream to contain drift. Besides, it is uncertain whether drift would occur in a real-world data stream. Therefore, the designed drift adaptation methods should also be applied in a non-drifted data stream.

2.3.1 Classification of concept drift adaptation methods

Concept drift adaptation methods have three learning components (Figure 2.3): 1) learning mode; 2) adaptation strategy; 3) predictor management (Gama et al., 2014). These components determine the classification of adaptation methods from the aspect of each component.

Learning mode refers to update the predictor by retraining or incremental learning (such as, retuning) when new data points are available. Retraining approaches need some data buffer to be stored in memory (Tsymbol, 2004). Once the new predictor is learned, the old predictor will be immediately discarded (Street and Kim, 2001; Zeira et al., 2004). To retrain the predictor more accurately and efficiently, windowing techniques have been broadly studied in this area window. The window size can be different according to the characteristics of the data. A typical example is ADWIN Bifet et al. (2007). The window grows by including useful data examples when no drift occurs and decreases by excluding old data examples when drift is identified. Similar techniques can be found in Tennant et al. (2017) where the model is only modified when the learning error reaches the error threshold, and this happens irregularly. A series of algorithms have been designed to choose the appropriate window (Alippi et al., 2011, 2012, 2013; Alippi and Roveri, 2008a,b). Incremental learning update the current predictor using the most recent data (Pratama et al., 2016), such as IKS dos Reis et al. (2016) and Learn⁺⁺ algorithms Elwell and Polikar (2011); Ditzler and Polikar (2013). Especially, an online learning mode updates the predictor by the most recent instance (Littlestone, 1988; Frías-Blanco et al., 2016).

Existing drift adaptation is implemented by two strategies: blind adaptation

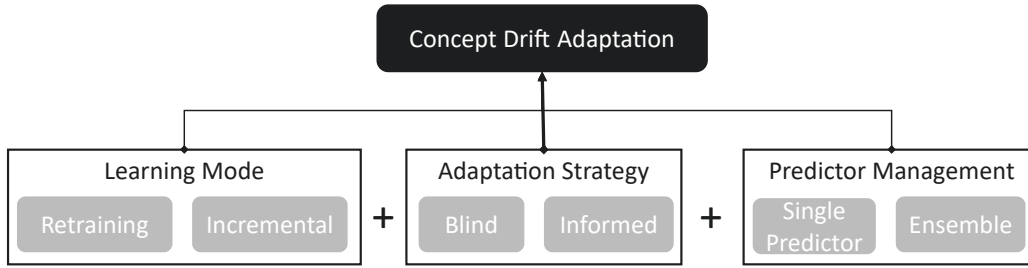


Figure 2.3: Learning components and their classification in concept drift adaptation methods.

strategy and informed adaptation strategy Gama et al. (2014). The adaptation method with a blind strategy is also categorised as “passive” approach and the adaptation method with an informed strategy categorised as “active” approach Ditzler et al. (2015). Blind adaptation strategies periodically adapt the predictor when every fixed-size sliding window of data arrives, no matter whether the drift truly occurs, or not Minku and Yao (2012). Most ensemble learning methods designed for data streams use a blind adaptation strategy, which has been extensively reviewed in Gomes et al. (2017a). In contrast, the informed adaptation strategies adapt the predictor based on drift detection results Lu et al. (2016).

Predictor management refers to the techniques for maintaining active predictors. Early research in this area normally applies a single predictor Lu et al. (2014, 2016). In recent studies, ensemble predictors have been widely used for its good performance and wide applications on different tasks (Mendes-Moreira et al., 2012; Gomes et al., 2017a).

In summary, the concept drift adaptation is a technique that continuously updates (retrains or incrementally retunes) the predictor (a single predictor or an ensemble predictor) by a blind or an informed strategy. In this research, the drift adaptation methods are reviewed by the adaptation strategies.

2.3.2 Blind concept drift adaptation

Concept drift has not been widely recognized in most earlier studies and there is a shortage of drift detection techniques, but there are nevertheless a few methods capable of solving the concept drift problem. One such attempt is the IB3 algorithm designed in Aha et al. (1991). IB3 monitors each instance during a time interval and employs a significance test to determine which instances should be accepted, dropped or monitored in future training. Similar methods can be found in Maloof and Michalski (2004). Blind adaptation strategies do not involve drift detection techniques, such as evolving neural networks in Pratama et al. (2017), AUE2 Brzezinski and Stefanowski (2014), DWMIL Lu et al. (2017) and DTEL Sun et al. (2018). An online blind adaptation method is proposed which can update predictors based on selected instances that were most relevant to the newly-arrived data Song et al. (2017). Paper Žliobaitė et al. (2015) suggests using a temporally augmented classifier which incorporates a higher order temporal dependency for data stream classifications.

Many ensemble learning techniques in the concept drift area use a blind adaptation strategy Gomes et al. (2017a); Krawczyk et al. (2017), such as OOB and UOB Wang et al. (2015). Early ensemble algorithms such as SEA (streaming ensemble algorithm) and AWE (accuracy updated ensemble) apply a simple updating strategy in which a classifier trained with a newly-arrived data chunk is added to the ensemble, and the weakest of the previous classifiers are removed based on their performance on the new data chunk Street and Kim (2001); Wang et al. (2003). A similar ensemble mechanism can be found in CDC, DWM and AddExp algorithms Stanley (2003); Kolter and Maloof (2007, 2005). Brzezinski

et al. extended AWE and proposed AUE1 and AUE2 algorithms by updating classifiers according to learning accuracy rather than simply selecting them into the ensemble Brzezinski and Stefanowski (2014). Ditzler et al. Elwell and Polikar (2011); Ditzler and Polikar (2013) proposed a series of ensemble algorithms: Learn++.NSE, Learn++.CDS and Learn++.NIE. The Learn++.NSE algorithm assumes that no previous data are available when the new batch of data arrives and that any past information must have been reflected by the previously generated classifiers. The ensemble combines dynamically weighted classifiers that give more weight to classifiers capable of identifying previously unknown instances and penalizes classifiers that misclassify. The weight is based on sigmoidal averaged time-adjusted error over all environments Elwell and Polikar (2011). Learn++.CDS and Learn++.NIE are the versions that have been developed for imbalanced data Ditzler and Polikar (2013).

When no drift detection techniques are involved, both instance-based adaptation methods (methods that update or weight the most useful instances) Oza (2005) and classifier-based adaption methods (methods that add, delete or weight different classifiers) (Gomes and Enembreck, 2013, 2014) assume that drift occurs constantly. They do not analyze the drift but keep updating the model even though no drift occurs. It is true that these models are suitable for the newly arrived data; however, it cannot be confirmed that they are suitable for learning the new patterns, because exactly when a new pattern occurs is unknown in these methods. The predictor could be ill-trained with the “always-adapt” mechanism in the blind strategy. It has been validated by Song et al. (2019a) that a blind online adaptation can lead to large errors if concepts reoccur. Despite the drawbacks, these methods have been successfully applied in real data streams.

2.3.3 Drift detection

Concept drift detection is to detect the appearance of drift Alippi et al. (2017). Two commonly used detection methods are learner error-based and statistic-based detection Baena-Garcia et al. (2006); Sakamoto et al. (2015). The learner error-based drift detection monitors the learning error made by the current predictor, and retrain the model if the error is beyond a threshold Haque et al. (2016a). Using learning-error based criteria to detect whether the existing model is still suitable for an incoming instance. If the learning error is not convergently decreasing as it should in a stationary case, a drift is recognized. To distinguish actual drift from random error, a Hoeffding bound is usually introduced to restrict the false alarm (i.e., the ratio of falsely classified positive cases) Lughofer et al. (2016). For example, the decision tree models use learner error and Hoeffding bound to decide the split criterion Bifet et al. (2017). In Rutkowski et al. (2015), the authors provide a different mathematical justification for Hoeffding bound used in the decision tree. A P-tree is proposed in Shao et al. (2014) to use error-driven representativeness to identify prototypes. The statistic-based drift detection introduces or proposes a statistic and uses the statistical property of this statistic to infer drift information Ditzler et al. (2015), such as the Kolmogorov-Smirnov test dos Reis et al. (2016), density-difference estimation Bu et al. (2018), the likelihood ratio Alippi et al. (2017) and regional density Liu et al. (2018a).

Detecting drift by data distribution commonly relies on constructing statistics to measure the distance between two batches of the data stream and uses the distribution of these statistics to estimate the critical region by hypothesis testing

(Lu, 2012). Concept drift is defined as a change in distribution from one time point to another time point. It is theoretically reasonable that the instance at time point t is a sample selected from a specific population and that the instance at time point t' is from another population. However, when the sample information is used to infer whether there is a distribution difference between these two populations, the smallest sample size is required Lu et al. (2014, 2016). Two methods are usually used for the distribution of the statistics. One is to construct an appropriate statistic and give its asymptotic distribution by the central limit theorem Frías-Blanco et al. (2015); Zambon et al. (2018). The other is to generate the distribution of the statistics by bootstrapping or the permutation test Dasu and Krishnan (2006); Gu et al. (2016); Harel et al. (2014).

There are also drift detection methods that do not need to rely on the learner error or statistics. For example, detecting drift by an adversarial net (Song et al., 2018); and using parameter changes to detect drift (Su et al., 2008).

2.3.4 Informed concept drift adaptation

Informed drift adaptation evolves a drift detection process, and use the drift information provided by the detection method to help adaptation (Boracchi et al., 2018). The informed drift adaptation methods are reviewed and summarized in Lu et al. (2018).

There are generally two ways to realize concept drift adaptation based on drift detection: 1) one is to use a drift detection technique to output a binary variable of “drift is detected” or “drift is not detected”, and only update the predictor when the drift is detected Bu et al. (2017). For example, Gama et al. (2004) proposed

observing the learning error and detecting drift using a warning level. Once a change is detected at a specific time point, a new decision model is induced based on instances after this time point. SAND was proposed to adapt to a new concept by detecting and storing outliers instead of using a drift detection method Haque et al. (2016a,b). The principle of SAND is to use the existing ensemble to predict normal points and temporarily store every detected outlier. Once the length of stored outliers is sufficiently large, a novel class detection module is invoked. If a novel class is detected, a new model is trained which replaces the oldest one in the ensemble. Minku and Yao introduced a drift detection method to determine the appropriate diversity of an ensemble Minku and Yao (2012) based on a diversity analysis in the presence of different drift types, which they presented in Minku et al. (2010). In tree models, many drift detection methods are introduced to handle concept drift Yang and Fong (2015); Bifet et al. (2017); Rutkowski et al. (2015); Hulten et al. (2001).

2) The other method of adaptation requires drift detection techniques to recognize drifted parts and make predictions based on that recognition (drift understanding) Lu et al. (2018). For example, the ADWIN algorithm proposed in Bifet et al. (2007) uses an adaptive windowing method in which the window size increases for greater accuracy before the drift occurs and shrinks automatically when drift is detected. A drift detection, localization and characterization method was proposed in Bose et al. (2011) which can detect drift as well as its localization by introducing the univariate two sample Kolmogorov-Smirnov test (KS test) and Mann-Whitney U test (MW test). This method is suitable for detecting significant differences between two overlapping or non-overlapping windows. Josep and Rochard introduced abstract interpretation theory to learn a polyhedron

consisting of a set of linear inequality constraints Carmona and Gavaldà (2012). Concept drift is detected based on the polyhedron in process mining. Josep and Rochard applied a drift detection method similar to ADWIN to generate a series of inequalities, and a sub-set of these inequalities was used to locate drift. Ning et al. Lu et al. (2016, 2014) proposed a drift detection method via the permutation test, which picks out the drift instances. The classifier is only learned according to the non-drift instances. Similar methods are proposed in (Liu et al., 2017a; Gu et al., 2016)

Detection-based adaptation is often time delayed, because an entire batch of new instances is needed to conduct the detection process Liu et al. (2018a). Most of these methods also replace the current learner with a new one after a drift has been detected, so they suffer inaccuracies in the process of both detection and adaptation. Sometimes, the drift detection is time-consuming because it needs to deal with the whole window of instances Lu et al. (2014).

2.3.5 Concept drift adaptation methods in regression cases

Existing drift adaptation mostly handles the classification tasks but limited research can be found for regression cases. As pointed out in Mendes-Moreira et al. (2012), successful classification techniques are rarely directly applicable to regression. For example, when the boosting algorithm was originally designed, it could not be directly applicable to regression because it assumes that the generalization error is in $[0.5, 1]$ Mendes-Moreira et al. (2012). This section specially reviews the concept drift adaptations in regression cases.

The issue of regression concept drift was formally emphasized by Ikonomovska et al. (2011a) as regression in time-changing data streams. To solve this problem, the authors proposed an algorithm called very fast incremental model trees with drift detection (FIMT-DD) to perform explicit change detection and informed adaptation. They introduced a deviation-based test called the Page-Hinckley (PH) test and a threshold to continuously detect whether the current data batch triggers an alarm to indicate a change in the distribution. They also introduced a Q statistic to reduce the false alarm of this drift detection. Ikonomovska et al. developed FIMT-DD to an online regression/model tree with options (ORTO) by adding options nodes to assist the split of equally discriminative attributes Ikonomovska et al. (2011b). A generic framework, Concept Neurons, was proposed in Moreira-Matias et al. (2016) which aims to handle drift in regression problems; the Concept stage is designed to monitor whether drift occurs, and the Neuron stage is designed to update the model using a residual-based version of the parameters inverse gradient. Duarte et al. fused the Page-Hinkley test in decision rules and proposed adaptive model rules (AMRules) to detect and react to regression concept drift by pruning the rule set Duarte et al. (2016).

A common feature of the above methods for addressing regression concept drift is that they have an embedded drift detection mechanism. There are also methods of handling regression concept drift that does not have a detection technique. For example, Wang et al. proposed using a constrained penalized regression combiner to track concept drift Wang et al. (2017). The main idea of their method is similar to ensemble learning, where the weights of predictors are determined by minimizing the proposed constrained penalized cost function. Although the authors defined a measure of concept drift, this measure is simply

used to measure the degree of drift in the data and does not assist future predictions. Other ensemble methods for handling regression concept drift are an online weighted ensemble (OWE) Soares and Araújo (2015) and a simple adaptive batch local ensemble Bakirov et al. (2015). Another recent drift adaptation algorithm that focuses on the regression problem is FP-ELM, which assumes that earlier data are always less related to the new concept than newly-arrived data. This method introduces dynamic forgetting parameters to gradually reduce the weight of the training samples in the non-stationary environment. The forgetting parameters are controlled by two user-specified parameters and the current level of error Liu et al. (2016).

2.3.6 Drift adaptation using fuzzy techniques

Fuzzy logic is able to describe a vague definition that has intrinsic advantages in describing the concept drift problem to some extent. For example, given two patterns existing in one batch of data instances in which one follows $y = x^2$ and the other $y = -x^2$, it is ambiguous to classify the point (0,0) to any pattern. Fuzzy logic is a perfect solution for expressing this type of uncertainty. In Pratama et al. (2017), an evolving recurrent fuzzy neural network is proposed to incrementally adapt to concept drift. A generalized interval type-2 fuzzy rule is used in their network architecture which can be automatically generated, pruned, merged and recalled in the single pass learning model. In Song et al. (2017, 2019a), a fuzzy c-means clustering technique is applied during learning the regression model to identify the most relevant data instances to the latest pattern in the training set. Fuzzy methods have advantages for designing windowing techniques for concept

drift adaptation. For example, a fuzzy windowing technique is proposed in Liu et al. (2017b) to detect drift more flexibly.

2.3.7 Drift adaptation in a noisy environment

In the batch-learning setting, noise is described as “irrelevant or meaningless data” (Xiong et al., 2006). Specifically, Xiong et al., summarized noise from a task-based aspect as data that significantly hinder the aimed data analysis, for example, the noise caused by an imperfect data collection process and ordinary data objects that are irrelevant or only weakly relevant to the aimed data analysis (Xiong et al., 2006). Data cleansing or noise removal techniques are designed in a batch-learning setting to remove the irrelevant noise and enhance data analysis process (Hernández and Stolfo, 1998).

Concept drift methods especially drift detection techniques sometimes have overlap with the techniques used to deal with noise such as anomaly detection (Chandola et al., 2009). This is reasonable, as concept drift detection can be considered as anomaly detection repeatedly over time. However, it is important to distinct these two ideas from each other especially these two problems occur in a data stream at the same time (Gomes et al., 2017a).

There are some studies in the literature tested the proposed drift adaptation method in a noisy environment, such as (Harries et al., 1998; Hulten et al., 2001; Liu et al., 2016; Xu and Wang, 2017; Sun et al., 2018) but all of these methods mainly aim to specifically solve the concept drift problem, and the noise problem is just mentioned in the experiments where the tested data is added with the noise at some levels. Limited methods in the literature are designed to handle

concept drift problem in a noisy environment.

The earliest formal solution of concept drift adaptation in a noisy environment is STAGGER Schlimmer and Granger (1986b), where the clustering result is predicted by the description of dually weighted Boolean functions. STAGGER continuously monitors and updates the numerical weights of characterizations of a feature in order to chase the newest data pattern. Rather than deleting noise, STAGGER updates weights of features to present the extent to which particular combinations of features indicate a specific category because it assumes that description will not be a perfect predictor of a category in noisy data. STAGGER has shown good performance on systematic noise and fair performance on random noise. A drawback of STAGGER is that it handles the clustering task, and this “weights against noise” strategy is not applicable in a regression task because the extent to which a description indicates categories no longer exists in a regression task. In Bakker et al. (2009), the authors studied the effectiveness of three change detection methods on identifying outliers and drift—a nonparametric change detection, a parametric test and an ADWIN (refer to Bifet et al. (2007)) method. Drift is dealt in the same way to outlier noise rather than being distinguished from noise. Therefore, this method is not able to unambiguously distinguish outliers from drift in principle. In addition, they focused on a special case of online mass flow prediction in circulating fluidized bed (CFB) boilers, and therefore it remains unknown that whether the proposed method is also suitable to other practical cases. Lu et al. developed a noise-enhanced fast context switch (NEFCS) algorithm to exclude the noise from the case base and effectively update the case base when drift occurs Lu et al. (2016). To the best of our knowledge, this is the only research that introduced professional denoising techniques in the

drift adaptation process to solve the concept drift problem in a noisy data stream. In NEFCS, an anomalous instance is identified as noise only if it is detected as noise by a blame-based noise reduction rule and locates out of concept drift competence areas. In this way, NEFCS is able to distinguish noise from novel concepts.

All the above-mentioned methods discuss the discrete noisy label problem. If the label is a continuous label or noise exists in the feature space, these methods will be invalid. As far as we know, the only research of handling noise on the continuous label and feature space is proposed in Song et al. (2019b), which considers data stream as a collection of time series and discuss the concept drift problem when those time series have signal noise.

DRIFT ADAPTATION BY IDENTIFYING CONCEPTS BY FUZZY CLUSTERING PROCESS

As it is unknown when concept drift occurs, the training set used to train the predictor probably contains several patterns, leading to the situation that the same input value maps to different outputs. This chapter presents a drift adaptation method to deal with that situation. This chapter begins with an introduction of the problem and tasks (Section 3.1). We list the definitions and notations in Section 3.2. The solution is explained in Section 3.3 with its experimental evaluations in Section 3.4. Last, a summary concludes this chapter with discussions about potential future research (Section 3.5).

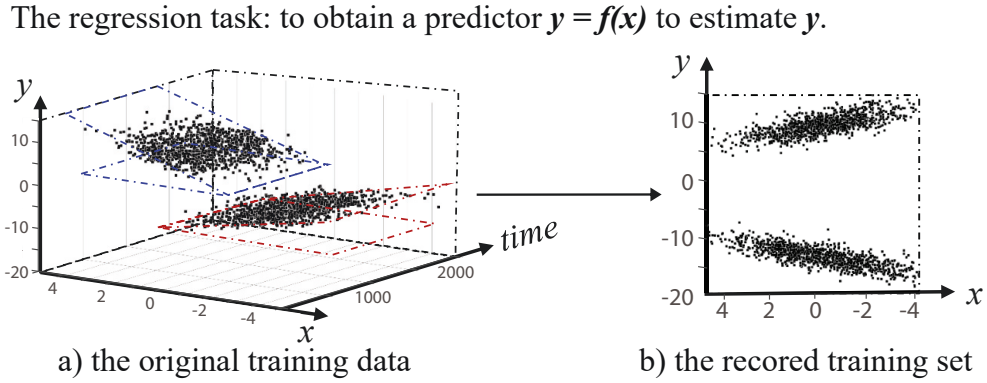


Figure 3.1: The training set for a data stream.

3.1 Introduction

Once a concept drift occurs in a data stream, the data instances used to learn the predictor will probably contain more than one patterns, such as the case presented in Figure 3.1. Figure 3.1 shows the training data for a **regression task** that given the value of x , a predictor—a linear or non-linear function, is learned to estimate the corresponding value of y .

In Figure 3.1 a), each dot presents a data instance of a data stream. The input x and the output y have linear patterns. The pattern is changed from the blue one to the red one after drift occurred at the 1001th time step. However, the length of the training data is set to contain 2000 instances. In addition, the time stamp will be lost after these data instances are recorded in the training set, as is shown in Figure 3.1 b). Therefore, there two different concepts in the training data simultaneously, and it is unknown which concept each instance belongs to.

For the training set in Figure 3.1 b), the same value of x have two corresponding y values. If we directly learn a predictor using this training set, for example a linear predictor by the ordinary least squares(OLS) regression method, the

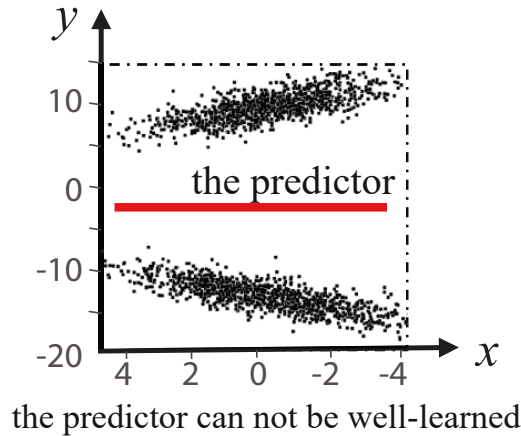


Figure 3.2: Learn a predictor by the OLS method using the training set that contains two patterns.

learned predictor can not well present any pattern as is shown in Figure 3.2.

If the drift occurs suddenly, a data instance will certainly belong to one pattern. However, if the drift occurs incrementally, some of the data instances will simultaneously belong to two patterns to some extent, and it will be inappropriate to use crisp logic to present the belonging relationship between the data instances and patterns. The prediction for data streams with mixed drift problem is much more complicated than the task that considers only one type of drift. For data streams with mixed drift, the regression tasks consist of four sub-tasks: 1) recognizing patterns, 2) learning predictors for each pattern, 3) classifying the current instance as a specific pattern, and 4) designing an adaptation method to solve tasks 1)-3) quickly and repeatedly.

To solve the above four tasks, fuzzy logic is introduced into the learning process. A fuzzy clustering-based adaptive regression approach called FUZZ-CARE is proposed in this chapter especially for prediction tasks in data streams with mixed drift. A fuzzy clustering process is embedded in FUZZ-CARE by adding

the objective function of fuzzy clustering to the objective function of learning the regression predictor. This design enables it to learn how many patterns exist in the observed data instances, i.e., the training set, and the membership degree of each instance belonging to each pattern during the process of learning the parameters for the predictor. The proposed FUZZ-CARE is therefore able to make accurate predictions for data streams with concept drift in which the training set probably contains several patterns. This solves tasks 1) and 2). Task 3) is tackled under the assumption that each pattern will exist for at least for two time steps. This assumption is necessary because when concept drift occurs is never known before the true output value is observed. If the new pattern only shows at one time step, there is no need to apply this information about drift to improve prediction of the upcoming data instances because the upcoming data instances follow another new pattern. This is a reasonable assumption in real-world applications, especially in scenarios where the concept reoccurs, such as the financial market, weather forecasting, traffic monitoring, and so on. Under this assumption, the current data instance automatically inherits the membership of its previous instance before the output value is observed, and the membership is renewed when the true output value is obtained. Clearly, the time lag of our method is one data instance, and the longer the patterns last, the more effectively FUZZ-CARE performs. We also introduce a kernel function to enhance the classifier for high dimensional tasks. Task 4) is solved for adaptation for mixed drift by introducing three updating strategies: a window-based updating strategy (incremental strategy), an instance-based strategy (online strategy) and a combination strategy of both. These strategies enable FUZZ-CARE to adapt to cases of mixed drift.

The proposed FUZZ-CARE method demonstrates the following advantages on solving the concept drift problem.

- FUZZ-CARE can recognize different patterns in the training set, and measure the degree to which a data instance belongs to different patterns.
- The predictors will continuously adapt to new patterns as well as tracking back to previously learned predictor when a previous pattern reoccurs.
- FUZZ-CARE is able to handle data stream that have different types of concept drift, especially drift that has a mix of gradual and reoccurring concept.
- FUZZ-CARE is also suitable for regression tasks of data streams without drift where FUZZ-CARE functions as a weighted regression predictor. This is very important to apply this method in practice because it is unknown whether a real-world data stream contains drift.

3.2 Definitions and Notations

To mimic the characteristic of data stream where data instances are observed in a sequential way, the learning and evaluation process for data stream is repeatedly activated when new data instances are observed. Specifically, a trained predictor is applied to predict the label value of the new data instance before the true label is obtained. After the true label is obtained, an evaluation process is activated and the instance is included to retrain the existing predictor.

Definition 3.1 (Data Stream). A data stream $D_t = \{(\mathbf{X}_t, y_t) | t = 1, \dots, \infty\}$, generated from distribution \mathcal{P}_t with $p_t(\mathbf{X}, y)$ its probability function or probability density function (*pdf*), is received, where $\{\mathbf{X}_t \in \mathbb{R}^d\}$ is the attribute variable (or the input) consisting of d time series, for some d , and $\{y_t \in \mathbb{R}^1\}$ is the label variable (or the scalar output).

So far, concept drift is defined as Definition. 3.2. This widely accepted definition of concept drift has highlighted the characteristics of *drift*, but has not explained the meaning of “*concept*”. When studying the problem of concept drift, the term “concept” is used to represent the hidden data patterns such as the probability distributions and relationships between X and y . Concept drift is caused by the hidden context, rather than stochastic disturbances. Unlike outliers, a concept will last for a period after it shows, rather than existing momentarily. To present this characteristics of concept, a constraint is added to the current definition of concept drift as is presented in Definition 3.3.

Definition 3.2 (Concept Drift (short version)). Concept drift is defined if the underlying distribution changes, i.e., $p_{t+1}(\mathbf{X}, y) \neq p_t(\mathbf{X}, y)$.

Definition 3.3 (Concept Drift (full version)). Concept drift occurs in a data stream if $\exists t_{d^{(i)}}$ that

$$(3.1) \quad \begin{cases} p_{t+1}(\mathbf{X}, y) \neq p_t(\mathbf{X}, y), \text{ for } t = t_{d^{(i)}} \\ p_{t+1}(\mathbf{X}, y) = p_t(\mathbf{X}, y), \text{ for } t \in (t_{d^{(i)} + \tau_i}, t_{d^{(i+1)}}) \end{cases}$$

where $\forall i, t_{d^{(i+1)}} - t_{d^{(i)}} > 1, t \in \mathbb{Z}^+$ presents the time step, $d^{(i)}$ is an order statistics

denoting the i^{th} drifted time point, and $1 < \tau_i < t_{d^{(i+1)}} - t_{d^{(i)}}$ is specifically for the occurrence of incremental drift.

Remark 3.1. *In the full version definition, a data stream contains concept drift if the data pattern changes at least once, namely $\{t_{d^{(i)}}\} \neq \emptyset$ that $p_{t+1}(\mathbf{X}, y) \neq p_t(\mathbf{X}, y)$, for $t = t_{d^{(i)}}$; in addition, the changed pattern is not ephemeral, but will last for a period (at least last for two time steps), which is manifested by $\forall i, t_{d^{(i+1)}} - t_{d^{(i)}} > 1$. The pattern stays the same in this period that $p_{t+1}(\mathbf{X}, y) = p_t(\mathbf{X}, y)$, for $t \in (t_{d^{(i)}+\tau_i}, t_{d^{(i+1)}})$; here $\tau_i = 1$ when the drift occurs suddenly while $\tau_i > 1$ when the drift occurs incrementally in the period of $(t_{d^{(i)}+1}, t_{d^{(i)}+\tau_i})$. **All drift adaptation methods are at least one-instance delayed. Without the constraint that a new pattern will retain for a period, the adaptation is invalid in principle.***

According to Definition 3.3, a concept will exist for at least a time period of τ once it appears, and the occurrence of concept drift at t_d means the end of one concept. During the time period of one specific concept, the learning aim is to obtain a predictor H_c for $p_c(\mathbf{X}, y)$.

Definition 3.4 (Learning Aim at t -step). To predict the value of the label variable for a data stream at time step t , the learning aim is to obtain a predictor H_t for $p_t(\mathbf{X}, y)$, which can be denoted as

$$(3.2) \quad H_t = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \ell(h, \mathbf{X}, y | (\mathbf{X}, y) \in p_t(\mathbf{X}, y)),$$

where \mathcal{H} is the hypothesis set, $\ell : \mathbb{R}^1 \times \mathbb{R}^1 \rightarrow \mathbb{R}_+$ is the loss function used to measure the magnitude of error.

In the regression task, the loss function is the squared loss, i.e., $\ell(h, \mathbf{X}, y) = \|h(\mathbf{X}) - y\|_2^2$. This chapter only discuss the concept drift problem in a regression task, and the squared loss will be used as the loss function.

Definition 3.5 (Learning Aim for Data Streams). The aim of the whole learning process for a data stream is

$$(3.3) \quad \min_{h_1, h_2, \dots, h_t, \dots} \sum_t \ell(H_t, \mathbf{X}, y | (\mathbf{X}, y) \in p_t(\mathbf{X}, y)).$$

In this chapter, we use the following notations.

- I_{tk} : the value of a characteristic function
- μ_{tk} : fuzzy membership of t -th instance belonging to k -th cluster
- $\{C_k\}$: the k -th cluster centroid
- J : objective function
- J_R : regression term in J
- J_C : clustering term in J
- $\{\theta_k\}$: the parameters for the k -th predictor
- λ_1, λ_2 : weights for the regression term and the regularization term
- η : the Lagrange multiplier
- $\mathcal{K}(\cdot)$: the kernel function

In this chapter, it is assumed that a window of historical data containing K different patterns is initially available. Each pattern can be well expressed by a

set of predictors $\{H_t^k | k = 1, \dots, K\}$. The predictors start to receive a data instance $\mathbf{X}_t \in \mathbf{R}^d$ at a time step t , and the output for this instance is simultaneously predicted as $\hat{y}_t \in \mathbf{R}^1$ based on the current predictor set $\{H_t^k\}$. The true value of this instance, y_t follows. Once y_t is observed, $\{H_t^k\}$ is modified based on (\mathbf{X}_t, y_t) . After receiving a window of new data, these predictors will be replaced by the newly trained predictors. Our goal is to predict the incoming instances as accurately as possible by updating $\{H_t^k\}$.

3.3 A Fuzzy Clustering-based Drift Adaptation Method—FUZZ-CARE

3.3.1 Embedding clustering in objective function

Given the assumption that there is a batch of data instances containing K patterns, the appropriate prediction should be learn K predictors for K patterns. To achieve this prediction, we propose the following objective function:

$$(3.4) \quad H_t = \underset{h^{k_0} \in \mathcal{H}}{\operatorname{argmin}} \sum_{k=1}^K \ell(h^k, \mathbf{X}_t, y_t) \mathbf{I}_{tk},$$

where \mathbf{I}_{tk} is defined as (3.5), and $k_0 = \underset{i}{\operatorname{argmin}} \ell(h^i, \mathbf{X}_t, y_t)$

$$(3.5) \quad \mathbf{I}_{tk} = \begin{cases} 1, & \text{for } k = \underset{i}{\operatorname{argmin}} \ell(h^i, \mathbf{X}_t, y_t) \\ 0, & \text{otherwise.} \end{cases}$$

The core idea of objective function (3.4) is that: Given the current instance belonging to the k -th pattern, the objective function requires H_t to obtain the minimum error on the current data point. Simultaneously, \mathbf{I}_{tk} determines which pattern the current instance belongs to. The objective function seeks to find the optimum for both H_t and k . Here \mathbf{I}_{tk} is a crisp variable, denoting that an instance exactly belongs to a certain pattern, which is not realistic in many real-world cases because the data do not always have well-separated subgroups (Pal and Sarkar, 2014; Zuo et al., 2016). For example, given two patterns of data instances in which one follows $y = x^2$ and the other $y = -x^2$, it is ambiguous to classify the point $(0,0)$ to any pattern. Fuzzy logic is a perfect solution for expressing this type of uncertainty (Zuo et al., 2018; Liu et al., 2018b). In light of this, \mathbf{I}_{tk} is replaced by the fuzzy membership u_{tk} to measure membership degrees. Given $\{\mu_{tk}\}$ the membership of t -th instance belonging to k -th cluster, $\{\mathbf{C}_k\}$ the k -th cluster centroid, $\{\mathbf{X}_t\}$ the input variables at time step t and $\{\boldsymbol{\theta}_k\}$ the parameters for the k -th predictor, we propose to minimize the following objective function:

$$\mathbf{J} = J_R + \lambda_1 J_C + \lambda_2 \sum_{k=1}^K \|\boldsymbol{\theta}_k\|_2^2 \quad (3.6)$$

$$\text{s.t. } \sum_{k=1}^K \mu_{tk} = 1, t \in [1, \dots, N].$$

where J_R and J_C are defined as (3.7) and (3.8) respectively. In (3.8), L^2 norm is used to measure the similarity between the data and the clustering centroids. L^2 norm can also be replaced by other distances or measurements such as fuzzy

entropy (Xie et al., 2018).

$$(3.7) \quad J_R = \sum_{t=1}^N \left(\sum_{k=1}^K \mu_{tk} \mathbf{X}_t \boldsymbol{\theta}_k - y_t \right)^2.$$

$$(3.8) \quad J_C = \sum_{k=1}^K \sum_{t=1}^N \mu_{tk}^2 \|\mathbf{X}_t - \mathbf{C}_k\|_2^2.$$

λ_1 and λ_2 are two pre-assigned parameters. The value of λ_1 denotes the compared importance between clustering and learning process. $\lambda_1 \leq 1$ means that the identifying pattern is more important than the training pattern, and $\lambda_1 \geq 1$ means that the training pattern is more important than the identifying pattern. λ_2 is to control the regularization. In this chapter, $\lambda_1 = 1$, $\lambda_2 = 0.5$ if they are not specified.

3.3.2 Finding an optimum predictor with the minimum

loss

To find the optimum the predictor, two approaches are used here: gradient descent(GD) is applied to find an optimum for $\boldsymbol{\theta}_k$; and Lagrange multipliers for μ_{tk} and C_k . The Lagrange function is defined as:

$$(3.9) \quad \mathcal{L} = J + \sum_{t=1}^N \eta_t \left(\sum_{k=1}^K \mu_{tk} - 1 \right).$$

Our target is to compute the optimal $\boldsymbol{\theta}_k$, C_k and μ_{tk} . Given $\mathbf{U}_X^{(k)} = \{ \mu_{X_{tj}}^{(k)} | \mu_{tk} \mathbf{X}_{tj} \}$, the partial differentiate of \mathcal{L} with respect to these three variables is separately

calculated as:

$$(3.10) \quad \left\{ \begin{array}{l} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_k} = (\mathbf{U}_X^{(k)})^T \left(\sum_{k=1}^K \mathbf{U}_X^{(k)} \boldsymbol{\theta}_k - \mathbf{y} \right) + 2\lambda_2 \boldsymbol{\theta}_k \\ \frac{\partial \mathcal{L}}{\partial \mathbf{C}_k} = -2\lambda_1 \sum_{t=1}^N \mu_{tk}^2 (\mathbf{X}_t - \mathbf{C}_k) \\ \frac{\partial \mathcal{L}}{\partial \mu_{tk}} = 2 \left(\sum_{k=1}^K \mu_{tk} \mathbf{X}_t \boldsymbol{\theta}_k - y_t \right) \mathbf{X}_t \boldsymbol{\theta}_k + \\ \quad 2\lambda_1 \|\mathbf{X}_t - \mathbf{C}_k\|_2^2 \mu_{tk} + \eta_t. \end{array} \right.$$

Given the partial differentiate of θ_k and α the learning rate, $\boldsymbol{\theta}_k$ is updated as:

$$(3.11) \quad \boldsymbol{\theta}'_k = \boldsymbol{\theta}_k + \alpha \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_k}.$$

We carry out the optimum of μ_{tk} and C_k through an iterative optimization method according to (Yang et al., 2011; Huang et al., 2012). When $\nabla_{\mu_{tk}, \mathbf{C}_k, \eta_t} (\boldsymbol{\theta}_k, \mu_{tk}, \mathbf{C}_k, \eta_t) = 0$ we have:

$$(3.12) \quad \mathbf{C}_k = \frac{\sum_{t=1}^N (\mu_{tk}^2 \mathbf{X}_t)}{\sum_{t=1}^N \mu_{tk}^2},$$

and

$$(3.13) \quad \mu_{tk} = \frac{2y_t(\mathbf{X}_t \boldsymbol{\theta}) - 2\mathbf{X}_t \boldsymbol{\theta}_k \sum_{l \neq k} \mu_{tl} \mathbf{X}_t \boldsymbol{\theta}_l - \eta_t}{2[(\mathbf{X}_t \boldsymbol{\theta}_k)^2 + \lambda_1 \|\mathbf{X}_t - \mathbf{C}_k\|_2^2]}.$$

Next, we calculate η_t by the constrain in (3.6). Put (3.13) into $\sum_{k=1}^K \mu_{tk} = 1$, η_t is computed as

$$(3.14) \quad \eta_t = \frac{-1 + \sum_{k=1}^K \frac{2y_t(\mathbf{X}_t \boldsymbol{\theta}_k - S^c)}{2[(\mathbf{X}_t \boldsymbol{\theta}_k)^2 + \lambda_1 \|\mathbf{X}_t - \mathbf{C}_k\|_2^2]}}{\sum_{k=1}^K \frac{1}{2[(\mathbf{X}_t \boldsymbol{\theta}_k)^2 + \lambda_1 \|\mathbf{X}_t - \mathbf{C}_k\|_2^2]}}$$

where $S_k^c = 2\mathbf{X}_t \boldsymbol{\theta}_k \sum_{l \neq k} \mu_{tl} \mathbf{X}_t \boldsymbol{\theta}_l$. By replacing η_t in (3.13) by (3.14), we obtain the iteration of μ_{tk} .

3.3.3 The kernel clustering-based version

The current cluster may not be effective for high dimensional and non-linear clustering tasks. To overcome this drawback, kernel functions have been introduced into the clustering process to solve non-linear problems. With the kernel function, J_C is rewritten as

$$(3.15) \quad J_C^{\mathcal{K}} = \frac{\lambda_1}{N} \sum_{k=1}^K \sum_{t=1}^N \mu_{tk}^2 \|\Phi(\mathbf{X}_t) - \Phi(\mathbf{C}_k)\|_2^2,$$

where

$$(3.16) \quad \begin{aligned} \|\Phi(\mathbf{X}_t) - \Phi(\mathbf{C}_k)\|_2^2 &= \mathcal{K}(\mathbf{X}_t, \mathbf{X}_t) \\ &\quad - 2\mathcal{K}(\mathbf{X}_t, \mathbf{C}_k) + \mathcal{K}(\mathbf{C}_k, \mathbf{C}_k). \end{aligned}$$

The kernel-based objective function is (3.17).

$$(3.17) \quad \begin{cases} \mathbf{J}^{\mathcal{K}} = \mathbf{J}_R + \lambda_1 \mathbf{J}_C^{\mathcal{K}} + \lambda_2 \sum_{k=1}^K \|\boldsymbol{\theta}_k\|_2^2 \\ \text{s.t. } \sum_{k=1}^K \mu_{tk} = 1. \end{cases}$$

As the kernel function does not affect the updating process of $\boldsymbol{\theta}$, $\boldsymbol{\theta}_k$ will still be updated by (3.11). In the kernel version, μ_{tk} is iterated as

$$(3.18) \quad \mu_{tk} = \frac{2y_t \mathbf{X}_t \boldsymbol{\theta}_k - S_l^c - \eta_t^{\mathcal{K}}}{2[(\mathbf{X}_t \boldsymbol{\theta}_k)^2 + \lambda_1 \|\mathbf{X}_t - \mathbf{C}_k\|_2^2]},$$

where $S_l^c = 2\mathbf{X}_t \boldsymbol{\theta}_k \sum_{l \neq k} \mu_{tl} \mathbf{X}_t \boldsymbol{\theta}_l$ and $\eta_t^{\mathcal{K}}$ is (3.19).

$$(3.19) \quad \eta_t^{\mathcal{K}} = \frac{-1 + \sum_{k=1}^K \frac{2y_t (\mathbf{X}_t \boldsymbol{\theta}_k - S_l^c)}{2[(\mathbf{X}_t \boldsymbol{\theta}_k)^2 + \lambda_1 (\mathbf{X}_t, \mathbf{X}_t) - 2\mathcal{K}(\mathbf{X}_t, \mathbf{C}_k) + \mathcal{K}(\mathbf{C}_k, \mathbf{C}_k)]}}{\sum_{k=1}^K \frac{1}{2(\mathbf{X}_t, \mathbf{X}_t) - 2\mathcal{K}(\mathbf{X}_t, \mathbf{C}_k) + \mathcal{K}(\mathbf{C}_k, \mathbf{C}_k)}}.$$

For a Gaussian kernel with $\mathcal{K}(x, y) = \exp(-\|x - y\|^2 / 2\sigma^2)$, \mathbf{C}_k is iterated as

$$(3.20) \quad \mathbf{C}_k = \frac{\sum_{t=1}^N (\mu_{tk}^2 \mathcal{K}(\mathbf{X}_t, \mathbf{C}_k)) \mathbf{X}_t}{\sum_{t=1}^N \mu_{tk}^2 \mathcal{K}(\mathbf{X}_t, \mathbf{C}_k)}.$$

This chapter aims to validate whether the clustering helps learn the regression predictor for the data stream with concept drift. We want to select a kernel function that is available for all tested data streams rather than selecting the best kernel function specifically for each data stream. Therefore, the widely

Algorithm 3.1: The learning process in FUZZ-CARE

Input : $X_T, Y_T, \lambda_1, \lambda_2, \alpha, K$.
Output: \hat{H} : the trained predictor with following parameters:
 U : fuzzy membership Mixture
 C : the cluster center
 Θ : parameters of regression predictors
 K_0 : the optimal clustering number
Initialization: $U \leftarrow \text{Rand}(0, 1)$; $C, \Theta, co \leftarrow 0$

```

1 for  $k = 2$  to  $K$  do
2   for  $o = 1$  to  $MaxIteration$  do
3     Update  $\Theta^{(k)}, U^{(k)}, C^{(k)}$  by (3.11), (3.18) and (3.20);
4     Calculate  $J^{\mathcal{K}}$  in (3.17);
5     if  $J^{\mathcal{K}}(o) > J^{\mathcal{K}}(o - 1)$  then
6        $co = co + 1$ ;
7       if  $co \geq 6$  then
8         break;
9       end
10    else
11       $co = 0$ ;
12    end
13  end
14   $\hat{y}_t^{(k)} = \hat{H}(X_t | U, C, \Theta, k)$ ;
15   $K_o = \underset{k}{\operatorname{argmin}} \sum_t |\hat{y}_t^{(k)} - y_t|$ ;
16 end
17 return  $\hat{H}(\cdot | U, C, \Theta, K_o)$ 

```

used Gaussian kernel is selected, and the experiment results show that the Gaussian kernel is available in all the tested data streams. Given the training set $[X_T, Y_T] = \{X_t, y_t | t = 1, \dots, T\}$, λ_1 and λ_2 in (3.6), α the learning rate, and K the maximum number of clusters, the pseudo code of the training process in FUZZ-CARE is listed in Algorithm 3.1. Line 3 updates the parameters in each iteration, and the iteration ends when o reaches the *MaxIteration* or $J^{\mathcal{K}}$ continuously increases for six iterations.

3.3.4 The general procedure and pseudocode of FUZZ-CARE

So far, we have a method that can provide a linear or non-linear fuzzy boundary to separate the mixed patterns into several groups and assign each instance the membership degree that it belongs to these groups. Its mechanism is simply represented in Figure 3.3. In Figure 3.3, the black dots represent a 2-dimensional dataset, where two patterns exist in the same training set. The red line on the left is the fitted line of the linear regression method. The red and blue lines on the right are the fitted lines of FUZZ-CARE. A conventional machine learning method builds a predictor directly on the training set. If multiple patterns exist in the training set, it is likely that an input will map to several probable outputs, as in the two outputs case in Figure 3.3. Under such circumstances, the learner will be confused and cannot be well trained. Therefore, the red line is in the middle of the data and does not fit any pattern. In contrast, FUZZ-CARE solves this problem by embedding clustering in the learning procedure. It identifies two patterns in the training set and builds two separate linear predictors. However, a precondition for using it to predict a new instance is that this instance can be presented by a linear combination of the existing patterns. If an instance belongs to a totally new pattern, it is impossible to obtain an accurate prediction. To make our approach adaptable to situations in which new patterns occur and may reoccur, three updating strategies have been introduced. One is a fixed window-based updating procedure to incrementally include new patterns in our pattern base; one is an online strategy which updates the predictor parameters fully based on the predictor of the previous instance when the true value of a new

3.3. A FUZZY CLUSTERING-BASED DRIFT ADAPTATION METHOD—FUZZ-CARE

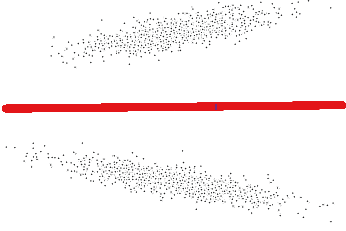
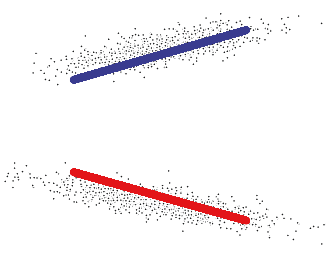
Method	Conventional learning	FUZZ-CARE
Mechanism	Directly learns a predictor	Learns with clustering
Example		
Result	One input maps to two outputs Predictor cannot be well learned	Well learned

Figure 3.3: The difference between FUZZ-CARE and other learning methods when the same input maps to two outputs. Black dots represent a 2-dimensional dataset, where two patterns exist in the same training set. The red line on the left is the fitted line of the linear regression method. The red and blue lines on the right are the fitted lines of FUZZ-CARE.

instance is observed; and one is a combination of the previous two strategies. The combination strategy uses an online strategy prior to the arrival of a window of instances, and when a whole window of instances has arrived, the window-based updating strategy is applied. Given the upcoming data instances $\{X_t, y_t\}$, and $\hat{H}_t(\cdot | \mathbf{U}, \mathbf{C}, \Theta, K_o)$, X_T, Y_T, K from Algorithm 3.1, the procedure of Algorithm 3.2: FUZZ-CARE is presented. Model_U and Window_U are switches for controlling an online or incremental updating strategy. Model_U = 1 means that an online strategy is used, and Window_U = 1 that an incremental updating strategy is used. The combination strategy is applied when both equal 1. The flowchart of FUZZ-CARE combined with different strategies is shown in Figure 3.4. During the experiments, we assume that at least one batch of historical data is available, and this historical data will be used as the initial training set to learn the first prediction predictor in FUZZ-CARE. If there are not enough data instances in

Algorithm 3.2: FUZZ-CARE

Input : $X_t, \hat{H}_t(\cdot|U, C, \Theta, K_o), X_T, Y_T, K$
Output: \hat{y}_t

```

1 for  $t = T + 1$  to  $\infty$  do
2    $\hat{y}_t = \hat{H}_{t-1}(X_t|\mu_{t-1}, C, \Theta, K_o)$ ;
3    $X_T = [X_T, X_t]$ ;
4   return  $\hat{y}_t$ ;
   Input :  $y_t$ .
5    $Y_T = [Y_T; y_t]$ ;
6   if  $Model\_U = 1$  then
7     Update  $\mu_t, C, \Theta$ 
8   if  $Window\_U = 1$  then
9     if A new window of instances is arrived then
10      Train  $\hat{H}_t$  by Algorithm 3.1
11    end
12  else
13     $\hat{H}_{t+1} \leftarrow \hat{H}_t$ 
14  end
15 end

```

the historical data, it is necessary to wait to collect data to compose the training set.

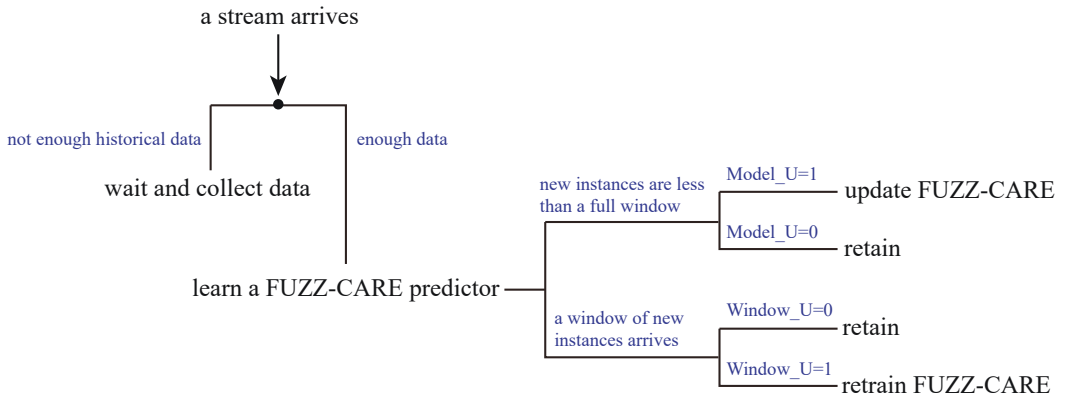


Figure 3.4: FUZZ-CARE flowchart showing different updating strategies.

3.4 Experimental Evaluations

The experiments in this paper are designed to demonstrate that the proposed FUZZ-CARE can be applied for regression of data streams with or without concept drift. However, the ground truth of whether a data stream contains drift or which types of drift exist in a real-world data stream is unknown. Therefore, we first conduct experiments on the synthetic data streams in Section 3.4.1. As the drift is manually added to the same original data (noted as Non-Drift), the prediction accuracy of Non-Drift can be used as a baseline to test whether the concept drift problem has been solved. The experiments in this section are to verify that FUZZ-CARE improves prediction accuracy because it solves the concept drift problem. In the experiments on synthetic data, all four types of concept drift are involved and mixed. Section 3.4.2 presents the prediction results for real-world data streams, where FUZZ-CARE is compared to other state-of-the-art regression methods for data streams with concept drift. Section 3.4.3 shows the effectiveness of FUZZ-CARE on data without concept drift. The experiments in this section is to demonstrate that FUZZ-CARE can be used in all kinds of datasets although FUZZ-CARE especially tackles the concept drift problem. None of the datasets in this section have a time label, so they are considered not to have concept drift. The consistently good performance of the above three parts shows that FUZZ-CARE can be used for regression of data streams, especially data streams with mixed concept drift. Table 3.1 gives a brief structure of the experimental design.

Table 3.1: Experimental design in this section.

Section	Data	Experimental aim	Main Results
3.4.1	d1	to verify whether FUZZ-CARE can solve different types of concept drift and mixed drift	Table 3.2
3.4.1	d1	compare FUZZ-CARE ¹ and FUZZ-CARE ² by their learned parameters and learning errors	Figure 3.6
3.4.2	d2	to demonstrate that FUZZ-CARE can be used for regression of data streams	Table 3.3
3.4.3	d3	to demonstrate that FUZZ-CARE can also used for regression of stationary data	Table 3.4
3.4.4	d2	to demonstrate that FUZZ-CARE has significant advantages when handling real-world concept drift problems	Table 3.5, Table 3.6
3.4.5	d2	to demonstrate the robustness of FUZZ-CARE	Figure 3.7

d1: synthetic regression data
d2: real-world regression data streams
d3: real-world stationary regression datasets

As a data stream $\{(\mathbf{X}_t, y_t) | t = 1, \dots, \infty\}$ is assumed to have infinite length, a finite period of the data is used to validate the effect of the proposed FUZZ-CARE, and the data used in this section is considered to be an available period of an infinite data stream, denoted by $\{(\mathbf{X}_t, y_t) | t = 1, \dots, T\}$. A prequential evaluation is applied to test the effectiveness of predictors that each data instance is first used to test the predictor, and then to train the predictor. During the experimental procedure, data instances are assumed to arrive one by one. After enough historical data have been obtained, denoted as $\{(\mathbf{X}_t, y_t) | t = 1, \dots, t_0\}$, FUZZ-CARE is activated to predict the upcoming data instances, namely $\{(\mathbf{X}_t, y_t) | t = t_0, \dots, T\}$. There are many similar predictors with different parameters in the proposed FUZZ-CARE approach. Each predictor is obtained at a specific time step t and used to predict y_t . The effect of each predictor is identified by the *absolute error* at time step t . After the absolute error has been computed at t , the true value of y_t is obtained, and this data instance (\mathbf{X}_t, y_t) is used to train the predictor at time step $t + 1$. The effect of FUZZ-CARE is identified by the average of the absolute error over all available time steps, namely the *mean absolute error* (MAE in (3.21)). In all experiments, the pre-assigned parameters were as follows: $\alpha = 0.05$, $\lambda_1 = 1$, $\lambda_2 = 0.5$, $K = 5$, and $\sigma^2 = 0.01$ in the kernel function. Section 3.4.4 show the statistical test result of the accuracy comparison between FUZZ-CARE and other state-of-the-art methods. The parameter analysis and computation complex are conducted in Section 3.4.5.

$$(3.21) \quad \text{MAE} = \frac{1}{(T - t_0)} \sum_{t=t_0}^T |\hat{y}_t - y_t|,$$

where y_t and \hat{y}_t are the true and evaluated output value separately.

In this paper, the real-world data is downloaded from several popular data

sources, which are given when the data is introduced. As these data have been widely used to validate data stream mining methods, they are well-cleaned, and FUZZ-CARE does not embed any pretreatment method. FUZZ-CARE can be activated after data pretreatment techniques such as simplifying the data for better prediction performance when it is used to predict other data streams.

3.4.1 Validation on synthetic data

Data: Similar to the generation procedure of synthetic data in (Li et al., 2018), we generated six datasets by several parameter-changing linear functions: *Non-Drift* is a data stream without any drift; *Virt-Drift* only contains virtual drift, i.e., only the way of generating input differs by time; *Sudd-Drift* contains a real sudden drift; *Incr-Drift* means a real incremental drift occurs over a period; *Rec-Drift-Grad* contains two different patterns which appear alternately; *Rec-Drift-Mix*, contains a pattern that incrementally changes to a new one, then sharply changes back to the initial pattern after lasting for some time. The datasets were generated as follows:

1) Data that contains no drift (Non-Drift). Two random samples X_1 and X_2 were drawn from a normal distribution $\mathcal{N}(\mu, \sigma^2 | \mu = 10, \sigma^2 = 100)$ as the first two values for the input. The rest of the input was generated by $X_t = \beta_1 X_{t-1} + \beta_2 X_{t-2} + \eta_t$, and the corresponding output series was generated by $Y_t = \theta_0 + \theta_1 X_t + \epsilon_t$ where $\{\eta_t\}$ and $\{\epsilon_t\}$ are random error series. We generated 2000 data samples in this way with $\beta_1 = 0.5, \beta_2 = -0.2, \theta_0 = 10, \theta_1 = 1$.

2) Data that contains virtual drift (Virt-Drift). The first 1000 data samples were generated in the same way as Non-Drift and with same parameters.

The subsequent 1000 samples were generated in the same way but with the parameters $\beta_1 = -0.5, \beta_2 = 0.2$. A sudden virtual drift clearly occurred at the 1001st sample.

3) Data that contains sudden real drift (Sudd-Drift). The first 1000 data samples were the same as Non-Drift. The subsequent 1000 samples were generated in the same way as No-Drift but with the parameters $\theta_0 = -10, \theta_1 = -1$. Real sudden drift occurred at the 1001st sample.

4) Data that contains incremental real drift (Incr-Drift). The first 1000 data samples were the same as Non-Drift. The input of the subsequent 1000 samples were the same as Non-Drift. The output of the 1001st to 1500th were generated by $\{\theta_0 + f_0(t)\} + \{\theta_1 + f_1(t)\}X_t + \epsilon_t$ where $f_i(t) = (\theta'_i - \theta_i) \times \frac{t-1000}{500}$. The output of the last 500 samples were generated by $Y_t = \theta'_0 + \theta'_1 X_t + \epsilon_t$. The parameters were $\theta_0 = 10, \theta'_0 = -10, \theta_1 = 1$ and $\theta'_1 = -1$. An incremental drift occurred from the 1001st to 1500th samples, and the new pattern proceeded after 1500th samples.

5) Data that contains gradual real drift and reoccurring concept (Rec-Drift-Grad). The first 2000 samples were generated in the same way as Sudd-Drift. The first 1000 samples presented a pattern and the last 1000 samples presented a different pattern. 100 samples belonging to either of these two patterns were drawn as the subsequent samples. This procedure was repeated until 12000 samples in total were obtained. Except for the first 2000 samples, these two patterns appeared every 100 samples in a totally random way, so gradual drift appeared in some periods (as shown in the blue rectangle in Figure 3.5, and each pattern lasted for some time (100 time steps here). This example is reasonable to be a special case of reoccurring concept.

6) Data that contains sudden and incremental drift, and reoccurring concept (Rec-Drift-Mix). Consists of six Incr-Drift datasets, in the order of one followed by the next, i.e., every 2000 samples were the Incr-Drift dataset in 4). In this dataset, an incremental drift is followed by a sudden drift, and past patterns reoccur.

The synthetic data are presented in Figure 3.5, and the black and red dots denote two different patterns. In the sub-figures 1)–4), the dots are drawn in a 2-dimensional plane with axis of X_t and Y_t axis; while in 5) and 6) subplots, they are drawn in a 3-dimensional sphere with an additional axis of time T . It can be seen in subplot 5) that a gradual drift occurs in the period highlighted by the blue outline: the first time the red pattern changes to a black pattern, it remains in that pattern for a relatively short period before turning back to the red pattern. Instead of staying at the red pattern, it changes to the black pattern again, this time for a longer period. Subplot 6) represents a reoccurring concept with mixed sudden and incremental drift: incremental drift appears when the black pattern changes to red, while sudden drift appears when the red pattern changes to black.

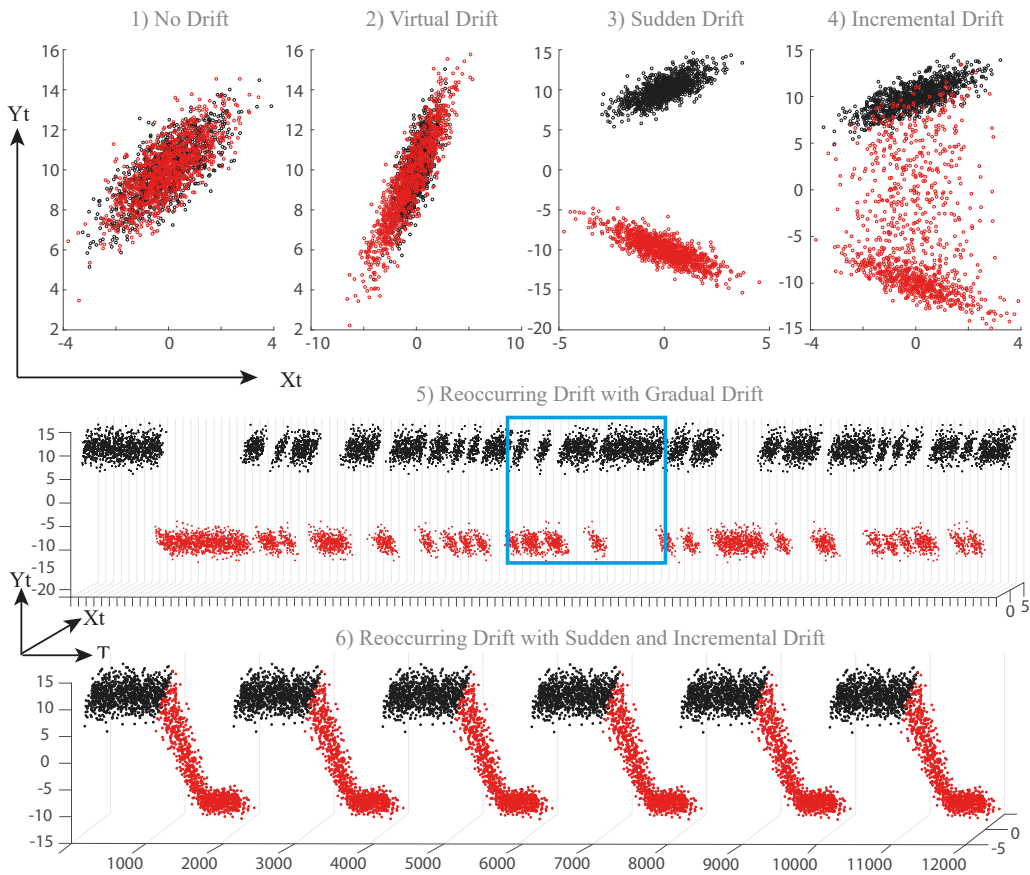


Figure 3.5: Generated synthetic data with different types of drift. We generate one no-drift data on which we base five examples of drifting data. The red and black dots represent two different patterns. In the sub-figures 1)–4), the dots are drawn in a 2-dimensional plane with axis of X_t and Y_t axis; while in 5) and 6) subplots, they are drawn in a 3-dimensional sphere with an additional axis of time T .

Table 3.2: A comprehensive comparison of different editions of FUZZ-CARE method on different types of drift.

FUZZ-CARE ^{0,1,2,3}	Linear Regression		⁰ Model_U = 0 Window_U = 0		¹ Model_U = 0 Window_U = 1		² Model_U = 1 Window_U = 0		³ Model_U = 1 Window_U = 1	
	<i>N</i> = 500, <i>W</i> = 100	- <i>N</i> = 2000	- <i>N</i> = 2000	- <i>N</i> = 2000	- <i>N</i> = 2000	- <i>N</i> = 2000	- <i>N</i> = 2000	- <i>N</i> = 2000		
Non-Drift	0.80	-	1.22	-	1.21	-	0.99	-	1.18	-
Virt-Drift	0.78	-	1.42	-	1.37	-	1.06	-	1.30	-
Sudd-Drift	13.54	-	11.89	-	4.96	-	1.69	-	3.34	-
Incre-Drift	10.20	-	8.64	-	5.38	-	3.66	-	4.03	-
Rec-Drift-Grad	8.80	10.00	9.07	1.90	2.29	1.89	6.39	4.19	2.34	2.19
Rec-Drift-Mix	8.17	8.19	7.68	2.56	3.05	2.70	29.07	11.42	2.13	1.91

Results: We comprehensively compare the different editions to investigate the applicability of our method. The results are summarized in Table 3.2. Except for the “Linear Regression” column, all other columns show the result of FUZZ-CARE method and demonstrate the outcome of gradually switching on different updating strategies. The results from a linear predictor are listed here as a baseline. The length of the training set for the synthetic data experiments, unless otherwise specified, is $N = 500$ and the window size is $W = 100$. The editions of our method are controlled by two parameters: Model_U and Window_U. Model_U = 1 means that we use the incremental updating strategy during the experiments, and Window_U = 1 is for the online strategy.

1) The original FUZZ-CARE (FUZZ-CARE⁰) is effective when the training set contains all potential patterns. As noted in the FUZZ-CARE⁰ column, when $N = 500$, FUZZ-CARE seems to be useless because it performs as poorly as Linear Regression. However, the MAEs decrease to 1.90 and 2.56 for RecDrift-Grad and Rec-Drift-mix when $N = 2000$, which is a big improvement. This is because the learned predictor only contains the black pattern when $N = 500$, but contains both black and red pattern when $N = 2000$. This improvement exists whether drift occurs in a/an sudden, incremental or gradual way.

2) A window-based updating strategy assists FUZZ-CARE⁰ to make predictions for new patterns that will appear in the future. In the FUZZ-CARE¹ column, our method performs satisfactorily even when $N = 500$. Therefore, we conclude that the window-based updating strategy is able to overcome the failure of the original FUZZ-CARE to adapt to new patterns by periodically absorbing new patterns.

3) Updating the predictor’s parameters in every instance is not a

good strategy when mixed drift occurs. In Table 3.2, it can be seen that FUZZ-CARE² obtains extremely poor accuracy on Red-Drift-Mix. This failure is less dependent on whether the initial training set contains all potential patterns because the prediction is still poor when $N = 2000$. Figure 3.6¹ presents the estimated parameters $\hat{\theta}$ of FUZZ-CARE², as well as those of FUZZ-CARE¹ as the control group. In subplot a), black triangles or dots are the estimated values of θ at every instance while the blue and red dots represent FUZZ-CARE². There are two linear patterns in this data. θ_0 and θ_1 denotes two parameters of one pattern, and θ'_0 and θ'_1 denotes the corresponding parameters for the other pattern. Subplot a) shows that after training the predictor, the parameters of FUZZ-CARE¹ and FUZZ-CARE² are at a similar level, and $\hat{\theta}_0$ of different fuzzy clusters are the same. As the new data instances arrive, $\hat{\theta}$ of FUZZ-CARE¹ starts to update every 100 instances. After updating for 5 windows, θ_0 splits to around 10 and -10, and θ_1 are around 1 and -1. This estimation is very closed to the ground truth. However, $\hat{\theta}$ of FUZZ-CARE² shows an increasingly unsteady trend as the new data instances arrive.

4) FUZZ-CARE with the combination of window-based and instance-based updating strategies is an effective solution for predicting when drift will appear in the data stream. The column FUZZ-CARE³ column in Table 3.2 shows that FUZZ-CARE achieves satisfactory results in each synthetic dataset, especially Rec-Drift-Grad and RecDrift-Mix.

¹To draw this figure, we estimate the predictor with a non-normalized data.

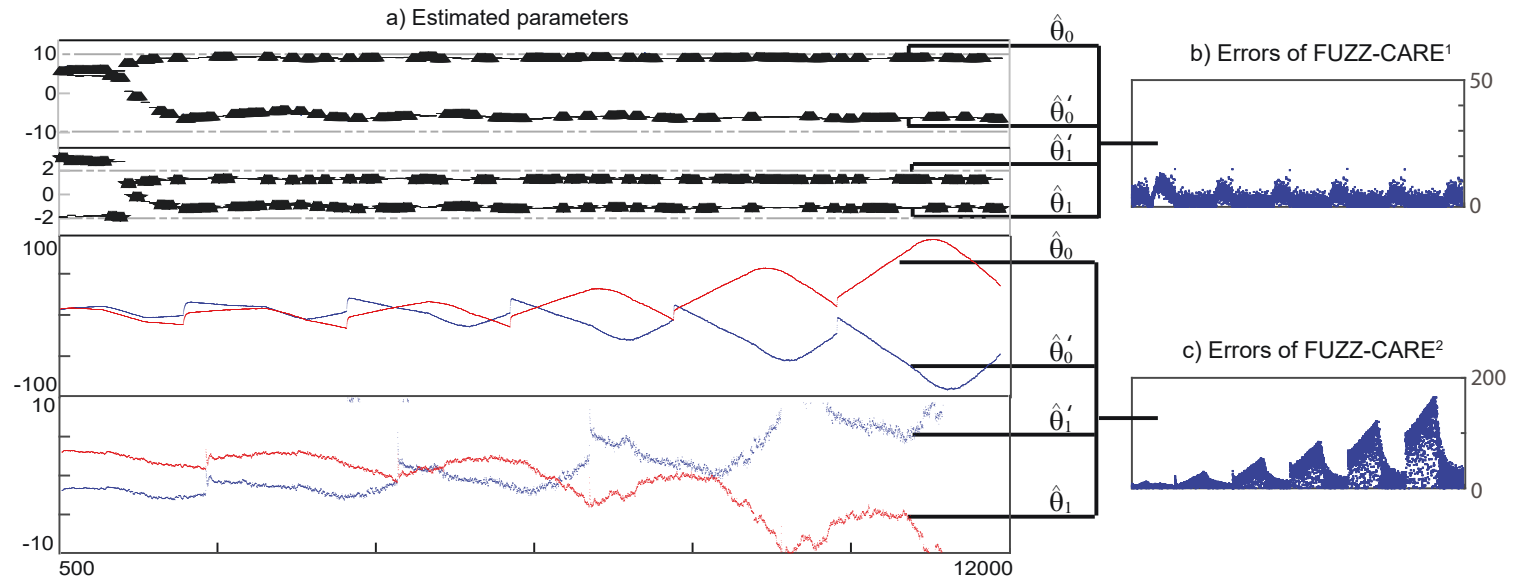


Figure 3.6: Estimated parameters and errors of FUZZ-CARE^{1,2} for Rec-Drift-Mix.

Discussion: We conclude the validation on synthetic data by making the following three points: 1. The type of drift does affect the performance of an algorithm. For example, FUZZ-CARE¹ achieves the best result for Rec-Drift-Grad, while FUZZ-CARE³ is preferred in the Rec-Drift-Mix case; 2. A sole updating strategy may prove to be useless when mixed drift occurs, such as FUZZ-CARE² in Rec-Drift-Grad; 3. Generally speaking, FUZZ-CARE performs better when abundant historical data are available because they are more likely to contain useful knowledge. 4. We suggest to use FUZZ-CARE with a combined updating strategy if there is not any prior experience that which types of drift exist in the dataset, because FUZZ-CARE³ has a good and the most stable performance in general.

3.4.2 Comparison on real-world data streams

In this section, the effectiveness of FUZZ-CARE is validated by experiments on real-world data streams. We presented the prediction accuracy of FUZZ-CARE⁰ (does not respond to drift) and FUZZ-CARE³ (keeps updating even when there is no drift), and compared it to 5 benchmark drift adaption methods. The benchmarks comprise two tree predictors—FIMT-DD (Ikonomovska et al., 2011a) and ORTO (Ikonomovska et al., 2011b), two rule predictors—AMR and its ensemble version, metaAMR (Duarte et al., 2016), and Perceptron (Bifet et al., 2010c). In FIMT-DD, linear regression predictors and the incremental (stochastic) gradient descent method are used in the leaves of the tree. ORTO is an upgraded version of FIMT-DD using on-line option trees. In AMR and metaAMR, a linear regression predictor is contained in each rule and trained

using an incremental gradient descent method. Perceptron is a basic online regression benchmark predictor. All benchmarks are implemented by MOA (Bifet et al., 2010a)(<https://moa.cms.waikato.ac.nz/>).

Data: Eight real-world data streams were employed in our experiments. Their detailed descriptions are as follows:

1) **CCPP** contains 9568 data instances collected from a Combined Cycle Power Plant over six years (2006-2011), when the power plant was set to work at full load. Features consist of the hourly average ambient variables temperature (T), ambient pressure (AP), relative humidity (RH) and exhaust vacuum (V) to predict the net hourly electrical energy output (EP) of the plant. The dataset is available from the UCI machine learning repository (<http://archive.ics.uci.edu/ml>). Although it is a real-world data stream, it has been proved in (Wang et al., 2017) **not to have any drift**. We test it assuming that it may contain drift and re-evaluate the drift issue based on the test results.

2) **House** collects all the housing block groups in California from the 1990 Census. A block group on average includes 1425.5 individuals living in a geographically compact area. It contains 20,640 data instances with eight features comprising median income, housing median age, total number of rooms, total number of bedrooms, population, number of households, latitude and longitude, and the target output is median house price. This data is not a data stream as it does not have time stamp. We use this data set here because the data size is large, and this data is used by existing drift adaptation methods for regression cases to validate the modelling effectiveness Ikonomovska et al. (2011a); Duarte et al. (2016). It is accessible at <http://lib.stat.cmu.edu/datasets/houses.zip> (Pace and Barry, 1997).

3) Sensor contains 2,219,803 consecutive records of temperature, humidity, light and sensor voltage collected from 54 sensors deployed in the Intel Berkeley Research Lab over a two-month period. The dataset is available at (Zhu, 2010). To make it a regression task, we selected the records of four sensors—sensors 3, 8, 20, and 46—located in different parts of the Lab, and used temperature, humidity, light as input features, and sensor voltage as the target out. The light during working hours is generally stronger than during the night, and the temperature of specific sensors may rise regularly during meetings. The lengths of sensors 3, 8, 20, and 46 are 46,633, 15,808, 28,832, and 52,988 respectively.

4) SMEAR is a set of 30-minute interval environment observation data collected from the SMEAR II station which contains 140,576 data instances of 43 variables from 00:15, on 15 April 2005 to 23:45 on 14 April 2013. The regression task is to predict solar-radiation using 43 variables. Six of the 43 variables are time stamps that are not considered prediction features in the predictor. The remaining environmental features have been introduced in (Žliobaitė et al., 2014b). There are many missing values in this data and we eliminate them in the same way as (Žliobaitė et al., 2014b).

5) Solar is provided by NASA and contains 32,686 records of meteorological data from the HI-SEAS weather station from 23:55:26 29 September 2016 to 00:00:02 1 December 2016 (Hawaii time) in the period between Mission IV and Mission V. The data interval is roughly 5 minutes. The input features are temperature (unit: degrees Fahrenheit), humidity (unit: percent), barometric pressure (unit: Hg), wind direction (unit: degree), wind speed (unit: miles per hour) and the target variable is solar radiation (unit: watts per meter²). The dataset is available at <https://www.kaggle.com/dronio/SolarEnergy/data>.

Table 3.3: Validation on real-world data streams. MAE and its corresponding rank are used as the evaluations

Data Streams	ORTO	FIMTDD	metaAMR	AMR	Per	K-means	FUZZ-CARE ⁰	FUZZ-CARE ³
CCPP	4.53E+02 (8)	3.57E+00 (3)	3.32E+00 (1)	3.42E+00 (2)	3.65E+00 (4)	5.26E+00 (5)	1.46E+01 (7)	5.59E+00 (6)
House	9.42E+04 (7)	5.54E+04 (5)	4.60E+04 (2)	4.83E+04 (3)	4.90E+04 (4)	6.72E+04 (6)	9.87E+04 (8)	3.94E+04 (1)
Sensor3	6.62E-02 (6)	7.14E-03 (2)	1.58E-02 (5)	7.69E-03 (3)	6.85E-03 (1)	7.15E-02 (7)	1.78E-01 (8)	1.55E-02 (4)
Sensor8	1.69E-01 (7)	9.68E-03 (4)	7.26E-03 (3)	6.57E-03 (2)	5.95E-03 (1)	3.23E-02 (6)	2.01E-01 (8)	1.72E-02 (5)
Sensor20	9.60E-01 (8)	7.95E-01 (7)	1.14E-02 (3)	8.19E-03 (2)	7.92E-01 (6)	7.11E-02 (5)	6.99E-02 (4)	7.90E-03 (1)
Sensor46	4.00E-01 (8)	1.57E-01 (5)	1.74E-01 (6)	2.02E-01 (7)	1.56E-01 (4)	4.73E-02 (1)	1.00E-01 (3)	5.25E-02 (2)
SMEAR	3.39E+01 (6)	2.34E+01 (4)	1.98E+01 (3)	1.44E+01 (2)	3.75E+01 (7)	2.19E+10 (8)	2.43E+01 (5)	1.04E+01 (1)
Solar	2.25E+02 (6)	1.14E+02 (4)	9.39E+01 (2)	9.52E+01 (3)	1.30E+02 (5)	2.10E+02 (6)	5.25E+02 (7)	8.66E+01 (1)
Average Rank	7.00	4.25	3.13	3.00	4.00	5.50	6.25	2.63

Results: If a method applies a window-based updating strategy during the experiments, such as FIMT-DD and our methods, the training set is the first 2000 data instances and the window size is 200. The general accuracy of our method and the compared methods is shown in Table 3.3.

1) We conclude that all eight data streams contain drift because FUZZ-CARE³ is uniformly better than FUZZ-CARE⁰. Even in cases such as CCpp, Sensor3, Sensor8, where FUZZ-CARE is not as good as some benchmarks, the performance of FUZZ-CARE³ is still significantly improved compared to FUZZ-CARE⁰. Therefore, we recommend their use to study future concept drift problems.

2) Fuzzy c-means is better than K-means at identifying existing patterns in data streams. The MAEs in the K-means column are computed by replacing the fuzzy c-means clustering method with the K-means method in FUZZ-CARE. Fuzz c-means clustering is better than K-means on all data streams except Sensor 46, where it is slightly worse. However, K-means obtains a very poor result for SMEAR. As discussed in Section 3.3, it is difficult to unequivocally identify that an instance belongs to a certain pattern in real-world cases. A crisp clustering method may lead to serious failure, such as K-means in the SMEAR case.

3) In general, we conclude FUZZ-CARE³ is an effective method for solving regression tasks for data streams under the concept drift problem. This is because of all tested methods, FUZZ-CARE³ has the smallest average rank of 2.63 over these data streams.

4) FUZZ-CARE is especially suitable for applying in reoccurring concept of mixed cases. Unlike synthetic data streams, it is difficult to confirm

exactly which types of drift occur in real-world data streams, nor is the frequency of their occurrence known. Therefore, we always assume that each data stream may contain all types of drift. However, we can use prior experience on specific topics to infer the drift information for certain data streams. In a typical economic problem, for example, house price prediction is very like to follow certain marketing rules; SMEAR and Solar are atmospheric tasks, which follows periodically physical rules. Some of these rules have been discovered by economists and physicists, but some have not. These rules result in the reoccurring concept phenomenon in data streams. Therefore, we deem House, SMEAR and Solar to contain reoccurring concept based on this prior experience. The experiment results in Table 3.3 validate this argument. FUZZ-CARE has been designed to handle drift by aiming at reoccurring concept, and it performs better than other drift adaptation methods on the data streams House, SMEAR, and Solar.

5) The effectiveness of FUZZ-CARE is not affected by the initialization of a membership matrix. During the fuzzy clustering process, a membership matrix is randomly generated as the initial membership matrix. To validate whether this randomly generated membership matrix affects FUZZ-CARE's performance, we repeat the experiments 20 times. Although there is a slight difference in the predicted value of some instances, the MAEs of each data stream are the same for all 20 times as they are in Table 3.3.

3.4.3 Comparison on stationary datasets

Following the detailed analysis of our method on real-world data streams, we consider the performance of FUZZ-CARE on real-world stationary datasets.

Although FUZZ-CARE is especially proposed for concept drift adaptation in data streams, it should also be effective for solving typical regression tasks, i.e., no drifting datasets. In this section, we validate the method on stationary datasets.

Data: 13 stationary regression datasets of various topics are used. All datasets and their detailed descriptions are available on OpenML (OpenML). The data size is given in Table 3.4. The length of each stationary dataset is small in relation to the data stream, so we use the first 50 instances as the training set, and set a window size of 50.

Results: 1) **FUZZ-CARE can also be applied to predict stationary regression datasets.** The evaluation accuracy of FUZZ-CARE is presented in Table 3.4. The Size column is “the number of instances \times the number of features”; The Ranks column and the Average rank show the ranking of each method in ascending order. The average rank of FUZZ-CARE⁰ is 3.85 and that of FUZZ-CARE³ 1.92, which denotes that FUZZ-CARE is also suitable for predicting stationary regression datasets.

2) **Given the same training set, FUZZ-CARE performs better with updating strategies.** The length of “1-Pollution”, “2-Mbgrade”, and “3-Auto93” datasets is less than 101, which means that only the online updating strategy is applied. The prediction accuracy of FUZZ-CARE⁰ is almost the same as that of FUZZ-CARE³. However, FUZZ-CARE³ significantly improves the performance of the other datasets, especially the larger datasets such as “6-Lowbwt”, “9-Cpu”, “12-Auto-MPG” and “13-Pbc”. Therefore, we recommend using FUZZ-CARE with updating strategies even for stationary regression tasks.

Table 3.4: Comparison results on 13 stationary datasets.

Datasets	Size	ORTO	FIMT-DD	metaAMR	AMR	Per	FUZZ-CARE ⁰	FUZZ-CARE ³
1-Pollution	60×16	1.38E+02	1.38E+02	5.12E+01	5.12E+01	1.09E+02	4.00E+01	4.09E+01
2-Mbgrade	61×3	4.68E-01	4.68E-01	2.67E-01	2.67E-01	5.69E-01	2.90E-01	2.92E-01
3-Auto93	93×23	1.72E+01	1.72E+01	7.71E+00	7.71E+00	2.57E+01	1.46E+01	1.59E+01
4-EchoMonths	130×10	2.23E+01	2.23E+01	1.39E+01	1.39E+01	2.30E+01	1.22E+01	1.13E+01
5-AutoPrice	159×16	9.61E+03	9.61E+03	4.77E+03	4.77E+03	7.69E+03	7.88E+03	4.75E+03
6-Lowbwt	189×10	5.33E+02	5.33E+02	4.29E+02	4.29E+02	4.29E+02	7.62E+02	3.08E+02
7-Pharynx	195×12	4.52E+02	4.52E+02	3.38E+02	3.38E+02	3.43E+02	4.21E+02	2.61E+02
8-PwLinear	200×11	8.53E+00	8.53E+00	2.86E+00	2.86E+00	3.30E+00	3.38E+00	2.48E+00
9-Cpu	209×8	2.20E+02	2.21E+02	9.90E+01	9.90E+01	1.81E+02	2.73E+02	1.35E+02
10-Bodyfat	252×15	1.26E+01	1.10E+01	5.48E+00	5.47E+00	6.57E+00	5.85E+00	5.05E+00
11-BreastTumor	286×10	1.31E+01	1.07E+01	8.27E+00	8.27E+00	8.34E+00	8.06E+00	8.32E+00
12-AutoMpg	398×8	1.30E+01	4.68E+00	6.20E+00	6.20E+00	2.25E+01	6.03E+00	4.40E+00
13-Pbc	418×19	1.45E+03	1.26E+03	9.03E+02	9.17E+02	1.98E+03	1.01E+03	5.47E+02
Average Rank		5.77	5.31	2.54	2.62	5.23	3.85	1.92

3.4.4 Statistical test on data streams

To validate that the effectiveness of our method is significant, we introduce Friedman test and its Post-hoc test after Conover to test whether the MAEs of the above-mentioned methods differ significantly (Pohlert, 2014). The χ_R^2 statistic for Friedman test is computed as

$$(3.22) \quad \chi_R^2 = \frac{12}{nM(M+1)} \sum_{m=1}^M R_m^2 - 3n(M+1),$$

where M is the number of dependent treatment groups which is the number of methods in our case, n is the number of blocks, which is the number of data streams, R_i^2 is the squared rank sum of the i -th group.² If the null hypothesis of the Friedman test is rejected, we use a post-hoc test after Conover for pairwise comparisons. The absolute difference between two group rank sums is significantly different if the following inequality is satisfied:

$$(3.23) \quad |R_i - R_j| > t_{1-\frac{\alpha}{2};(n-1)(M-1)} \times \sqrt{\frac{2M(1 - \frac{\chi_R^2}{n(M-1)})(\sum_{i=1}^n \sum_{m=1}^M R_{i,m}^2 - \frac{nM(M+1)^2}{4})}{(M-1)(n-1)}}.$$

The result of Friedman test and its post-hoc test are given in Table 3.5 and Table 3.6³ where “Friedman Test” is the result for Friedman test and “Friedman

²Please note that it has been wrongly written by R_i in (Pohlert, 2014) if you refer to this citation.

³In Table 3.5 and Table 3.6, “+”, “*”, “***”, and “****” means this value is significant at the level of 0.1, 0.05, 0.01 and 0.001 respectively. “df” denotes the freedom degree.

- post-hoc test after Conover" is for the pairwise comparison. In the Friedman test, χ_R^2 is the computed Friedman statistics based on the MAE values in Table 3.3, and its significance is given by the value of the P-value of χ_R^2 . In the post-hoc test part, $R_i - R_{\text{FUZZ-CARE}^3}$ is the difference between rank sums of other methods and FUZZ-CARE³, and the P-value tests the significance of this difference.

Result⁴:

1) In general, the prediction accuracy of the various methods is different and FUZZ-CARE³ is better than ORTO, FIMT-DD, Per, Kmeans and FUZZ-CARE⁰ over all data streams. However, there is no significant difference between FUZZ-CARE³ and metaAMR or AMR if their performance is evaluated over all data streams. We conclude this because the P-value of χ_R^2 is significant and the pairwise comparisons between methods are consistently significant except for metaAMR and AMR column in Table 3.5.

2) FUZZ-CARE³ is better than all methods over data streams with reoccurring concept. The results in Table 3.6 are computed in the same way as in Table 3.5; the sole difference is that we only use the MAEs of House, Sensor20, Sensor46, SMEAR and Solar. It can be seen that all the statistics are significant, which further validates that our method is especially suitable for data streams with reoccurring concept.

⁴The probability of statistics are computed by *chi2cdf* function and *tcd* function in MATLAB R2016b.

Table 3.5: Friedman test and its post-hoc test of all the methods over all eight data streams.

Friedman Test			χ_R^2	P-value of χ_R^2			df
			25.8333	0.0005***			7
Friedman - post-hoc test after Conover	ORTO	FIMT-DD	metaAMR	AMR	Per	Kmeans	FUZZ-CARE ⁰
$R_i - R_{\text{FUZZ-CARE}^3}$	36	13	4	3	11	23	30
P-value	0.0000***	0.0486*	0.3026	0.3490	0.0794 ⁺	0.0022**	0.0001***

80

Table 3.6: Friedman test and its post-hoc test of all the methods over House, Sensor20, Sensor46, SMEAR and Solar data streams.

Friedman Test			χ_R^2	P-value of χ_R^2			df
			19.6000	0.0065**			7
Friedman - post-hoc test after Conover	ORTO	FIMT-DD	metaAMR	AMR	Per	Kmeans	FUZZ-CARE ⁰
$R_i - R_{\text{FUZZ-CARE}^3}$	30	19	10	11	20	20	22
P-value	0.0001**	0.0071**	0.0898 ⁺	0.0706 ⁺	0.0051**	0.0051**	0.0026**

3.4.5 Parameter analysis and computation complexity

Figure 3.7 illustrates how the accuracy will change when $\alpha, \lambda_1, \lambda_2$ have different values. We take three groups of experiments which validate the sensitivity of α, λ_1 and λ_2 , represented in three subplots. Each line in the subplot represents the results for a data stream denoting changes in accuracy when the parameter differs. We put the MAE results of all the data streams in the interval $[0, 13]$, and the legend gives the magnitude of each stream. For example, House(E+04) means that MAEs at a specified parameter for this data stream are the values on the y-axis $\times 10^4$.

We tested α from 0.005 to 0.015, λ_1 from 0.5 to 1.5, and λ_2 from 0.1 to 1.1. It can be seen that when λ_1 and λ_2 fluctuate around our pre-assigned value (1 for λ_1 and 0.5 for λ_2), the accuracy does not change significantly. Therefore, $\lambda_1 = 1$ and $\lambda_2 = 0.5$ is suggested for the other applications if no other related prior knowledge is provided. They can also be set by choosing an appropriate combination with less cross-validation error, which is a general way to determine pre-assigned parameters. The accuracy is also stable in the α case, except for the data streams SMEAR and Solar. the accuracy for Solar increases as α increases, therefore $\alpha = 0.01$ could be larger to get better performance. In the case of SMEAR, we found that accuracy is low when $\alpha \leq 0.01$, but very high and unstable when $\alpha > 0.01$ (the accuracy for SMEAR when $\alpha > 0.01$ is not given here because it is too high to be contained in the same plot). Therefore, we recommend a small learning rate for SMEAR.

The computation complexity of FUZZ-CARE in each learning process is determined by three factors, i.e., the length of the training data N , the cluster

number k and the iterations I . At each iteration in Algorithm 3.1, the complexity of iterating the membership matrix of μ_{tk} and the cluster center \mathbf{C}_k is bounded by $O(kN)$; given μ_{tk} and \mathbf{C}_k , θ'_k can be directly computed by (3.11). The max iteration is not always used, so the complexity for each learning process is upper bounded by $O(N \times I + k + k)$. To select the optimal clustering number from 2 to K , the computation complexity is upper bounded by $\sum_{k=2}^K O(NI + 2k)$. As how many patterns in a data stream is assumed to be unknown from the aspect of the learning process because the length of a data stream is supposed to be infinite which means there will always come new data instances in the future. The predictor with a larger K is more likely to contains all potential patterns but it takes more time to find the optimal K_0 . Therefore, if the optimal K_0 is far less than K , K should be adjusted to be smaller and K should be larger otherwise.

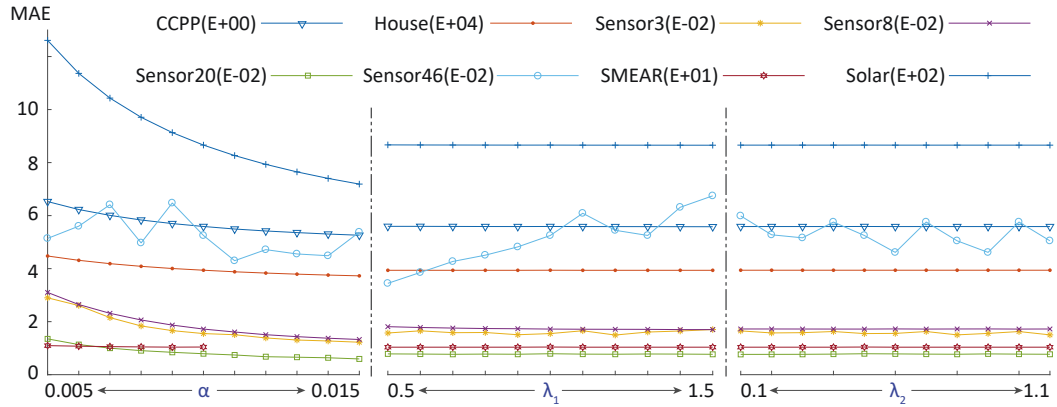


Figure 3.7: Parameter analysis of FUZZ-CARE.

3.5 Summary

This chapter analyzes the concept drift problem by separating drift into *permanent* drift and *alternate* drift, and proposing that they should be tackled with different approaches. A drift adaptation method called FUZZ-CARE is presented to solve solve the concept drift problem especially mixed drift.

FUZZ-CARE simultaneously and dynamically identifies and learns existing patterns, and makes predictions based on these patterns for upcoming data. Rather than training a general predictor for multi-pattern mixed data, we propose to first identify patterns and then treat each pattern individually. However, it is unknown how many patterns exist in the data and what they are. We therefore embed a fuzzy clustering technique in the learning procedure to identify and learn all patterns at the same time. The prediction is made based on the learned patterns and the membership matrix from the fuzzy clustering results. Kernel functions are introduced to enhance FUZZ-CARE for high dimensional situations.

FUZZ-CARE is validated in experiments on synthetic data streams with all types of concept drift, real data streams and stationary datasets. The results show that FUZZ-CARE is able to effectively handle the mixed concept drift problem in data streams, and that it achieves outstanding results in cases of reoccurring concept. In addition, FUZZ-CARE is also suitable to predict stationary data.

The main potential drawback of FUZZ-CARE is that it is computationally costly when a large number of data instances are included in the training set. This disadvantage could be alleviated by deleting redundant instances from the training set. Therefore, how to identify the redundant instance is an interesting

direction in the future. As FUZZ-CARE is a cluster-based approach and outliers have a high interference effect on the clustering results, it is meaningful to embed a denoising technique to improve the accuracy and stability of the clustering process. However, designing a suitable denoising technique for data streams with concept drift is a very challenging problem, because when only a few data instances of new patterns arrive, they are very likely to be wrongly identified as outliers. Excluding useful information in these data instances from the training set inhibits the model from swiftly responding to the new patterns. A noise identification method is important for data streams with concept drift to avoid identifying data following new patterns as noise. In Chapter 6, a noise-tolerant edition of FUZZ-CARE is presented.

DRIFT ADAPTATION BY SEQUENTIALLY UPDATED DATA SEGMENTS

When a data stream contains concept drift, a drift adaptation method needs to update predictors so that the updated predictor can always present the newest data pattern. Using “reliable” data to learn a data-driven predictor is critical to guarantee the performance of the predictor. To select the best data for learning predictors, this chapter presents an adaptation method that can sequentially pick out the best data segments from the training data to learn the predictor. In this chapter, Section 4.1 briefly introduces the problem. Definitions and notations are listed and explained in Section 4.2. Section 4.3 details the proposed segment-based adaptation method, and it will be validated by comprehensive experiments in Section 4.4. Section 4.5 concludes this chapter.

4.1 Introduction

The informed drift adaptation methods propose to update the predictor only if drift is detected. Existing informed drift adaptation methods need to wait for an entire batch (time window) of data to detect drift and then update the predictor (if drift is detected), which causes adaptation delay. The existing informed adaptation methods only use the result of drift detection process to assist adaptation (Lu et al., 2016). They need a batch of data to determine whether drift exactly occurs (Boracchi et al., 2018; Lu et al., 2018), which results in the drift reaction delay.

This chapter demonstrates how we can overcome the adaptation delay and select the most relevant data to the new pattern. we propose a sequentially updated statistic, called *drift-gradient*, and develop a segment-based drift adaptation(SEGA) method to update our best predictor when every new instance arrives. During the learning process, the training data is ordered by time and divided into several equal length segments, and a predictor is learned with the data in each segment. When a new instance arrives, we first update the drift-gradient on each segment in the training data. Based on the updated drift-gradient, SEGA retrains our best predictors with the segments that have the minimum drift-gradient.

Drift-gradient is to quantify the *increase of segmented symmetric degree* (SSD) when *only one* new instance is available at each time step. SSD is a new statistic proposed to measure the distributional discrepancy between an old segment and the newest segment in the training set. The advantage of drift-gradient is that it does *not* need to compute the value of SSD before and after the new instance arrives, and therefore can be sequentially updated with low computational cost.

A lower value of drift-gradient on an old segment represents that the distribution of the new instance *is closer* to the distribution of this old segment. Therefore, the predictor trained with this segment should be a better predictor for the new instance.

SEGA has been validated by extensive experiments on both synthetic and real-world, classification and regression data streams. To our best knowledge, this is the first time that the drift adaptation method has been comprehensively tested on both continuous and discrete label variables. SEGA has been compared to 15 benchmarks (where 7 of them are adaptation methods for regression task, and 8 are for classification task) and validated on 16 synthetic data (where 6 of them are for the regression task and 10 are for the classification task) and 14 real-world data streams (where 7 of them are for the regression task and 7 are for the classification task). The experimental results show that SEGA outperforms competitive blind and informed drift adaptation methods.

Compared to other drift adaptation methods, SEGA demonstrates the following advantages.

- A *segmented symmetric degree* (SSD) is proposed to measure distributional discrepancy between old segments and the newest segment in the training set. SSD is better than a one-sided measurement when two distributions have different variances;
- A sequentially updated statistic, *drift-gradient* is proposed to quantify the increase of SSD when every new instance arrives *without* computing the value of SSD before and after the new instance arrives;
- An *online drift adaptation* method, SEGA is developed based on drift-

gradient. SEGA can effectively overcome the adaptation delay issue in the existing informed drift adaptation methods.

4.2 Definitions and Notations

The definitions of data stream, concept drift and the learning aim are the same to the definitions in Section 3.2.

In this chapter, we use the following notations.

- P, P_1, P_2, Q : data samples
- $N_P, N_{P_1}, N_{P_2}, N_Q$: sample size
- p, p_1, p_2, q : probability distributions
- $|K_{i,S}(k)|$: the number of the k nearest neighbors of an instance i that are from the sample S

To compute $|K_{i,S}(k)|$, we use a KNN search to determine the neighbors for each instance. For both regression and classification task, we first normalize each variable (including the target variable) into $[0, 1]$ by using the min-max normalization. After that, the distance is computed by the Euclidean distance. However, using Euclidean distance denotes that we assume $(\mathbf{X}_t, y_t) \in \mathbb{R}^{d+1}$ which is not true in real application, especially the classification task. The KNN could also be determined by other distance, which can further improve the method.

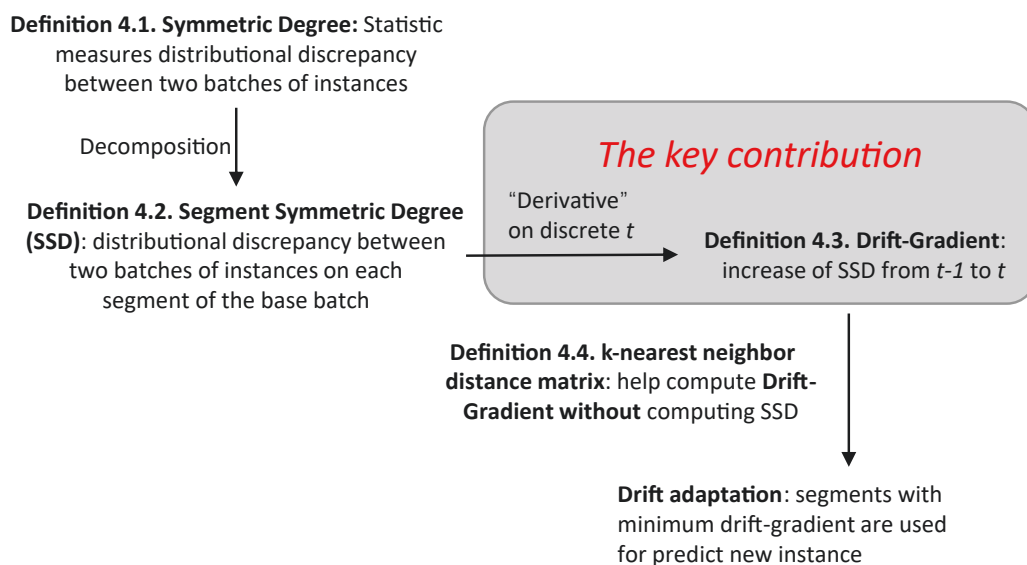


Figure 4.1: Relationship between the definitions proposed in this chapter.

4.3 A Segment-based Drift Adaptation

Method—SEGA

The core technique in SEGA is the sequentially-updated statistic, called *drift-gradient* and based on it, a *segment-based adaptation*(SEGA) method is developed to update the predictor when every new instance arrives. Drift-gradient is to quantify the increase of *segmented symmetric degree* (SSD) when *only one* new instance is available at each time point. SSD is a new statistic that can measure the distributional discrepancy between old segments and the newest segment. The relationship between the proposed new definitions is presented in Fig. 4.1.

4.3.1 The segmented symmetric degree (SSD)

SSD is the decomposition of a symmetric degree (SD) on segments. SD defined in Definition 4.1 is to measure the distributional discrepancy between two samples of data.

Definition 4.1 (Symmetric Degree). Given two data samples P and Q with p and q its distribution, the symmetric degree $\bar{d}_{P,Q}(k)$ is defined as the average density difference of P 's and Q 's k -nearest neighbors.

$$(4.1) \quad \bar{d}_{P,Q}(k) = \frac{1}{N_P} \sum_{u \in P} \left(\frac{|K_{u,P}(k)|}{N_P} - \frac{|K_{u,Q}(k)|}{N_Q} \right) + \frac{1}{N_Q} \sum_{v \in Q} \left(\frac{|K_{v,Q}(k)|}{N_Q} - \frac{|K_{v,P}(k)|}{N_P} \right),$$

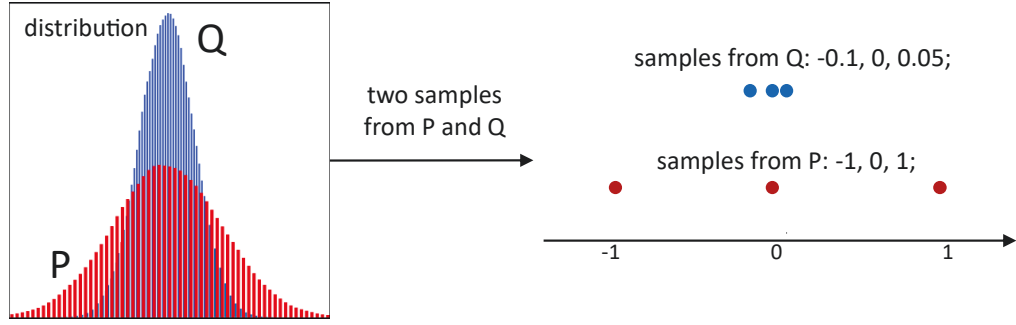
where for any u, v , $|K_{u,P}(k)| + |K_{u,Q}(k)| = k$ and $|K_{v,Q}(k)| + |K_{v,P}(k)| = k$.

If there are more than one candidates for the k th-nearest neighbor of u , the k th-nearest neighbor is randomly chosen from the candidates in P . Similarly for u, v 's k th-nearest neighbor is determined from the candidates in Q if there are more than one candidate. The first term in the summation of Definition 4.1, measures the average density difference over all of P 's neighborhoods and the second term measures the average density difference over Q 's neighborhoods. Clearly, the proposed $\bar{d}_{P,Q}(k)$ is symmetric on these two samples, given the same k , namely $\bar{d}_{P,Q}(k) = \bar{d}_{Q,P}(k)$. A larger absolute value of SD, means that P is more likely to be different from Q . As SD is a summation-based statistic, it converges to a normal distribution according to the central limit theorem, when N_P and N_Q approaches infinity. Similar proof can be found in our previous studies (Liu et al., 2017a, 2018a).

Next, a simple example in Figure 4.2 is introduced to show why the symmetric measurement is better than the one-side measurement. The one-sided

measurement only counts the density difference on either P or Q 's neighborhoods such as the measurement in (Liu et al., 2017a, 2018a). In Figure 4.2, the red dots represents sample P and the blue dots represents sample Q . According to the distribution of P and Q , they have the same expectation but P has larger variance. If the density difference is restricted to P 's neighborhoods, as shown in subplot (a). When $k = 2$, the neighborhoods of P is framed by the black dotted ellipse and there is no density difference in any ellipse. If the density difference is restricted to Q 's neighborhoods in subplot (b), the neighborhoods of Q is framed up by three ellipses. In the black ellipse, there is no density difference between the blue and red dots. However, in the red ellipses, they all contain blue dots. The density of the blue dots is 1 and the density of the red dots is 0, giving a density difference of 1. Therefore, if a one-sided measurement of density difference is applied based on the sample that has larger variance, such as is in subplot (a), it is invalid to reflect the variance discrepancy between two samples.

Given SD able to measure the difference between two samples, let the current training data be sample P and the newest batch of data instances be the sample Q , where $N_P \gg N_Q$, SD can be used to test whether Q 's distribution is different from P . SSD is the decomposition of SD which can consider the distributional discrepancy on each segment of the training data. We firstly explain the decomposition of SD in the case of two segments. Given P_1 and P_2 two absolute complements in P that $P_1 \cup P_2 = P$ and $P_1 \cap P_2 = \emptyset$, SD can be rewritten as (4.2). Detailed derivation from SD to SSD can be found in Appendix A.1.



*If using a **one-sided** measurement d to measure distributional discrepancy, $d(P, Q) \neq d(Q, P)$, as is presented in a) and b)

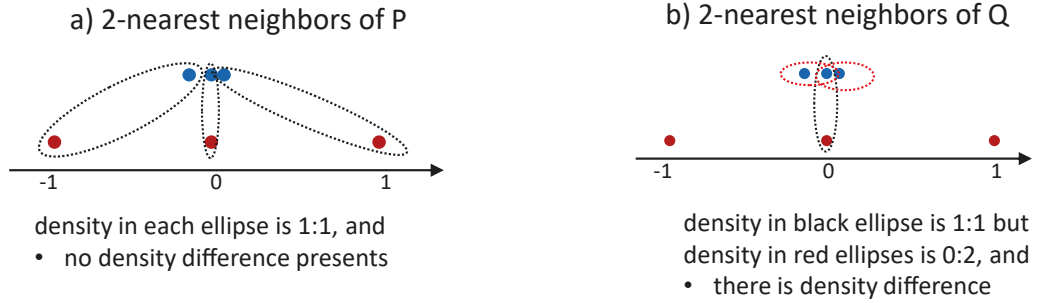


Figure 4.2: An example to show the drawback of a one-sided measurement.

$$\begin{aligned}
 \bar{d}_{P,Q}(k) &= \frac{1}{N_P} \sum_{u \in P_1} \left(\frac{|K_{u,P}(k)|}{N_P} - \frac{|K_{u,Q}(k)|}{N_Q} \right) + \frac{1}{N_P} \sum_{u \in P_2} \left(\frac{|K_{u,P}|}{N_P} - \frac{|K_{u,Q}(k)|}{N_Q} \right) + \\
 (4.2) \quad & \frac{1}{N_Q} \sum_{v \in Q} \left(\frac{|K_{v,Q}(k)|}{N_Q} - \frac{|K_{v,P}(k)|}{N_P} \right).
 \end{aligned}$$

Given $ssd_{P_1,Q}(k)$ and $ssd_{P_2,Q}(k)$ in (4.3) and (4.4) separately, $\bar{d}_{P,Q}(k) = ssd_{P_1,Q}(k) + ssd_{P_2,Q}(k)$.

$$\begin{aligned}
 (4.3) \quad \text{ssd}_{P_1, Q}(k) &= \frac{1}{N_P} \sum_{u \in P_1} \left(\frac{k - |K_{u, Q}(k)|}{N_P} - \frac{|K_{u, Q}(k)|}{N_Q} \right) - \\
 &\quad \frac{1}{N_Q} \sum_{v \in Q} \frac{|K_{v, P_1}(k)|}{N_P} + \frac{1}{2N_Q} \sum_{v \in Q} \frac{|K_{v, Q}(k)|}{N_Q}.
 \end{aligned}$$

$$\begin{aligned}
 (4.4) \quad \text{ssd}_{P_2, Q}(k) &= \frac{1}{N_P} \sum_{u \in P_2} \left(\frac{k - |K_{u, Q}(k)|}{N_P} - \frac{|K_{u, Q}(k)|}{N_Q} \right) - \\
 &\quad \frac{1}{N_Q} \sum_{v \in Q} \frac{|K_{v, P_2}(k)|}{N_P} + \frac{1}{2N_Q} \sum_{v \in Q} \frac{|K_{v, Q}(k)|}{N_Q}.
 \end{aligned}$$

Similar to the case of two segments, SD can be decomposed into more than two segments.

Definition 4.2 (Segmented Symmetric Degree). The segmented symmetric degree is defined as the distributional discrepancy between the i th segment of P to Q that

$$\begin{aligned}
 (4.5) \quad \text{ssd}_{P_i, Q}(k) &= \frac{1}{N_P N_Q} \left\{ \sum_{u \in P_i} \left(-\frac{N_Q}{N_P} - 1 \right) |K_{u, Q}(k)| - \right. \\
 &\quad \left. \sum_{v \in Q} |K_{v, P_i}(k)| \right\} + \frac{k N_{P_i}}{N_P} + \frac{1}{2N_Q} \sum_{v \in Q} \frac{|K_{v, Q}(k)|}{N_Q}.
 \end{aligned}$$

$ssd_{P_i, Q}$ measures the distributional discrepancy from Q to each subset of P . A smaller $ssd_{P_i, Q}$ indicates a better presentation of Q by P_i .

4.3.2 Drift-gradient and how to sequentially update it

If we wait for an entire batch of data to detect drift and then update the predictor (if drift is detected), which causes adaptation delay. To overcome the adaptation delay, we propose a sequentially updated statistic, called Drift-gradient.

Drift-gradient is to quantify the increase of SSD. We have discussed in the previous subsection that a smaller $ssd_{P_i, Q}$ indicates a better presentation of Q by P_i . Therefore, a lower value of drift-gradient on i th segment represents that the distribution of Q is *closer* to the distribution of the i th segment. In this section, we will explain how to update the drift-gradient with computing the values of SSD.

We notice that SSD for different segments have common items. Therefore, we simplified SSD from two aspects: 1. we consider P is evenly segmented that $N_{P_i} = N_0$; 2. The common items in $ssd_{P_i, Q}$ is excluded to computing drift-gradient as it does not affect the result.

Given P and its segments with $N_{P_i} = N_0$, the common items for all i in $ssd_{P_i, Q}(k)$ is (4.6), leaving the discrepancy part of $ssd_{P_i, Q}$ as is in (4.7). Clearly,

$$ssd_{P_i, Q}(k) = C(k) + \frac{1}{N_P N_Q} \delta_{P_i, Q}(k)$$

$$(4.6) \quad C(k) = \frac{kN_0}{N_P} + \frac{1}{2N_Q} \sum_{v \in Q} \frac{|K_{v, Q}(k)|}{N_Q}$$

$$(4.7) \quad \delta_{P_i, Q}(k) = - \sum_{v \in Q} |K_{v, P_i}(k)| - \left(1 + \frac{N_Q}{N_P}\right) \sum_{u \in P_i} |K_{u, Q}(k)|$$

Compared to Definition 4.2, (4.7) is more concise. The task of quantifying the increase of SSD can be equivalently transferred to the task of quantifying the increase of δ .

Definition 4.3 (Drift-Gradient). Given the simplified SSD at a specific time step denoted by δ , drift-gradient is defined as

$$(4.8) \quad \nabla\delta_i(k, t) = \delta_{P_i, Q_t}(k) - \delta_{P_i, Q_{t-1}}(k)$$

Given P obtained from the past, contains N_P data instances and Q the upcoming batch (batch size: N_Q) of data instances arrives instance by instance, the key to online compute $\nabla\delta$ is to determine the relationship between $\delta_{P_i, Q_t}(k)$ and $\delta_{P_i, Q_{t-1}}(k)$, where Q_t represents the state of Q at time t . Clearly, $Q_t = Q_{t-1} \cup v_t$ where v_t is the t -th data instance. It should be noticed that drift-gradient is not a mathematical gradient. δ is discrete because of $t \in \mathbb{Z}^+$. Therefore, $\nabla\delta$ is not computed by the derivative of a function of a real value. It is called drift-gradient because it denotes the rate of change of δ , which is similar to a gradient denoting the rate of change of a function.

Next, we are going to discuss how to compute drift-gradient when every new instance arrives. As shown in (4.8) and (4.7), drift-gradient is defined as the change of $\delta_{P_i, Q_t}(k)$, and $\delta_{P_i, Q_t}(k)$ mainly consists of $\sum_{v \in Q} |K_{v, P_i}(k)|$ and $\sum_{u \in P_i} |K_{u, Q}(k)|$. If we know how $\sum_{v \in Q} |K_{v, P_i}(k)|$ and $\sum_{u \in P_i} |K_{u, Q}(k)|$ changes from their previous values, we do not need to compute the value of $\delta_{P_i, Q_t}(k)$ and $\delta_{P_i, Q_{t-1}}(k)$ to obtain drift-gradient. Therefore, the sequential update of $\nabla\delta$ contains of two parts when we substitute (4.7) into (4.8): 1) the sequential updates of $\sum_{v \in Q} |K_{v, P_i}(k)|$ denoted by $\nabla\delta_Q$; 2) the sequential updates of $\sum_{u \in P_i} |K_{u, Q}(k)|$ denoted by $\nabla\delta_{P_i}$. Next, these two parts will be discussed separately.

- *Sequentially update* $\sum_{v \in Q} |K_{v, P_i}(k)|$:

Given $\sum_{v \in Q_{t-1}} |K_{v, P_i}(k)|$ represents the number of k -nearest neighbors of current data instances arrived in Q at time point $t-1$, $\sum_{v \in Q_t} |K_{v, P_i}(k)|$ is computed as

$$(4.9) \quad \sum_{v \in Q_t} |K_{v, P_i}(k)| = \sum_{v \in Q_{t-1}} |K_{v, P_i}(k)| + |K_{v_t, P_i}(k)|.$$

Therefore,

$$(4.10) \quad \nabla \delta_Q = |K_{v_t, P_i}(k)|.$$

- *Sequentially update* $\sum_{u \in P_i} |K_{u, Q}(k)|$:

$\sum_{u \in P_i} |K_{u, Q}(k)|$ can not be directly iterated as the case in $\sum_{v \in Q} |K_{v, P_i}(k)|$. In order to implement the update, k -nearest neighbor distance matrix is defined as Definition 4.4.

Definition 4.4 (k -nearest neighbor distance matrix). Given a $m \times n$ dimension matrix $A = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$, its k -nearest neighbor distance matrix is defined as

$$(4.11) \quad D(k) = \begin{bmatrix} d_{(1)}^1 & d_{(2)}^1 & \cdots & d_{(k)}^1 \\ d_{(1)}^2 & d_{(2)}^2 & \cdots & d_{(k)}^2 \\ \vdots & & \ddots & \vdots \\ d_{(1)}^n & d_{(2)}^n & \cdots & d_{(k)}^n \end{bmatrix},$$

where $d_{(k)}^j, (j = 1, \dots, n)$ is the k th order of distance from \mathbf{a}_j to A . The first column of $D(k)$ is $\mathbf{0}$ because $d_{(1)}^j$ is always the distance from \mathbf{a}_j to itself. For

example, $A = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) = \begin{bmatrix} 1 & 0 & 5 \\ 3 & 2 & 1 \end{bmatrix}$, its distance matrix is $\begin{bmatrix} 0 & \sqrt{2} & \sqrt{20} \\ \sqrt{2} & 0 & \sqrt{26} \\ \sqrt{20} & \sqrt{26} & 0 \end{bmatrix}$,

and corresponding 3-nearest neighbor distance matrix $D(3) = \begin{bmatrix} 0 & \sqrt{2} & \sqrt{20} \\ 0 & \sqrt{2} & \sqrt{26} \\ 0 & \sqrt{20} & \sqrt{26} \end{bmatrix}$.

Similarly, $D(2) = \begin{bmatrix} 0 & \sqrt{2} \\ 0 & \sqrt{2} \\ 0 & \sqrt{20} \end{bmatrix}$.

Given $\mathbf{a}_j = (\mathbf{X}_j, y_j)^T$ and $P_i = (\mathbf{a}_1, \dots, \mathbf{a}_{N_{P_i}})$ where $j \in (1, \dots, N_{P_i})$, its k -nearest neighbor matrix $D_i(k)$ can be computed by (4.11), denoted by $D_i(k) = (\mathbf{d}^1, \mathbf{d}^2, \dots, \mathbf{d}^{N_{P_i}})^T$. Given $D_i(k)$ and $\mathbf{Q} = (v_{t_0+1}, v_{t_0+2}, \dots, v_{t_0+N_Q})$ arriving by instance after t_0 , updating $\sum_{u \in P_i} |K_{u, \mathbf{Q}}(k)|$ is to compare $D_i(k)$ and the distance from v_t to P_i — $d(v_t, P_i)$, as well as $d(v_t, P_i)$ and its previous values $d(v_{t-1}, P_i), d(v_{t-2}, P_i), \dots$. Next, the updating process for an arbitrary $u \in P_i$ will be discussed.

Given P_i and its k -nearest neighbor distance matrix $D_i(k)$, for an arbitrary $u \in P_i$, the ordered k -nearest neighbor distance of u is one row of $D_i(k)$, denoted as \mathbf{d}^u . $d_{v_t, u}$ represents the distance between u and the newly arrived instance v_t from \mathbf{Q} . The one-step (one-instance) updating process of $|K_{u, \mathbf{Q}_t}(k)|$ is in Algorithm 4.1 where the updated variable are denoted with t . K_d counts how many items in \mathbf{d}^u are larger than $d_{v_t, u}$. $K_d \leq 1$, which means the newly arrived v_t is no nearer

to u than u 's current nearest neighbors, and therefore $|K_{u,Q_t(k)}| = |K_{u,Q_{t-1}(k)}|$. Once $K_d > 1$, it also needs to consider whether a previous v is excluded from u 's k -nearest neighbors, as v_t becomes u 's k nearest neighbor. To implement that, the variable o_t is introduced to record the order of $d_{v_t,u}$ and the furthest neighbor in \mathbf{d}^u will be deleted from \mathbf{d}^u (implemented in line 8). By this design, when the length of current \mathbf{d}^u is less than $\max \mathbf{O}_{Q_t}$, this denotes that the previous $d_{v,u}$ are still less than the furthest neighbor from P_i . Therefore, $|K_{u,Q_t(k)}| = |K_{u,Q_{t-1}(k)}| + 1$. Otherwise, $|K_{u,Q_t(k)}|$ will not update, because although v_t becomes one of u 's k -nearest neighbors, a previous v is excluded from the neighbors at the same time. An intuitive perception of this design is that δ_Q is only updated when the threshold for v becoming u 's k -nearest neighbors is leveled up.

Given $\sum_{u \in P_i} |K_{u,Q_{t-1}}(k)|$, $\sum_{u \in P_i} |K_{u,Q_t}(k)|$ can be updated by the sum of $|K_{u,Q_t}(k)|$ and $\nabla \delta_{P_i} = \sum_{u \in P_i} |K_{u,Q_t}(k)| - |K_{u,Q_{t-1}}(k)|$. When a new instance from Q arrives, the drift-gradient on each segment in P is computed as $\nabla \delta_i = \nabla \delta_Q + \nabla \delta_{P_i}$.

To validate the effectiveness of the proposed drift-gradient, we conduct experiments of whether the drift-gradient can correct identify the most appropriate segment in the training set. We generate 50-dimensional instances from three different uniform distributions denoted by P1, P2 and P3. P1 is generated by `numpy.random.seed(1)` in python, P2 is generated with the mean 0.4 larger than P1, and P3 is generated with the mean 0.4 larger than P2. The training set consists of three segments, which are shown in Table 4.1. Then we generate 200 instances from P3, and 200 instances from P2 as the testing set, and they arrive one instance by one instance. In Experiment1, P3 arrives before P2, while in Experiment2 P2 arrives before P3. In Table 4.2, the Type I error of P1 is how many true P1 instances are correctly labelled by P1 on average. As there are no

Algorithm 4.1: One-step update of $|K_{u,Q_t}(k)|$

Input : $d^u, d_{v_t,u}, \mathbf{O}_{Q_{t-1}}, |K_{u,Q_{t-1}}(k)|$.
Output : $|K_{u,Q_t}(k)|$
Initialization: $\mathbf{O}_{Q_0} \leftarrow []$

- 1 **Compute** $K_d = |d^u > d_{v_t,u}|;$
- 2 **if** $K_d > 1$ **then**
 - 3 *% the new instance is nearer to u than u's current neighbors*
 - 4 **Compute** $o_t = |d^u| + 1 - K_d;$
 - 5 **Compute** $\mathbf{O}_{Q_t} = [\mathbf{O}_{Q_{t-1}}; o_t]$
 - 6 **if** $|d^u| \leq \max \mathbf{O}_{Q_t}$ **then**
 - 7 **Update** $|K_{u,Q_t}(k)| = |K_{u,Q_{t-1}}(k)| + 1;$
 - 8 **Delete** d^u [end];
 - 9 **else**
 - 10 *% a previous v is excluded from u's neighbors*
 - 11 **Update** $|K_{u,Q_t}(k)| = |K_{u,Q_{t-1}}(k)|;$
 - 12 **end**
- 13 **else**
 - 14 *% the new instance is no nearer to u than u's current neighbors*
 - 15 **Update** $|K_{u,Q_t}(k)| = |K_{u,Q_{t-1}}(k)|;$
- 16 **end**
- 17 **return** $|K_{u,Q_t}(k)|, \mathbf{O}_{Q_t}$

instances from P1 in the new instances, this value is 0. The Type II error of P1 is how many true P2 and P3 instances are wrongly labelled by P1 on average. As there are 10 P2 instances wrongly labelled by P1 in Experiment1 in the new instances, this value is 10/400. Similar computation is conducted on P2 and P3. The Type I and Type II error shows drift-gradient is able to select the segment that is closer to the new instances.

4.3.3 The general procedure and pseudocode of SEGA

The previous subsection solved how to sequentially update drift-gradient. Based on this solution, an online drift adaptation method, SEGA is presented in this

Table 4.1: Validation of the effectiveness of drift-gradient

Experiment1	segment1	segment2	segment3	new instance arrives one by one					
numbers of instances	400	400	400	200			200		
true distribution	P1	P2	P3	P3			P2		
lablled by drift-gradient	-	-	-	P1	P2	P3	P1	P2	P3
numbers of instances	-	-	-	0	2	197	10	173	17
Experiment2									
true distribution	P1	P2	P3	P2			P3		
lablled by drift-gradient	-	-	-	P1	P2	P3	P1	P2	P3
numbers of instances	-	-	-	3	183	14	0	11	189

Table 4.2: Type I and Type II error of drift-gradient under three distributions

	Experiment1		Experiment2		Average	
	Type I	Type II	Type I	Type II	Type I	Type II
P1	0	10/400	0	3/400	0	0.01625
P2	1-173/200	2/200	1-183/200	11/200	0.11	0.0325
P3	1-197/200	17/200	1-189/200	14/200	0.035	0.0775

section.

For a data stream $\{D_t = (\mathbf{X}_t, y_t), t = 1, \dots, \infty\}$ with \mathbf{X} its attributes and y the corresponding label, $\{D_t, t \leq T_0\}$ is assumed to be already obtained as the historical data. The data instances after T_0 will arrive one by one, by time and \mathbf{X}_t is observed before y_t . The online prediction is to first apply the current trained predictor to predict y_t for $t > T_0$ given the value of \mathbf{X}_t . After the true value of y_t is obtained, this newly arrived D_t will be used to update the current predictor.

SEGA uses an update process that combines batch and online adaptation. Before a full batch of new instances arrives, an online adaptation is activated to tune the current predictor, and once a full batch of new instances is obtained, the current predictor will be retrained as is in a batch adaptation. Algorithm 4.2 presents how SEGA update the predictors from \hat{H}_{t-1} to \hat{H}_t . The parameter w denotes the the length of the segments in the training set. s presents the number of segments that each training set contains. During the experiments, the s is

Algorithm 4.2: SEGA

```

Input :  $D_t, \hat{H}_{t-1}(\cdot|\Theta), P, Q, w, c, s.$ 
Output :  $\hat{H}_t, P, Q.$  %  $\hat{H}_t$  is used to predict  $y_{t+1}$ 
1  $Q = [Q; D_t];$ 
2 if  $|Q| < w$  then
3   % sequentially update predictors and store new instance in Q
4   for  $P_i$  in  $P$  do
5     % compute drift gradient for each segment
6     Compute  $\nabla\delta_Q$  by (4.10);
7     for  $u$  in  $P_i$  do
8       Update  $|K_{u, Q_t(k)}|$  by Algorithm 4.1;
9        $\nabla\delta_{P_i} \leftarrow \nabla\delta_{P_i} + (|K_{u, Q_t}(k)| - |K_{u, Q_{t-1}}(k)|)$ 
10    end
11     $\nabla\delta_i = -\nabla\delta_Q - (1 + w/|P|)\nabla\delta_{P_i}$ 
12  end
13   $i_c = \text{argsort } \nabla\delta_i[1 : c];$  % c segments with minimal drift gradient;
14   $\hat{H}_c(\cdot|\Theta) = \frac{1}{c} \sum_{i \in i_c} \hat{H}_i(\cdot|\Theta);$  % combined by average;
15   $\hat{H}_t(\cdot|\Theta) \leftarrow \hat{H}_c(\cdot|\Theta)$ 
16 else
17   % a batch of new data has been stored, initializing the buffer Q
18   Segment  $P$  into  $P_1, \dots, P_s;$ 
19    $P \leftarrow [Q, P_2, \dots, P_s];$  % the training set is updated;
20    $\hat{H}_t(\cdot|\Theta) \leftarrow \hat{H}_1(\cdot|\Theta);$ 
21    $Q = [ ];$  % initialize Q
22 end

```

assigned a fixed size and a period of data instances of $s \times w$ will be picked from the historical data as the initial training data. c is an ensemble coefficient to control how many segments are picked. The row 7-10 updates δ_P by $\nabla\delta_P$ can be skipped for a faster computation with slightly lower accuracy.

The flowchart of SEGA is shown as Figure 4.3. The training set will be separated into disjointed segments and the drift-gradient is computed to dynamically select the best segments for prediction. If the whole training set is used to train the predictor and retrain every buffer or every instance, it is a sliding window

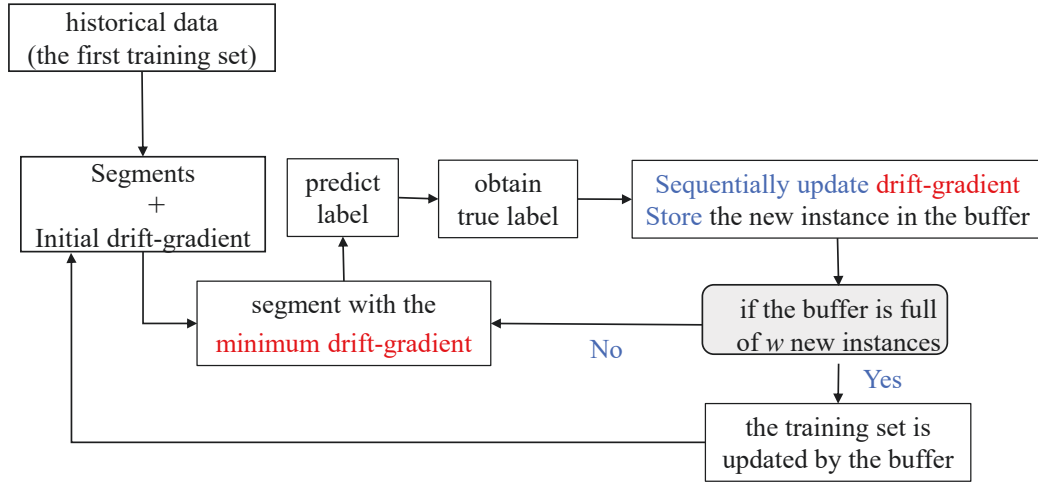


Figure 4.3: Flowchart of SEGA.

adaptation. In the experiment section, the sliding window adaptation will be compared as a baseline method, to show the necessity of segments and the effectiveness of drift-gradient for handling the concept drift problem.

4.4 Experimental Evaluations

In this section, SEGA is compared to 15 baselines on 16 synthetic data and 15 real-world data streams. The **experimental results and analysis** are given in Section 4.4.1 and 4.4.2. As the predictors used for SEGA in tested data streams are different for regression and classification tasks, the **experimental configuration** will be introduced and specified at the beginning of each subsection. **Friedman test** of comparison between SEGA and other baseline methods are conducted in Section 4.4.3. The **parameter analysis and computation complexity** are presented in Section 4.4.4.

Four kinds of data streams are involved: synthetic regression data, synthetic

classification data, real-world regression data and real-world classification data. To our best knowledge, this is the first time that the concept drift problem has been comprehensively tested on both continuous and discrete label variables. It is not known if a data stream contains drift. The types of drift that exists in a real-world data stream is also unknown, Therefore, drift is manually added into the synthetic data streams. Experiments on the synthetic data streams are to validate that SEGA can solve concept drift problems. In the experiments on synthetic data, all types of drift have been evolved. Details of drift types will be introduced in each subsection of experiments.

Data streams are supposed to be infinite but to validate the algorithm and present its effectiveness, it is essential to obtain a finite period of data streams. A common way to evaluate the algorithm effectiveness on data streams is prequential evaluation, where each data instance is first used to test the predictor, and then to train the predictor. A fixed length of historical data instances are available before conducting the experiments. Future data instances are available one by one during the experimental procedure. In the experiments, we consider prediction accuracy as the validation criterion including *mean absolute error* (3.21) in regression tasks and *accuracy* in classification tasks (4.12).

$$(4.12) \quad \text{Acc} = \frac{TP + TN}{TP + TN + FP + FN},$$

We have conducted comprehensive experiments to validate the effectiveness of SEGA to handle the concept drift problem. The experimental design is as shown in Table 4.3.

Table 4.3: Experimental design in this section.

Section	Data	Experimental aim	Main Results
4.4.1.1	d1	to validate SEGA can pick out best segments for learning and handle the concept drift in regression tasks	Table 4.4
4.4.1.2	d2	the aim is the same as above but for classification cases	Table 4.5
4.4.2.1	d3	to demonstrate that SEGA can also used for real-time regression	Table 4.6
4.4.2.2	d4	the aim is the same as above but for classification cases	Table 4.6
4.4.3	d3, d4	to demonstrate that SEGA has significant advantages when handling real-world concept drift problems	Table 4.8, Table 4.9
4.4.4	d3, d4	to demonstrate the robustness of SEGA	Figure 4.4, Figure 4.5

d1: synthetic regression data

d2: synthetic classification data

d3: real-world regression data streams

d4: real-world classification data streams

4.4.1 Evaluation on synthetic data

In this section, SEGA will be evaluated on six synthetic data of regression tasks and ten synthetic data of classification tasks. The aim of the experiments on synthetic data is to validate that SEGA is effective to handle drift problem. Therefore, SEGA will be compared to its corresponding non-adaptation edition on data streams containing different types of drift, including no drift. As discussed in Section 4.3.3, SEGA will also be compared to its sliding window edition to validate the effectiveness of segment.

4.4.1.1 Regression Tasks

Data Description. The synthetic data of regression data is the same as the synthetic data in Section 3.4.1, which includes one original data stream that does not contain drift, one data stream containing virtual drift and four data streams with real drift. Since the drifted data is generated based on the non-drifted data, it is clear to understand the extent that SEGA solves the problem of concept drift in a data stream.

Configuration. In the regression tasks, the predictor in SEGA is a linear predictor with $L2$ -norm (ridge regression) where $\alpha = 0.01$, each segment is of 100 instances, the training set contains 10 segments (namely, the length of training set is 1000), and the ensemble coefficient $c = 1$.

Analysis of Results on Synthetic Regression Tasks. The experimental results are presented in Table 4.4. The column, Data Description, lists the length of each data and the type of drift it contains. Prediction accuracy is shown in the Tested Models column. Linear column denotes a ridge regression predictor that's

trained on the first training set and is used for all the testing data without retrain. The SlidWin column presents the prediction results of a sliding window edition. In the sliding window edition, the training set will be updated by including the newly arrived instance and discarding the oldest instance. For each newly arrived instance, a new predictor will be trained on the current training set, to predict the label for the next instance. SlidWin can partly solve the problem of concept drift, as it obtain better prediction accuracy than Linear on some data streams with real drift. Our method, displayed in the SEGA-Linear column, uses a drift-gradient to choose the best segment for prediction to adapt to new concept. Model Effectiveness column evaluates SEGA to determine how much SEGA differs from an ideal edition, where the drift-gradient correctly designates the best segment for each tested instance. The MAE in the SEGA-Ideal column is computed as follows:

- Predict the label by the trained predictor on each segment. As the segment number is 10, there are 10 predicted values for each test instance.
- Given the true value of the label, choose the best result from 10 predictions with the minimum absolute error for the prediction of this instance.
- Compute MAE. This is the minimum MAE that SEGA could obtain, because it is computed with the true label.

The Effectiveness column is computed by MAE of SEGA-Linear and SEGA-Ideal that

$$(4.13) \quad \text{Effectiveness} = \frac{MAE_{SEGA-Linear}}{MAE_{SEGA-Ideal}}.$$

SEGA-Linear is more applicable on the data with a larger value of the effectiveness.

The MAE results in Table 4.4 shows that:

1) *Linear, SlidWin and SEGA-Linear performs the same when data contains no drift or virtual drift.* The synthetic data in Table 4.4 are generated by a linear function with a random error, and a linear predictor is used during the experiments. Therefore, the experimental results are less affected by the cause of an inappropriate predictor. If the data is generated by a very complex function, and it has to use a powerful predictor trained on a large size of instances to accurately present the true hypothesis, the effectiveness of solving concept drift will highly be impaired by the poor performance of the predictor trained on a limited size of instances. In the Non-Drift data stream, it can be seen that the prediction accuracy of Linear, SlidWin and SEGA-Linear are all close to 0.800, which shows that for this data stream, if no drift occurs, there is no difference between learning a predictor on the full training set and on one of its segments. This is also one of the required applicable conditions when choosing the appropriate predictor in SEGA. It has been discussed in (Song et al., 2019a) that virtual drift has little influence on the prediction results if the change of $p(\mathbf{X})$ is independent to $p(y|\mathbf{X})$. Here, the same results show —Linear, SlidWin and SEGA obtain almost the same accuracy on the Virt-Drift stream.

2) *Simple retraining is not suitable for all types of drift.* By comparing the Linear, SlidWin and SEGA-Linear columns in Table 4.4. SlidWin is to simply retrain the current predictor when every new instance arrives, and does not include any design on how to use drift information to help adaptation. According to the value of MAE of SlidWin, it performs well on most data streams and can

partly handle the concept drift problem under some conditions. For example, the MAEs of SlidWin are much smaller than the corresponding MAEs of Linear on Sudd-Drift, Incr-Drift and Rec-Drift-Mix. Specifically, SlidWin is as good as SEGA on Sudd-Drift and Incr-Drift streams. However, SlidWin can not fully solve the reoccurred concept problem. SlidWin performs worse than SEGA on two streams containing reoccurred drift, and even worse than the Linear model on the Rec-Drift-Grad stream.

3) Retraining by segment is an effective way to solve concept drift problem. Our proposed SEGA method can outperform other method when any type of drift or mixture of occurs.

4) The proposed SEGA method can accurately identify the best segment in the training set by drift-gradient. Comparing the column of SEGA-Linear and SEGA-Ideal, it can be seen that when real drift occurs, the prediction accuracy of SEGA-Linear is very close to the accuracy of SEGA-Ideal. As SEGA-Ideal predicts the label based on the true value of that label, it is considered to be the optimal prediction if using one segment of training set to predict labels. According to the Model Effectiveness result, our proposed SEGA can obtain 95.2% prediction capability of a prediction where the true label is known on Sudd-Drift, and a prediction capability of 96.1%, 79.2% and 93.9% on Incr-Drift, Rec-Drift-Grad and Rec-Drift-Mix respectively. This demonstrates the effectiveness of the drift-gradient in SEGA.

Table 4.4: Validation of Regression Tasks on Synthetic Data (MAE as the Evaluation Criterion).

Data Streams	Data Description		Tested Models			Model Effectiveness	
	instances	Drift Type	Linear	SlidWin	SEGA-Linear	SEGA-Ideal	Effectiveness
Non-Drift	2000	no drift	0.800	0.811	0.810	-	-
Virt-Drift	2000	sudden virtual drift	0.780	0.798	0.793	-	-
Sudd-Drift	2000	sudden real drift	13.540	2.728	2.717	2.592	95.2%
Incr-Drift	2000	incremental real drift	10.200	2.315	2.307	2.220	96.1%
Rec-Drift-Grad	12000	sudden and gradual real drift	8.800	9.118	1.265	1.047	79.2%
Rec-Drift-Mix	12000	sudden, incremental and reoccurring real drift	8.170	2.486	1.680	1.584	93.9%

4.4.1.2 Classification Tasks

Data Description. For the classification task, ten widely used synthetic data is introduced to validate the effectiveness of SEGA on the drift problem in the classification task. All the synthetic data are available from (Liu, 2019).

Configuration. In the classification tasks, the predictor in SEGA is a weighted K-nearest neighbor (KNN) classifier with $K = 5$, the length of each segment is 200, the training set contains 10 segments, and the ensemble coefficient $c = 1$.

Analysis of Results on Synthetic Classification Tasks. The prediction results of classification tasks on synthetic data streams are shown in Table 4.5. These synthetic data contain various types of drift. Similar to the regression case, SEGA is compared to its non-adaptation edition and the sliding window edition. In classification tasks, KNN is applied as the basic predictor. The model effectiveness is not tested in classification tasks, since the prediction accuracy of SEGA-Ideal is more likely to be 1. This accuracy of 1 is largely due to the randomness, rather than the 100% accurate predictor. Therefore, it is not able to reflect the true capability of SEGA-Ideal.

Table 4.5: Validation on Synthetic Data of Classification Tasks (Acc as the Evaluation Criterion).

Data Streams	instances	Data Description	Tested Predictors		
		Drift Type	KNN	SlidWin	SEGA-KNN
SEAs	10,000	sudden real drift	80.95%	79.74%	80.79%
SEAg	10,000	gradual real drift	81.00%	79.73%	80.49%
HYPER	10,000	incremental and random reoccurring drift	77.38%	73.39%	73.73%
LEDs	10,000	sudden real drift	42.80%	40.83%	43.08%
LEDg	10,000	gradual real drift	42.80%	41.05%	42.99%
AGRs	10,000	sudden real drift	87.95%	49.23%	50.08%
AGRg	10,000	gradual real drift	87.95%	50.60%	50.71%
RTG	10,000	no drift	73.91%	62.78%	62.99%
RBF	10,000	virtual/real incremental drift	31.86%	39.99%	54.71%
RBFr	10,000	regionally virtual/real incremental drift	78.75%	76.38%	76.64%

The prediction results of classification tasks show that:

1) *Retrain by segments is better than Retrain by the whole training set.* This is concluded because according to Table 4.5, SEGA is no worse than SlidWin on all tested synthetic classification data streams.

2) *SEGA may not be suitable in cases where low-frequent and small drift exists.* In some data streams such as AGRs, although it contains drift, the accuracy of SlidWin is much worse than that of KNN, which denotes that non-adaptation is a better choice for this drifted data. This is because the drift in AGR data occurs at a low frequency and the new pattern has small difference from the previous one.

4.4.2 Evaluation on real-world data streams

In this section, SEGA will be evaluated on eight real-world data streams of regression tasks and seven real-world data streams of classification tasks. In the previous subsection, SEGA has been validated to solve concept drift effectively. The aim of this subsection is to compare it to some state-of-art regression and classification methods, which are specially designed for solving concept drift problems.

4.4.2.1 Regression Tasks

Data Description. The eight real-world regression data streams are: CCPP containing 9568 instances with four attributes; Sensor 3, 8, 20 and 46 containing 46,633, 15,808, 28,832 and 52,988 respectively with three attributes; SMEAR containing 140,576 instances with 43 attributes and Solar containing 32,686

instances with five attributes. Detailed information could be found in (Song et al., 2019a) with the download like in (Song, 2019). Among them, CCPP has been validated not to contain the concept drift problem (Yeon et al., 2010; Song et al., 2019a).

Configuration. In the regression tasks, the predictor in SEGA is ridge regression with $\alpha = 1e-04$ for CCPP, Sensor3, Sensor20, and Sensor46; $\alpha = 1e-03$ for Sensor8; $\alpha = 1e-02$ for SMEAR; and $\alpha = 1e-01$ Solar. The value of α is determined by parameter analysis, which will be provided in the next section. The length of each segment is 400 for SMEAR and 200 for the other data streams, because SMEAR has more attributes and a size of 200 is not large enough for training a predictor in this case. The K for searching nearest neighbors is half of the length of the segment, the training set contains 10 segments, the training set contains 10 segments, and the ensemble coefficient $c = 2$.

Analysis of Results on Real-world Regression Tasks. In the experiments of real-world regression data streams, the effectiveness of SEGA is presented by comparing it to six benchmark drift adaptation methods aiming to handle concept drift problems in a regression task. The benchmarks are ORTO (Ikonomovska et al., 2011b); FIMT-DD (Ikonomovska et al., 2011a); AMR and metaAMR (Duarte et al., 2016); Perceptron (Bifet et al., 2010c); and FUZZ-CARE (Song et al., 2019a). ORTO and FIMT-DD are tree model based methods, which use linear regression models and the stochastic gradient descent method in the leaves of the tree. ORTO and FIMT-DD detect the drift by Page-Hinckley (PH) test and use the detection result to adjust the tree structure. AMR and metaAMR are rule models and ensemble rules. For each rule model, a linear regression model is trained by an incremental gradient descent method. Perceptron is a

Hoeffding perceptron tree model which replaces the naive Bayes with perceptron predictor. FUZZ-CARE is implemented by the code in (Song, 2019) and all the other benchmarks are implemented by MOA (Bifet et al., 2010a). SlidWin uses ridge regression as the basic predictor.

The results of SEGA on real-world regression tasks are listed in Table 4.6 with the following concludes:

1) Using the whole training set is not the best strategy for drift adaptation.

According to the average rank results, it can be seen that SlidWin is the second worst drift adaptation method among these benchmarks. This phenomenon denotes that using the whole training set during the adaptation procedure is not a wise choice for the real-world data streams, because the drift situation is very complex in the real world.

2) SEGA is able to handle different drift cases in the real-world data. In (Song et al., 2019a), the authors have discussed that among the data streams in Table Table 4.6, Sensor20, Sensor46, SMEAR and Solar, are supposed to have significant reoccurring drift according to their experiments. As FUZZ-CARE is specially designed for reoccurring drift, it obtain better performance on these five data streams. Our proposed SEGA does not aim at solving a special type drift but is designed to be suitable for the occurrence of all types of drift. The highest average rank of SEGA compared to the other benchmarks validates the effectiveness of SEGA to solve concept drift problems in a data stream.

Table 4.6: Validation on Real-world Data of Regression Tasks (MAE as the Evaluation Criterion).

Data Streams	ORTO	FIMT-DD	metaAMR	AMR	Per	FUZZ-CARE	SlidWin _{ridge}	SEGA _{ridge}
CCPP	4.53E+02 (8)	3.57E+00 (3)	3.32E+00 (1)	3.42E+00 (2)	3.65E+00 (4)	5.59E+00 (7)	3.67E+00 (6)	3.67E+00 (5)
Sensor3	6.62E-02 (8)	7.14E-03 (3)	1.58E-02 (6)	7.69E-03 (4)	6.85E-03 (2)	1.55E-02 (5)	3.58E-02 (7)	6.13E-03 (1)
Sensor8	1.69E-01 (8)	9.68E-03 (5)	7.26E-03 (4)	6.57E-03 (2)	5.95E-03 (1)	1.72E-02 (6)	7.57E-02 (7)	7.14E-03 (3)
Sensor20	9.60E-01 (8)	7.95E-01 (6)	1.14E-02 (4)	8.19E-03 (3)	7.92E-01 (5)	7.90E-03 (2)	8.08E-01 (7)	7.35E-03 (1)
Sensor46	4.00E-01 (7)	1.57E-01 (4)	1.74E-01 (5)	2.02E-01 (6)	1.56E-01 (3)	5.25E-02 (2)	5.10E-01 (8)	5.87E-03 (1)
SMEAR	3.39E+01 (7)	2.34E+01 (5)	1.98E+01 (4)	1.44E+01 (2)	3.75E+01 (8)	1.04E+01 (1)	2.94E+01 (6)	1.73E+01 (3)
Solar	2.25E+02 (8)	1.14E+02 (5)	9.39E+01 (2)	9.52E+01 (3)	1.30E+02 (6)	8.66E+01 (1)	2.17E+02 (7)	1.09E+02 (4)
AvgRank(no CCPP)	7.67	4.67	4.17	3.33	4.17	2.83	7.00	2.17
AvgRank	7.71	4.43	3.71	3.14	4.14	3.43	6.86	2.57

4.4.2.2 Classification Tasks

Data Description. The seven real-world classification data streams are: Elec containing 45,312 instances with eight attributes; Weather containing 18,159 instances with eight attributes; Spam containing 9,324 instances with 39,916 attributes; Airline containing 539,383 instances with eight attributes; Cover-type containing 45,312 instances with 9 attributes; Usenet1 containing 1,500 instances with 99 attributes; and Usenet2 containing 1,500 instances with 99 attributes. More details of these data streams can be found in (Liu et al., 2018a; Bifet et al., 2010a).

Configuration. In the classification tasks, the predictor in SEGA is a weighted K -nearest neighbor classifier with $K = 5$. The length of each segment is 200 and each training set contains 10 segments for the data streams, except for Usenet1 and Usenet2, because Usenet1 and Usenet2 only have 1500 instances in total. For Usenet1 and Usenet2, the training set contains 7 segments. The K for searching nearest neighbors is half of the length of the segment, the ensemble coefficient $c = 6$.

Analysis of Results on Real-world Classification Tasks. For real-world classification data streams, the benchmarks here are all designed to specially solve the concept drift problem in the data stream of classification tasks, including ADWIN-ARF which uses ADWIN to detect drift and use an adaptive random forests for classification (Gomes et al., 2017b); NN-DVI, which detects drift via a density based distance and adapts to a new concept via competence model (Liu et al., 2018a); LevBag is ensemble method using an improved online bagging method to adapt to the changing data (Bifet et al., 2010b); SAM ensembles two

sliding window, with different window sizes on the KNN classifier (Losing et al., 2016); OnlineAUE (Brzezinski and Stefanowski, 2014) ensembles the classifier trained from blocks of training data which is similar to the segments in SEGA. OnlineAUE learns the weights of each block but SEGA uses the drift-gradient to select the best segment; IBLStream uses the instance-based model to adapt to the new concept by autonomously optimizing the size of the case base (Shaker and Hüllermeier, 2012); Learn++NSE is an ensemble method which use the tie-adjusted accuracy to determine weights (Elwell and Polikar, 2011). SlidWin uses the same basic classifier with SEGA, that is the KNN classifier for Weather, Spam, Airline and Covertypes, the Tree classifier for Elec, and the gradient boosting classifier for Usenet1 and Usenet 2.

The experimental results of SEGA on real-world classification tasks are listed in Table 4.7.

1) SEGA is also suitable to handle concept drift problems in classification tasks. Compared to the benchmarks which are specially designed to solve the concept drift problem in classification tasks, SEGA has the second highest average rank which validates the power of SEGA to solve the drift problem. In addition, the accuracy of SEGA can be further improved if appropriate predictors are chosen. For example, if decision tree is applied, the accuracy of SEGA on Elec will be 88.48%. We encourage users to try different predictor and ensemble parameters for getting better results of SEGA when they apply SEGA into a specific case.

Table 4.7: Validation on Real-world Data of Classification Tasks (Acc as the Evaluation Criterion).

	ADWIN-ARF	NN-DVI _{kNN}	LevBag _{kNN}	SAM _{kNN}	OnlineAUE	IBLStream	Learn++NSE	SlidWin _{kNN}	SEGA _{kNN}
Elec	88.17	86.67	81.91	82.78	87.74	77.05	70.52	62.81	83.46
	1	3	6	5	2	7	8	9	4
Weather	78.74	74.75	76.19	77.73	75.24	75.69	68.27	77.11	79.29
	2	8	5	3	7	6	9	4	1
Spam	95.60	94.65	93.22	95.79	84.29	92.78	70.56	66.90	94.57
	2	3	5	1	7	6	8	9	4
Airline	65.24	65.20	65.03	60.35	67.51	63.74	62.35	53.90	61.56
	2	3	4	8	1	5	6	9	7
Coverttype	92.11	94.04	94.00	91.71	90.01	92.26	64.03	71.88	94.72
	5	2	3	6	7	4	9	8	1
Usenet1	68.40	61.40	58.93	65.67	63.47	56.00	48.53	67.00	80.00
	2	6	7	4	5	8	9	3	1
Usenet2	71.93	71.40	67.33	71.00	68.87	67.67	66.67	70.00	71.00
	1	2	8	3	6	7	9	5	3
AveRank	2.14	3.86	5.43	4.29	5.00	6.14	8.29	6.71	3.00

2) *Combining the prediction results of selective segments by drift-gradient is an effective way to implement ensemble.* Most of the compared benchmarks are ensemble methods. Some benchmarks ensemble different classifiers while others ensemble the results computed on different data chunks which is similar to the SEGA method. OnlineAUE ensembles the results of data chunks. Therefore, OnlineAUE can particularly be compared to our SEGA method. According to the result table, SEGA obtain an average rank of 3.00, while OnlineAUE obtain 5.00. It can be seen that SEGA is much better than OnlineAUE on the average performance of the tested data streams. In addition, there is no need to train the ensemble weight in SEGA, which means SEGA is much quicker to implement ensemble. Therefore, it is not always recommended to ensemble all the available information. Selecting the most useful information from the training set is an more important aspect for drift adaptation.

Discussion. The proposed SEGA has been validated and compared from two aspects: experiments on synthetic data or real-world data and experiments on regression or classification tasks. Experiments on the synthetic data denotes that SEGA can improve the prediction accuracy because it can truly solve the concept drift problem in the data. Although the predictors in regression and classification are different, SEGA obtain uniformly good results, which denotes that SEGA is suitable to predict different kinds of data streams. Besides, the advanced performance of SEGA compared to the SlidWin method, demonstrates that it is not always best to use the whole training set to build the model when drift occurs, and the drift-gradient mechanism in SEGA can accurately and effectively select the most appropriate segments in training set for prediction.

4.4.3 Statistical test of real-world data streams

In this section, the results of the statistical test will be given to validate the significance of SEGA. The Friedman test and its post-hoc test after Conover are introduced as the testing method where the Friedman test is used to validate whether these drift adaptation methods are significantly different in general. Furthermore, the post-hoc test after Conover is used to validate the significance of pairwise comparison between SEGA and other methods. The statistical test includes tests on MAE of real-world regression data streams and Acc of real-world classification data streams.

The test process has been explained in Section 3.4.4. Given M the number of tested drift adaptation methods and n the number of data streams, the χ_R^2 statistic in the Friedman test is computed in (3.22) where R is the rank computed by MAE in regression cases and Acc in classification cases.

If the Friedman test reject the null hypothesis which means these drift adaptation methods are different in general, the post-hoc test will further test whether the difference between SEGA and other methods denoted by $R_i - R_{SEGA}$ is statistically significant ¹. $R_i - R_{SEGA}$ (refers to (3.23)) is significant if the following condition satisfies, where α is a preassigned significance level.

The results of statistical test are shown in Table 4.8 and Table 4.9 ². According to the statistical test, we can conclude:

¹the post-hoc test can test whether the difference between any two of the methods is significant. We only present the post-hoc test result between SEGA and other methods because we do not care whether other methods have significant difference between each other. More details of Friedman test and its post-hoc test can refer to (Pohlert, 2014).

²In these two tables, "+", "*", "**", and "***" means this value is significant at the level of 0.1, 0.05, 0.01 and 0.001 respectively. "df" denotes the freedom degree.

Table 4.8: Friedman test and its post-hoc test of all the methods over real-world regression data streams (no CCPP), where “Friedman Test” is the result for Friedman test and “Friedman - post-hoc test after Conover” is for the pairwise comparison. “+”, “*”, “**”, and “***” means this value is significant at the level of 0.1, 0.05, 0.01 and 0.001 respectively. “df” denotes the freedom degree.

Friedman Test	χ_R^2		P-value of χ_R^2		df		
	26.11		4.81e-2***		7		
Post-hoc test after Conover	ORTO	FIMT-DD	metaAMR	AMR	Per	FUZZ-CARE	SlidWin
$R_i - R_{SEGA}$	33	15	12	7	12	4	29
P-value	8.05e-06***	0.015*	0.0216*	0.1481	0.039*	0.2742	4.94e-05***

Table 4.9: Friedman test and its post-hoc test of all the methods over real-world classification data streams, where “Friedman Test” is the result for the Friedman test and “Friedman - post-hoc test after Conover” shows the pairwise comparison. “+”, “*”, “**”, “***” and “df” have the same meaning as they are in Table 4.8

Friedman Test	χ_R^2		P-value of χ_R^2		df			
	17.600		0.0244*		8			
Post-hoc test after Conover	ADWIN-ARF	NNDVI _{kNN}	LevBag	SAM _{kNN}	OnlineAUE	IBLStream	Learn++NSE	SlidWin
$R_i - R_{SEGA}$	-6	6	16	9	13	21	36	25
P-value	0.264	0.264	0.048*	0.173	0.088+	0.016*	2.00e-4***	0.0055**

1) *The Friedman test results are significant on of both regression and classification cases. The p-value of χ^2 in both cases is small which denotes a significant difference in the prediction accuracy among the tested adaptation methods.*

2) *In regression case, $SEGA_{ridge}$ is significantly better than most drift adaptation methods. In Table 4.8, all the p-values of the post-hoc test are significant except for the AMR and FUZZ-CARE column. This means, although SEGA is superior to AMR by a 7 rank difference and to FUZZ-CARE by a 4 rank difference, the difference is insignificant. Similarly, AMR and FUZZ-CARE cannot outperform SEGA, and this insignificance does not affect the effectiveness of SEGA handling the concept drift problem.*

3) *In the case of classification, $SEGA_{kNN}$ is significantly better than other adaptation methods except for ADWIN-ARF. In Table 4.9, the p-values of Friedman test is significant except for the comparison between ADWIN-ARF, $NNDVI_{kNN}$, SAM_{kNN} and $SEGA_{kNN}$. Similarly to the regression case, ADWIN-ARF, $NNDVI_{kNN}$, SAM_{kNN} are not better than $SEGA_{kNN}$.*

4.4.4 Parameter analysis and computation complexity

Figure 4.4 and Figure 4.5 show how the accuracy will change when the size of the segment w and the number of segments in the training set s have different values. In Figure 4.4, different values of the segment size w and the number of segments in the training set s are analyzed. The MAE results of all the data streams are put in the interval $[1, 9]$, and the legend gives the magnitude of each stream. For example, Sensor3(E-02) means the MAE at a specified parameter for this data stream is the value on the y-axis $\times 10^{-2}$.

4.4. EXPERIMENTAL EVALUATIONS

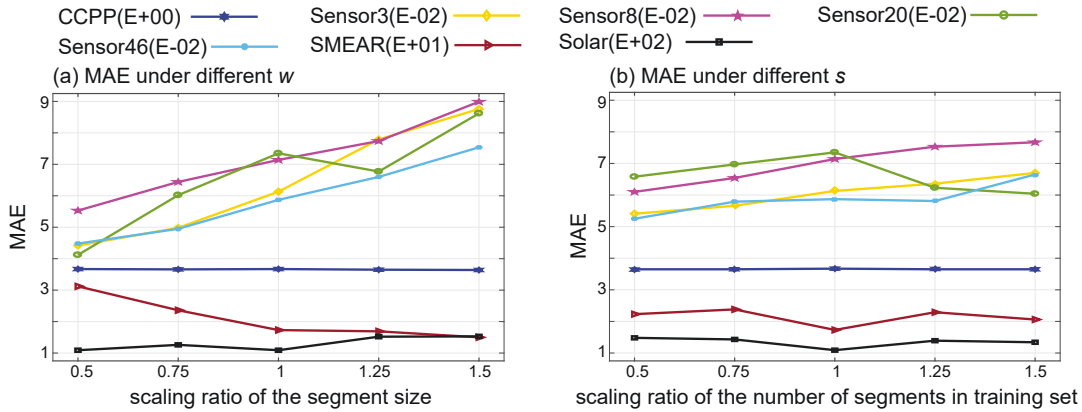


Figure 4.4: Parameter analysis for real-world data streams of regression tasks.

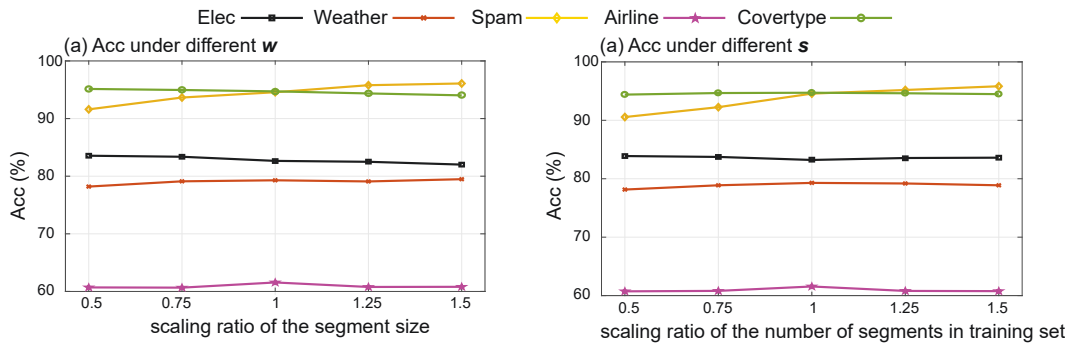


Figure 4.5: Parameter analysis for real-world data streams of classification tasks. Usenet1 and Usenet2 are not included because they have few instances.

We tested different w and s for all the tested real-world data streams by multiplying a scaling ratio with the current used w and s . For example, the accuracy of CCPP listed in Table 4.6, is computed with $w = 200$ and $s = 10$. In the subplot (a) in Figure 4.4, the first point in the CCPP line at 0.5 means the MAE is computed with $w = 0.5 \times 200 = 100$ and $s = 10$. For SMEAR, since its accuracy in Table 4.6 is computed with $w = 400$, the first point of SMEAR of subplot (a) in Figure 4.4, means this MAE is computed with $w = 0.5 \times 400 = 200$ and $s = 10$.

The batch size, which is the segment size w in SEGA, is a critical parameter for drift detection algorithms. For SEGA, if w is too small, the predictor trained on this segment may not be sufficiently trained. If w is too large, it is not necessary to separate the training set. The value of s determines how many previous instances is going to be stored to predict the upcoming instances. A larger s requires more storage. Therefore, for data streams with reoccurring concept problem, we suggest a larger s . In addition, a larger s means we need a larger storage, so $s \times w$ is also limited by the device. If w is large, it may not be able to choose a large s anymore. In addition, c determines how many predictors are ensembled. Therefore, for data streams that are difficult to predict, we suggest a larger c .

In general, SEGA is robust on these two parameters on most tested data streams according to the results shown in Figure 4.4 and Figure 4.5. For the data streams of Sensor3, Sensor8, Sensor20 and Sensor46, SEGA obtain better results with smaller w , which denotes that drift occurs at a relatively high frequency in these four data streams.

The computation complexity of SEGA is determined by the size of the segments (w) and the number of segments in each training set (s). The complexity of computing SSD is $s \times O(w^2)$ for every w instances. In each learning process, the complexity is $(s - 1) \times O(w)$ when computing the drift gradient of δ_Q and is less than $((s - 1)w) \times O((s - 1)w)$ when computing the drift gradient of δ_P . Given the size of training data as N , in each learning process, the computation complexity is upper bounded by $O(N^2)$ if SEGA updates δ_P and the complexity is upper bounded by $O(N)$ if the updates of δ_P are skipped. The run time of SEGA with kNN the predictor is listed in Table 4.10, where SEGA is implemented

in Python while others are implemented in Java. The execution time of SEGA could be shortened if parallel computing is involved to shorten the run time of for-loop. Meanwhile, although SEGA uses KNN search the neighbors for updating drift-gradient, and a KNN predictor for classification tasks, we do these two procedures separately, which has repeated computation. However, this makes SEGA more flexible when choosing predictor and computing the distance matrix.

4.5 Summary

This chapter proposes an online adaptation method, called SEGA, to predict labels for data streams of both regression and classification tasks. Instead of using the whole training data to retrain predictors, which is common in most recent research on concept drift adaptation, SEGA trains and updates predictors on selected segments of the training data. In SEGA, we propose a sequentially updated statistic, drift-gradient, to select the optimal segments when every new instance arrives. In this way, SEGA can overcome the delay of the informed drift adaptation method, as well as the instability of the blind drift adaptation method. Our SEGA method is validated by experiments on 30 data streams including synthetic or real-world data streams of classification or regression tasks. The consistent performance shows that SEGA is able to solve various types of drift under different situations.

Table 4.10: Run-time on real-world data streams (s CPU-time)

CPU(s)	ADWIN-ARF	NN-DVI _{kNN}	LevBag _{kNN}	SAM _{kNN}	OnlineAUE	IBLStream	Learn++NSE	SlidWin _{kNN}	SEGA_{kNN}
Elec	11.20	374.13	598.71	7.06	4.08	8.98	6.20	2.97	173.50
Weather	4.01	183.64	329.23	3.00	1.00	37.09	2.02	6.39	82.58
Spam	6.01	4899.40	11331.10	23.01	7.56	1131.02	5.02	33.09	343.20
Airline	355.19	5754.27	7336.86	40.01	196.15	1314.86	852.05	213.51	2295.45
Coverttype	123.04	86436.71	48499.82	196.06	112.04	2344.83	2357.61	333.53	3678.54
Usenet1	1.00	132.66	377.24	1.00	1.00	4.59	1.01	0.19	1.23
Usenet2	1.01	124.08	382.16	1.00	1.00	5.11	1.01	0.18	1.24

SlidWin, SEGA_{kNN} are implemented in Python, while other methods are implemented in Java by MOA

DRIFT ADAPTATION BY GENERATING SAMPLES OF NEW CONCEPT

Identifying a new concept in its early stages means there are few available instances of the new concept. It is difficult to learn a precise predictor for the new concept if there are not enough instances. To solve this problem, this chapter proposes to generate samples of new concept by inputting the previous data instances. We also propose a drift adaptation method based on the generating process. In this chapter, the problem is introduced in Section 5.1. Definitions and notations are listed and explained in Section 5.2. Section 5.3 explains how to generate samples of new concept and then make adaptation. The proposed adaptation method is validated by experimental evaluations in Section 5.4. Section 5.5 concludes this chapter.

5.1 Introduction

For a data stream with concept drift, the new concept will be identified in the upcoming data instances. However, identifying a new concept in its early stages means there are few available instances of the new concept with which to build a precise predictor. In addition, using drift detection method to identify drift requires at least one full batch of new instances; therefore, the length of the batch determines the detection lag. Reducing the detection lag means reducing the number of instances, and the accuracy of drift detection may suffer. Conversely, increasing the number of instances to a level sufficient for learning accurate predictors will result in greater detection lag.

Due to the problem that the data of the new concept is insufficient, drift adaptation methods are faced with a choice between insufficient learning and adaptation delay. The lack of data or imbalances in the data can be alleviated with data resampling techniques. For example, Wang et al. proposed an ensemble method based on resampling for online learning with imbalanced data (Wang et al., 2015) and Lu et al. applied a permutation test to detect concept drift (Lu et al., 2014). None of these methods, however, are able to generate instances that represent the new concept. If these new instances could reasonably be generated, a new concept could be learned with less new instances.

Motivated by this idea, and given the success of fuzzy theory in handling uncertainty problems (Zuo et al., 2016), we developed an adaptive fuzzy network (AFN) to adapt to concept drift in data streams. AFN has two functions: a drift detection module that identifies whether drift has occurred; and a drift adaption module that simulates the new concept. These two modules are integrated in

a fuzzy inference system (FIS), which provides a new solution for concept drift problem. In terms of drift detection, AFN does not need to monitor learner errors or generate statistics for measurement. In terms of drift adaption, AFN updates the model by generating samples of new concepts through previous data instead of directly discarding them.

Three sets of comparative experiments on nine data streams were conducted to separately demonstrate the effectiveness of the drift detection module, the drift adaption modules, and the FIS. The results consistently demonstrate the superiority of the three components and the full model.

The advantages of implementing adaptation by the proposed adaptive fuzzy network (AFN) are listed as follows.

- AFN can detect drift and adapt to the new concept simultaneously.
- AFN embeds generative adversarial nets (GAN) in a fuzzy network. The generative model can generate data for learning new concept.
- AFN designs a drift detection module on the adversarial model. The detection method does not rely on the learner error or a statistic.
- AFN is a network so it can be easily combined to other neural networks.

5.2 Definitions and Notations

The definitions of data stream, concept drift and the learning aim are the same to the definitions in Section 3.2. AFN contains a generative adversarial nets (GAN) and an adaptive neuro-fuzzy inference system (ANFIS). In this section, GAN and ANFIS will be introduced.

5.2.1 Generative adversarial nets (GAN)

Goodfellow (Goodfellow et al., 2014), proposed GAN for use in an adversarial process to estimate generative models, which is used to learn the distribution of data. The main idea of GAN is that a good generative model can confuse a well-trained discriminative model by labeling an arbitrary example as either data or fake data. Given $G(z; \theta_g)$, the generator, which is a differentiable function that maps a noise variable z to a distribution p_g , and the discriminator $D(x; \theta_d)$, which maps x to a single scalar, represents the probability that x came from the data rather than p_G , $D(x; \theta_d)$ is trained to maximize the probability of assigning the correct label to true and fake samples, while $G(z; \theta_g)$ is trained as a minimizer. Specifically, D and G play the following two-player min-max game with the value function $V(G, D)$:

$$\underset{G}{\min} \underset{D}{\max} V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log(D(\mathbf{x}))] + \mathbb{E}_{z \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

(5.1)

When $V(D, G)$ reaches its maximum value, $D(\mathbf{x})$ approaches its max of 1 and $D(G(\mathbf{z}))$ approaches its minimum of 0, which means the discriminator D classifies the example from the data as true data and the samples generated by G as fake data. Given a trained D , $\mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})]$ is a constant value, so minimizing $V(D, G)$ is the same as $D(G(\mathbf{z}))$ approaching 1. Hence, the generator lowers D 's accuracy. As G and D update, the optimal D for any given G is (5.2) and the optimal G for any given D is achieved if, and only if (5.3) is satisfied. The

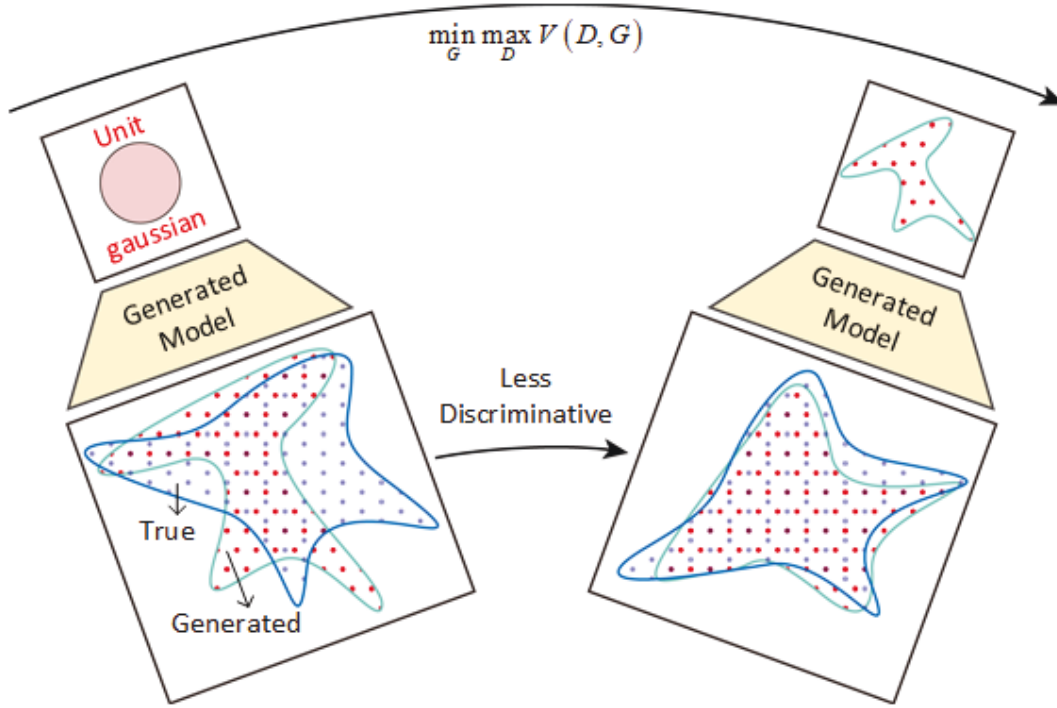


Figure 5.1: Generative Adversarial Net (GAN).

computation process for GAN is shown in Figure 5.1. Given a random Gaussian series, the generative model maps this series to $G(\mathbf{z})$. Then, a discriminator is applied to classify the generated data and real data. After all iterations are complete, the generated data should approach the same distribution as the real data.

$$(5.2) \quad D_G^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}$$

$$(5.3) \quad G_D^*(\mathbf{z}) = \{G | p_g = p_{data}\}$$

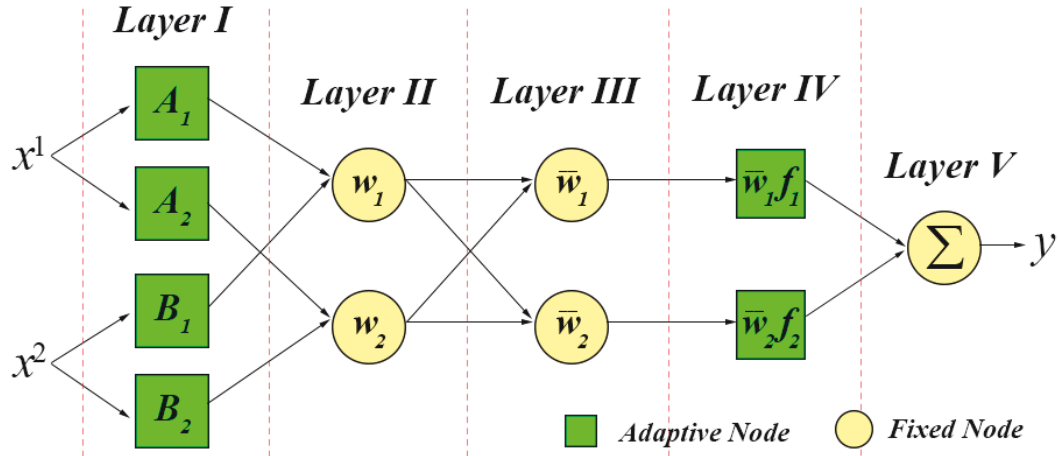


Figure 5.2: Adaptive neuro-fuzzy inference system (ANFIS).

5.2.2 Adaptive neuro-fuzzy inference system

Fuzzy logic has been widely used to handle uncertainty problems. Typically, an FIS maps a certain input to several fuzzy sets with corresponding degrees of membership. Jang (Jang, 1993) restructured FISs with two contributions: a standard method for transforming ill-defined factors into identifiable rules in the FIS; and the use of an adaptive network to tune the membership functions. This approach yielded ANFIS, which has been successfully used in many uncertain systems, such as wind energy and pollution forecasting (Song et al., 2015; Zhang et al., 2016). The ANFIS process is illustrated in Figure 5.2, given x the attribute variables, y the label variable. There are five processes in an ANFIS architecture. The circles represent fixed nodes without parameters, and the squares represent adaptive nodes, whose parameters are determined by the training data and a gradient-based learning procedure.

- **Layer I:** Maps a certain input x to a fuzzy set O_i^1 for every node i according to the member functions μ_{A_i} . This usually bell-shaped with the parameter

set $\{a_i, b_i, c_i\}$. The output of layer I is computed by (5.4) and (5.5).

$$(5.4) \quad O_{1,i} = \mu_{A_i}(x^1) = \exp\{-(x^1 - c_i)/a_i\}^2$$

$$(5.5) \quad O_{1,i} = \mu_{B_i}(x^2) = \exp\{-(x^2 - c_i)/a_i\}^2$$

- **Layer II:** The nodes in this layer are fixed. Each circle node performs the connection "AND" by multiplying the inputs and passing the product to the next node. Each node represents the "firing strength" of the instance for each rule, i.e., the degree to which the "if" component of a fuzzy rule is satisfied.

$$(5.6) \quad O_{2,i} = w_i = \mu_{A_i}(x^1) \times \mu_{B_i}(x^2), \quad i = 1, 2$$

- **Layer III:** The nodes in this layer are fixed. A normalized firing strength is calculated for every circle node in this layer. The firing strength is the ratio between the i^{th} rule's firing strength and the sum of all the rules' firing strengths.

$$(5.7) \quad O_{3,i} = \omega_i = \frac{\omega_i}{\sum_i \omega_i}$$

- **Layer IV:** The nodes in this layer are adaptive to an output. Assuming the rules of this system as presented in Rules 1 and 2, the output of the adaptive node is computed by (5.10).

Rule 1: if x^1 is A_1 and x^2 is B_1 , then

$$(5.8) \quad f_1 = p_1 x^1 + q_1 x^2 + r_1$$

Rule 2: if x^1 is A_2 and x^2 is B_2 , then

$$(5.9) \quad f_2 = p_1x^1 + q_2x^2 + r_1$$

$$(5.10) \quad O_{4,i} = \bar{\omega}_i(p_ix^1 + q_ix^2 + r_i)$$

- **Layer V:** The nodes in this layer are fixed. The overall output is the weighted average of all incoming signals.

$$(5.11) \quad O_{5,i} = \sum_i \bar{\omega}_i f_i = \frac{\sum_i \omega_i f_i}{\sum \omega_i}$$

Layer I in ANFIS is a fuzzification layer. We use fuzzy C -means clustering method to construct the fuzzy set (Yang et al., 2010). The process of ANFIS is presented in Algorithm 5.1, where k represents the number of clusters, and X_{train} and y_{train} and X_{test} and y_{test} represent the input and output in the training set and the test set, respectively.

5.3 Drift Adaptation by an Adaptive Fuzzy Network—AFN

The whole adaptation process of the fuzzy generative adversarial net (AFN) will be explained in Section 5.3.3. There are three parts in AFN: *detection* (Section 5.3.1), *generation* (Section 5.3.2) and *adaptation* (Section 5.3.3). Their corresponding techniques are the adversarial model, the generative model, and ANFIS (Section 5.2).

Algorithm 5.1: Adaptive Neuro-fuzzy inference system

Input : $X_{Train}, y_{train}, X_{test}, k$.
Output: \hat{y}_{test} : The estimation of y_{test} .
Initialization: $\nabla_M \leftarrow \mathbf{0}$.

- 1 **for** $i = 1$ **to** $MaxIteration$ **do**
- 2 Update the membership matrix U ;
- 3 Calculate the new cluster centers V ;
- 4 Calculate the clustering objective function;
- 5 **if** $|J_t - J_{t-1}| \leq \epsilon$ **then**
- 6 **break**;
- 7 **end**
- 8 **end**
- 9 **return** U, V .
- 10 Built the fuzzy rules (5.8) and (5.9) by U, V ;
- 11 Update ω in (5.10);
- 12 **return** fuzzy rules and ω ;
- 13 $\hat{y}_{test} = fis(X_{test} | \omega, rules)$;
- 14 **return** \hat{y}_{test} .

5.3.1 Detection by the adversarial model in GAN

The principle behind using an adversarial model to detect drift is that, if the discriminator can well separate two samples of data, a drift is identified. Here, a sigmoid function $h(x)$ is used to classify samples denoted by S_1 and S_2 . If the series $h(S_1)$ is significantly different from the series $h(S_2)$ according to a Z-test, the discriminator is thought as a “good” classifier which can separate two samples. The detection process using the adversarial model is shown in Algorithm 5.2, given X_p and y_p , i.e., the attributes and label variables of the previous data, and X_c and y_c , which correspond to the current data. It should be noticed that, the drift detection proceeds after we obtain the labels of the newly arrived data.

Algorithm 5.2: Detect drift by the adversarial model $D(x)$

Input : X_p, y_p, X_c, y_c
Output: $H = 1$: Drift is identified;
 $H = 0$: Drift is not identified.
Initialization: $H \leftarrow 0$.
1 $S_p \leftarrow [X_p \ y_p]$;
2 $S_c \leftarrow [X_c \ y_c]$;
3 Update the discriminator by stochastic gradient:
 $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(S_c) + \log(1 - D(S_p))]$;
4 $Z_p \leftarrow D_{\theta}(S_p)$;
5 $Z_c \leftarrow D_{\theta}(S_c)$;
6 $H \leftarrow Z - \text{test}(Z_p, Z_c)$;
7 **return** H .

5.3.2 Generating data by the generative model in GAN

Under the assumption that a mean drift is contained within the data stream, the generative model draws on previous data by adding a mean-difference matrix. The mean-difference matrix is computed by the average value of the previous data and the new data. The details are listed in Algorithm 5.3.

Algorithm 5.3: Generate data by the generative model $G(x)$

Input : X_p, y_p, X_c, y_c, H .
Output: G_X : The generated features;
 G_y : The generated label.
Initialization: $\nabla_M \leftarrow \mathbf{0}$.
1 $S_p \leftarrow [X_p \ y_p]$;
2 $S_c \leftarrow [X_c \ y_c]$;
3 **if** $H \neq 0$ **then**
4 $\nabla_M = \text{mean}(S_p) - \text{mean}(S_c)$;
5 $G_X = S_p + \nabla_M$.
6 **end**
7 **return** G_X .

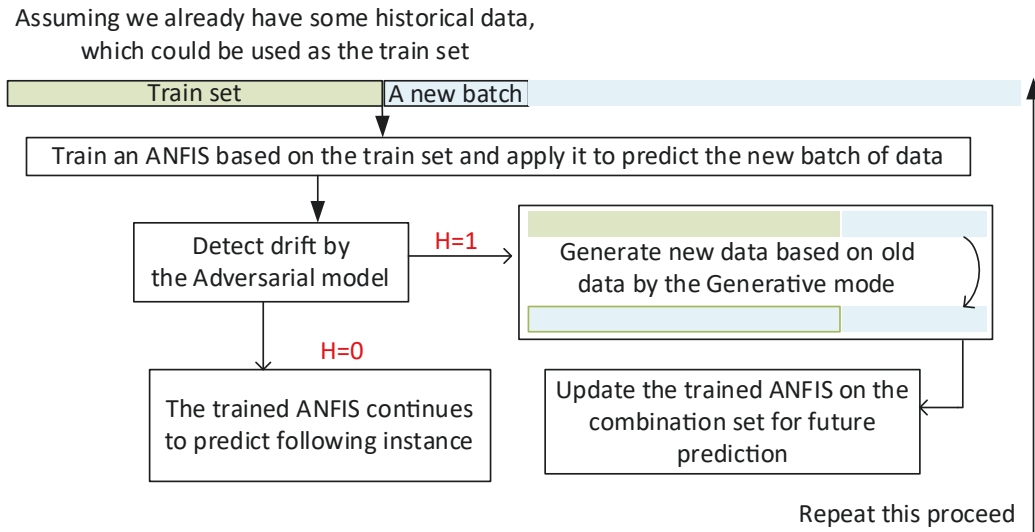


Figure 5.3: The flowchat of AFN.

5.3.3 The general procedure and pseudocode of AFN

The adaptation process in AFN is as follows: the adversarial model in GAN is used to recognize whether a sample is real or fake. A sample from the old concept is considered to be “fake”; a sample from the new concept is considered “real”. If the samples are well separated by the discriminator, a drift is detected. Once a drift is detected, a chunk of fake data representing a similar concept to the new concept is generated by the generative model rather than directly updating the model with the newly arriving data. The original GAN uses a random series to generate data. However, we use previous observations instead of a totally random series. For a real-world data stream, it is possible that previous instances contain useful information for training the new concept. Given this condition, using previous data will improve learning performance. Once the generated data is ready, ANFIS learns a predictor based on the union of the latest data and

the generated fake data. This predictor is trained to predict the next batch of instances. AFN integrates all three components. The flowchart of the integration is presented in Figure 5.3. The green bar represents the training set, namely the historical data and the blue bar represents the test sets. Before assessing the first window of newly arrived instances, the ANFIS model is trained on the training set. The trained model is then used to predict subsequent instances. ANFIS will be updated when a drift is detected by Adversarial model. First, the adversarial model detects whether or not a drift has occurred. If a drift is detected, the generative model is activated. The generated data combined with the current data are used as a new training set for the next prediction.

5.4 Experimental Evaluations

We evaluated AFN through a set of experiments in three respects: 1) the effectiveness of the adversarial model, 2) the effectiveness of the generative model, and 3) the effectiveness of the fuzzy prediction model. Accordingly, three comparisons were conducted: 1) a comparison of the method with and without the adversarial model to detect drift; 2) a comparison of the method with and without generating samples for the generative model for adaption; and 3) a comparison of fuzzy inference prediction between our method and other prediction methods. The experimental designed is summarized in Table 5.1.

Table 5.1: Experimental design in this section.

Data	Experimental aim	Results
d1	the effectiveness of the generative model in AFN	Table 5.2: Comparison3
d1	to validate the effectiveness of the adversarial model in AFN	Table 5.2: Comparison2
d1	to validate the effectiveness of ANFIS in AFN	Table 5.2: Comparison1
d1	To demonstrate the robustness of AFN	Table 5.3
d1: real-world classification data streams		

5.4.1 Experiment setup

Data Description. We apply the arabic data to to test the three aspects of AFN. The Arabic dataset is available in (dos Reis et al., 2016). The original data contains 8800 example consisting of audio feature of 44 males and 44 females. The dataset is strictly sorted into four equal sizes of alternating male and female voices. The label variable has ten classifications labeled by 0 to 9, which is difficult to evaluate with the *Acc*, *Rec* and *F1* measurements. We separated this dataset into several individual datasets, each representing a binary classification problem. For example, the first separated data stream only contained samples belonging to the class of "0" or "1". Before separation, the starting point of the drift was known from prior experience of gender changes every 2200 instances. After generation, the drift point was more random because the label variable was not sorted, which is common in real-world cases.

Evaluation Metrics. We assumed, for the purposes of the experiments, that a small batch of instances in the data stream with no concept drift were available. This batch of instances was used as the training set. The data stream arrived sequentially, so any drift would occur in the upcoming data. Each instance arriving after the model had been trained on the initial set of instances was a sample to be used for prediction. Once arrived, the instance's label variable was observable, it could be used to update the model for future instances. Our evaluation criteria included accuracy, precision, recall, and F1 score. Each metric was computed for every instance in the data stream except for the batch reserved as the training set. A model's performance was judged by the mean value of each evaluation criterion. The calculations for the evaluation criteria are shown in

5.12, 5.13, 5.14, 5.15.

$$(5.12) \quad Acc = \frac{|y = \hat{y}| + |\bar{y} = \hat{\bar{y}}|}{|y| + |\bar{y}|}$$

$$(5.13) \quad Pre = \frac{|y = \bar{y}|}{|y = \hat{y}| + |\bar{y} = \hat{\bar{y}}|}$$

$$(5.14) \quad Rec = \frac{|y = \bar{y}|}{|y|}$$

$$(5.15) \quad \frac{2}{F1} = \frac{1}{Pre} + \frac{1}{Rec}$$

where y denotes the sample belongs to a specified class, \bar{y} denoting the sample belongs to another class, \hat{y} and $\hat{\bar{y}}$ are their corresponding estimations.

Preassigned Parameters. The most important parameters in AFN are the length of the training set TN and the size of the non-overlapped sliding window w . We tested four groups of parameters: $S^1 : (TN = 200, w = 50)$, $S^2 : (TN = 200, w = 100)$, $S^3 : (TN = 500, w = 50)$, $S^4 : (TN = 500, w = 100)$. The results of the experiment are specified in Section 5.4.2 as S^1, S^2, S^3, S^4 . A further parameter in the ANIFS component of AFN is the number of clusters k . In this paper, $k = 5$. SVM is involved as a benchmark, its parameters are preassigned as follows. svm_type: one-class SVM, kernel_type: sigmoid: $\tanh(\text{gamma} * u * v)$, degree in kernel function: 3, gamma in kernel function: $1/\text{num_features}$, parameter nu of one-class SVM: 0.5, cache memory size in MB: 100, tolerance of termination criterion: 0.001, whether to use the shrinking heuristics: 1.

5.4.2 Experimental results

All the experimental results are listed in Table 5.2. The effectiveness of ANFIS is demonstrated by comparison between SVM-non-adaptive and ANFIS-non-adaptive row, the adversarial model is tested through comparing with ANFIS-non-adaptive, and AFN will be compared to ANFIS-adversarial and ANFIS-always-adaptive. The values in the "Pre." row are average value of nine datasets, and same as they are in the "Rec." and "F1" rows. "**SVM-non-adaptive**" includes the results for the method that builds an SVM model using the training set and applies it to predict the following instances without updates. Similarly, "**ANFIS-non-adaptive**" uses ANFIS to predict without updates. "**ANFIS-adversarial**" includes the results for the method that updates the ANFIS model when the adversarial model detects a drift. "**ANFIS-always-adaptive**" shows the results for the method that updates the ANFIS model according to windows. And AFN lists the results for the method that updates the ANFIS model using generated data when a drift is detected.

- **Comparison1:** whether fuzzy rules improve the adaptation performance. *Comparison between SVM-non-adaptive and ANFIS-non-adaptive.* ANFIS-non-Adaptive. This comparison demonstrates the effectiveness of the FIS. From these results, we conclude that applying fuzzy rules improves classification performance.
- *Comparison between ANFIS-non-adaptive and ANFIS-always-adaptive.* ANFIS-always-adaptive is a baseline that represents the best possible performance for a windows-based adaption strategy. It is also used to test whether drift exists in real data streams. Here, accuracy was significantly

improved when the model was continuously updated. Therefore, drift exists in these data streams.

- **Comparison2:** whether adversarial model effectively detects drift. *Comparison between ANFIS-adversarial and ANFIS-always-adaptive.* The adversarial model detects drift. It is less useful if its performance is close to ANFIS-non-adaptive's, and more useful if its performance is close to ANFIS-always-adaptive's. Table 5.2 shows that the accuracy of ANFIS-adversarial was higher than ANFIS-always-adaptive in all data streams except column G. This result indicates that the proposed adversarial model can detect drift in a precise manner and the always-adaptive strategy does not lead to the best result. In other words, incorrectly updating a model at non-drifted points leads to poorer outcomes.
- **Comparison3:** whether using generated data improves the adaptation performance. *Comparison between ANFIS-adversarial and AFN.* This comparison evaluates whether it is reasonable to abstract new knowledge from old data. The higher performance of AFN over ANFIS-adversarial and ANFIS-always-adaptive demonstrates that old data does contain useful information for predicting new data, and the proposed generative model can abstract this information.

Table 5.2: Main results table.

Criterion	Parameter	Comparison 1		Comparison2		Comparison3	
		SVM-non-adaptive	ANFIS-non-adaptive	ANFIS-adversarial	ANFIS-always-adaptive	ANFIS-adversarial	AFN
Acc. (%)	S1	43.42	80.39	97.16	96.45	97.16	98.26
		52.65	96.58	98.39	97.68	98.39	98.32
		62.26	92.65	98.58	98.13	98.58	98.97
		53.48	75.23	85.42	85.23	85.42	86.00
		44.71	51.74	66.65	66.19	66.65	67.48
		49.03	84.65	96.45	95.87	96.45	97.29
		46.71	94.77	97.48	97.74	97.48	97.81
		49.68	64.19	72.84	72.32	72.84	74.00
		49.61	95.35	96.32	95.94	96.32	96.71
Pre. (%)	S1	36.15	51.90	49.59	49.62	49.59	49.53
Rec. (%)	S1	36.21	84.51	89.37	88.94	89.37	89.87
F1	S1	0.36	0.64	0.64	0.63	0.64	0.64

Table 5.3: Robustness test of AFN with different parameters.

Criterion	Parameter	Methods	A	B	C	D	E	F	G	H	I
Acc. (%)	S^2	SVM-non-adaptive	43.47	52.80	61.87	53.07	44.33	49.00	46.60	49.47	49.87
		ANFIS-non-adaptive	80.93	96.53	93.07	75.53	52.60	85.00	95.13	64.33	95.33
		ANFIS-adversial	94.13	98.33	98.13	83.60	64.67	95.80	97.33	72.00	95.80
		ANFIS-always-adaptive	94.20	97.13	96.60	83.80	63.87	92.33	96.87	72.27	94.93
		AFN	97.27	98.13	98.20	86.07	67.07	96.13	97.47	73.20	96.73
Acc. (%)	S^3	SVM-non-adaptive	44.08	54.56	62.88	49.36	46.00	40.56	48.56	48.96	62.72
		ANFIS-non-adaptive	99.04	97.04	98.64	82.00	62.32	97.52	96.08	71.28	96.00
		ANFIS-adversial	98.64	98.24	97.68	83.44	64.16	96.48	97.28	72.24	96.32
		ANFIS-always-adaptive	98.64	97.84	97.68	83.12	64.16	96.56	97.20	71.52	96.48
		AFN	98.96	98.32	99.28	84.88	67.92	97.60	97.76	73.60	96.72
Acc. (%)	S^4	SVM-non-adaptive	43.67	54.67	62.58	48.92	45.67	40.75	48.67	48.75	62.75
		ANFIS-non-adaptive	99.00	96.92	98.67	82.08	62.75	97.58	96.42	71.25	96.17
		ANFIS-adversial	98.67	97.92	98.08	83.67	64.17	96.75	97.25	72.08	96.42
		ANFIS-always-adaptive	98.58	97.92	98.08	82.83	63.00	96.75	97.25	71.08	96.42
		AFN	98.92	98.25	99.33	84.92	66.58	97.33	97.67	73.00	96.75

Table 5.3 presents the robustness tests for AFN. The length of the training set and the window size are differently combined in the parameter column. In S^2 , the length of the training set and window size are (200, 100), S^3 is for their value of (500, 50) and S^4 is for (500, 100). Similar results were obtained to the result for S^1 . The only difference is that there is no need to use drift adaptive models for a large length of the training set because the training set contains enough information for different concepts. For example, the ANFIS model achieved its best performance for the data streams A, C, and F using the non-adaptive strategy, as indicated by the results for S^3 and S^4 . Even in this scenario, AFN and ANFIS-adversarial performed better than ANFIS-always-adaptive, which signals that our method is very robust.

5.5 Summary

This chapter presents a novel drift adaptation method, called adaptive fuzzy network (AFN). AFN is able to adapt to the potential concept drift in data streams and solve the difficulty of data shortage when a new concept is identified at its early stage. AFN is a neural network which embeds GAN in a fuzzy network.

AFN detects when a drift occurs using an embedded adversarial model and generates sample instances of the new concept when a drift is detected. AFN was evaluated with the Arabic data stream, where it is unknown when and how the drift occurs. The results show that the AFN improves the prediction performance. In addition, we test AFN by validating the effectiveness of each component in AFN. The experiments show each component in AFN is effective.

DRIFT ADAPTATION WITH NOISY DATA

Chapter 3 Chapter 4, and Chapter 5 provide three concept drift adaptation methods from three different aspects separately. In these chapters, the data stream is assumed to only contain the concept drift problem. However, data streams may also show other uncertain characteristics. In this chapter and the next chapter, we will discuss the concept drift problem in two scenarios that are more realistic. In this chapter, we will discuss the concept drift adaptation in a noisy data stream. This chapter is organized as follows: Section 6.1 introduces the scenario and give a general description of our solution. Definitions and notations are listed in Section 6.2. The solution details are explained in Section 6.3 with its experimental evaluations in Section 6.4. Last, a summary concludes this chapter with discussions in Section 6.5.

6.1 Introduction

Concept drift problem will be more challenging if problems found in a batch-learning setting also present in a data stream. One of the problems is to learn in a noisy environment (Gomes et al., 2017a). Noise is described as “irrelevant or meaningless data” (Xiong et al., 2006), and data cleansing or noise removal techniques are applied to remove the the irrelevant noise (Hernández and Stolfo, 1998). However, this description of noise is not suitable for noisy data streams because it does not cover the noise in signal processing or time series analysis.

The data stream may also contain signal noise. Compared to the stationary data, a data stream contains time stamps. Therefore, each variable in the data stream is a signal or a time series. In signal processing and time series analysis, the noise is not presented by several data instances. In stead, the noisy signal or series mixes with the designed signal or time series all the time such as background music in voice audio signals and Gaussian noise in time series analysis. Signal filtering is a common solution for this kind of noise, where the noise signal is normally abstracted by a decomposition and filtering technique (Xie et al., 2018). In order to distinguish these two kinds of noise, the noise appears with some points is categorized as *outliers* and the consistently existing noise as *signal noise* in this chapter.

The learning becomes more complex when concept drift and signal noise simultaneously occur in a data stream. Simple combination of concept drift methods and noise removal methods is less capable to deal with this kind of data streams. The main reason is that concept drift and noise are not always independent in a data stream, and they may have some overlaps sometimes.

For example, the fluctuation of noisy variables may be wrongly identified as a drift. In this case, a predictor trained with clean data may be replaced by a predictor trained with the noisy data because noisy data is considered as new data. In contrast, when an incremental drift starts to occurs, it would probably be identified as noise. In this case, the drift adaptation will be severely delayed because the obsolete predictor is not updated. As pointed out in (Gomes et al., 2017a), noise or outliers ought not be confused with drift.

In this chapter, we propose a noise-tolerant drift adaptation (NoA) method for predicting data stream with concept drift and signal noise. The drift adaptation process is implemented by FUZZ-CARE (Section 3) where the fuzzy clustering process can include the most relevant data instances in the training set to the latest pattern. Based on that, an incremental denoise technique is proposed and embedded in the drift adaptation process to remove the signal noise. Compared to the existing denoise techniques in the literature, the proposed denoise technique can incrementally update and therefore is able to be combined with the incremental adapted regression predictor.

The uniqueness of our method is to make prediction for data streams with both concept drift and signal noise problems. Details are shown in Table 6.1. Current denoising techniques do not consider the concept drift problem, and the denoising methods only focus on signal noise. Label-noise methods aim to learn with data with noisy labels (outliers). Most drift adaptation methods do not include noisy cases. Several drift adaptation methods that can solve the noise problem only consider the outlier noise leaving signal noise unsolved. NoA fills this gap.

Table 6.1: The uniqueness of NFA.

	Concept drift	Outliers	Signal Noise
Denoise methods	×	×	✓
Label-noise methods	×	✓	×
Most drift adaptation methods	✓	×	×
Limited drift adaptation methods	✓	✓	×
Our drift adaptation method	✓	×	✓

6.2 Definitions and Notations

The definitions of data stream, concept drift and the learning aim are the same to the definitions in Section 3.2. In NoA, the drift adaptation is implemented based on FUZZ-CARE, which has been presented in Chapter 3. An incremental denoise technique is designed based on the empirical mode decomposition (EMD) method to deal with the signal noise. In this section, the original empirical mode decomposition will be introduced.

The empirical mode decomposition (EMD) method (Huang et al., 1998) is one of the subspace filtering (SSF) technique. SSF is a class of signal enhancement technique based on matrix factorization, which has been proved to effectively remove the signal noise in a time series and therefore widely studied in the literature and applied in practice (Xie et al., 2018). The idea of EMD is using the Hilbert-Huang transform (HHT) to decompose the original signal into a finite and often small number of intrinsic mode functions (IMFs) and one residual series. Then, high-frequency series are deleted as noise from the signal. The sum of the rest IMFs with low-frequency constructs a new signal which does not contain noise anymore (Guo et al., 2012).

Definition 6.1 (Intrinsic Mode Functions (IMF)(Huang et al., 1998)). An IMF is

defined as a function that satisfies the following conditions:

- 1) in the whole data set, the number of extrema (sum of maxima and minima) and the number of zero crossings must either equal or differ at most by one;
- 2) at any point, the mean value of the envelope defined by the local maxima and the envelope defined by the local minima is zero.

For an arbitrary signal $x(t)$, it can be decomposed by the following sifting steps:

- 1) identify all the local extrema;
- 2) connect all the local maxima with a cubic spline line as the upper envelope;
- 3) connect all the local minima with a cubic spline line as the lower envelope;
- 4) compute the mean of the upper and lower envelope, designated as $m_k(t)$;
- 5) get the k -th component $h_{1,k}(t)$ in (6.1);
- 6) end this sifting process until $h_{1,k}(t)$ is an IMF.

$$(6.1) \quad \begin{cases} h_1(t) = x(t) - m_1(t), & k = 1 \\ h_{1,k}(t) = h_{1,k-1}(t) - m_k(t), & k \neq 1 \end{cases}$$

This sifting process will be repeated k times until $h_k(t)$ is an IMF. Alternatively, Huang et al. suggested using a Cauchy type of convergence test as the criterion for stopping the sifting process. The test requires the normalized

squared difference between two successive sifting operations, which is defined as (6.2).

$$(6.2) \quad SD_k = \sum_{t=1}^T \frac{|h_{1,k-1}(t) - h_{1,k}(t)|^2}{h_{1,k-1}^2(t)}$$

If SD_k is less than 0.2 (or 0.3 sometimes), set $c_1(t) = h_{1,k}(t)$ and $c_1(t)$ is the first IMF. $c_1(t)$ generally contains a component that has the finest scale or the shortest period of the signal. Separating $c_1(t)$, the residue $r_1(t)$ is obtained as (6.3).

$$(6.3) \quad r_1(t) = x(t) - c_1(t).$$

$r_1(t)$ contains a component with a longer period than $c_1(t)$. When $r_1(t)$ is computed, it will be treated as a new signal, and the sifting process is conducted on $r_1(t)$ to obtain $c_2(t)$ and $r_2(t)$; i.e.,

$$(6.4) \quad r_i(t) = r_{i-1}(t) - c_i(t), i = 2, 3, \dots, n.$$

This procedure will be repeated until any of the following predetermined criteria is satisfied:

- 1) either when the component $c_n(t)$ or residue $r_n(t)$ becomes less than the predetermined value of substantial consequence;
- 2) or when the residue $r_n(t)$ becomes a monotonic function from which no more IMF can be extracted.

The original signal can be reconstructed by summing up all the IMFs and the final residue by (6.5).

$$(6.5) \quad x(t) = \sum_{i=1}^n c_i(t) + r_n(t).$$

Using EMD to denoise a signal is based on the studies that high frequency components has little contribution to model fitting but it has a great disturbance for prediction accuracy. This is because is the most disorder and unsystematic part of the time series and has little regularity (Guo et al., 2012). Therefore, the signal after denoising is computed as (6.6).

$$(6.6) \quad x(t) = \sum_{i=2}^n c_i(t) + r_n(t).$$

6.3 A Noise-tolerant Drift Adaptation

Method—NoA

The noise-tolerant fuzzy c-means based drift adaptation method (NoA) is proposed based on FUZZ-CARE. FUZZ-CARE is an incremental drift adaptation method. To embed the denoise technique in FUZZ-CARE. We design an incremental EMD.

To design the incremental EMD, we need to overcome the problem that the training set probably contains two or more patterns when concept drift appears in a data stream. In this situation, the denoise techniques can not be well applied to remove noise because they may delete important information of a new pattern as noisy information. To avoid that, we propose a set of rules to incrementally activate the denoise process on a subset of the training set.

For a data stream $\{(\mathbf{X}_t, y_t)\}$, NoA is implemented as follows:

- 1) manually set T as the length of the training set, and w as the window size.
- 2) if the length of existing instances is less than T , NoA needs to wait and obtain more data;

- 3) once there are T observed instances, EMD is activated to conduct the noise process, and the denoised data is denoted by $(\mathbf{X}_t^d, y_t^d | t = 1, \dots, T)$;
- 4) train the initial predictor on the training set consisting of $\{(\mathbf{X}_t^d, y_t^d) | t = 1, \dots, T\}$, and this predictor is used to predict following w instances;
- 5) after the true values of $\{y_t | t = T + 1, \dots, T + w\}$ are observed, EMD is activated on these instances $\{(X_t, y_t | t = T + 1, \dots, T + w)\}$ (there are w instances in total), and the denoised data is denoted by $\{(X_t^d, y_t^d | t = T + 1, \dots, T + w)\}$;
- 6) combine $(X_t^d, y_t^d | t = T + 1, \dots, T + w)$ and $(X_t^d, y_t^d | t = 1, \dots, T)$ as the updated training set;
- 7) repeat steps 4)-6).

In this way, each data instance will enter to the EMD filter to “clean” the noisy information and then be delivered to train the drift adaptation predictor. The EMD is separately activated on a small subset of the training set, and therefore is less likely affected by the concept drift problem. For example, assuming we are at 1000th time point, and there will be a sudden real drift in the data stream at 1500th instance, the target is to predict the 2001th instance by the observed 2000 data instances. If we directly apply EMD on the 2000 data instances where there are two patterns, the information from one pattern is very likely identified as noise from the aspect of the other pattern. This induces a bad denoise result on the whole 2000 instances. In contrast, if we apply EMD every 200 instances, the bad denoise result will only appear during denoising the 1400th to 1600th instances. As for other 200 instances, we promise good denoise results because

they only contain single pattern. Another advantage is that denoising a small subset of data can save computational cost.

6.4 Experimental Evaluations

This section, the proposed noise-tolerant fuzzy c-means based drift adaptation method (NoA) is validated on synthetic data where the drift is manually added and the real-world data stream. The effectiveness is validated by the prequential evaluation, where instances are first used to test, and then to train (Bifet et al., 2015). The accuracy is evaluated by the mean absolute error (MAE) (3.21) and root-mean-square error (RMSE) in (6.7) where y_t and \hat{y}_t are the true and evaluated output value separately. An overview of the experiment design is shown in Table 6.2.

$$(6.7) \quad RMSE = \sqrt{\frac{\sum_{t=t_0}^T \hat{y}_t - y_t}{T - t_0}}$$

Table 6.2: Experimental design in this section.

Section	Data	Experimental aim	Main Results
6.4.1	d1	to show denoise process itself may be invalid when concept drift occurs	Table 6.3
6.4.1	d1	to demonstrate NoA can deal with concept drift problem when the data contains signal noise	Table 6.3
6.4.1	d1	to demonstrate that NoA is not a simple combination of the drift adaptation method and the signal denoising method	Table 6.3
6.4.2	d2	to validate the effectiveness of NoA on real-world data streams	Table 6.4

d1: Synthetic noisy data with drift
d2: a real-world regression data stream

6.4.1 Evaluation on synthetic data

Data Description.

Original data. We test NoA on all types of drift with different levels of noise. The original data denoted by **Non-Drift** contains no drift nor noise. Non-Drift is generated in the same manner of the non-drift data generated in Section 3.4.1.

Synthetic data with drift. Based on **Non-Drift**, we generate data streams with different types of drift by the same manner of the synthetic data generated in Section 3.4.1. Here, we briefly list their notation and which types of concept are contained in each synthetic data.

- 1) *Virt-Drift* contains a sudden virtual drift;
- 2) *Sudd-Drift* contains a sudden real drift;
- 3) *Incr-Drift* contains an incremental real drift;
- 4) *Rec-Drift-Grad* has gradual drift and reoccurring concepts;
- 5) *Rec-Drift-Mix* contains incremental drift, sudden drift, and reoccurring concepts.

Synthetic noisy data with drift: Based on the data in **Synthetic data with drift**, we generate data streams with drift and noise by adding different levels of noise to them. The level of noise is controlled by Signal-to-noise ratio (SNR). Four levels of noise are added with SNR= 1, 5, 10 and 20 separately. A smaller SNR indicates stronger noise. Therefore, there are 20 data streams with drift and noise in total. Figure 6.1 presents the original data, data with drift and one group of data with drift and noise where the SNR is 20. In Figure 6.1 the black dots denotes one pattern, and the red dots denotes a different pattern. As the black pattern appears before the red pattern, it is also noted as

the old pattern and the red is noted as the new pattern. For Non-Drift, Virt-Drift, Sudd-Drift and Incr-Drift, the dots are drawn in a 2-dimensional plane with axis of x and y axis; while Rec-Drift-Grad and Rec-Drift-Mix are drawn in a 3-dimensional sphere with an additional axis of time t . In addition, gradual drift occurs in the blue rectangle in 4) Rec-Drift-Grad. It can be seen that after adding noise into data, the pattern becomes less clear than before especially in the data with incremental drift where the noise and drift are difficult to distinguished from each other.

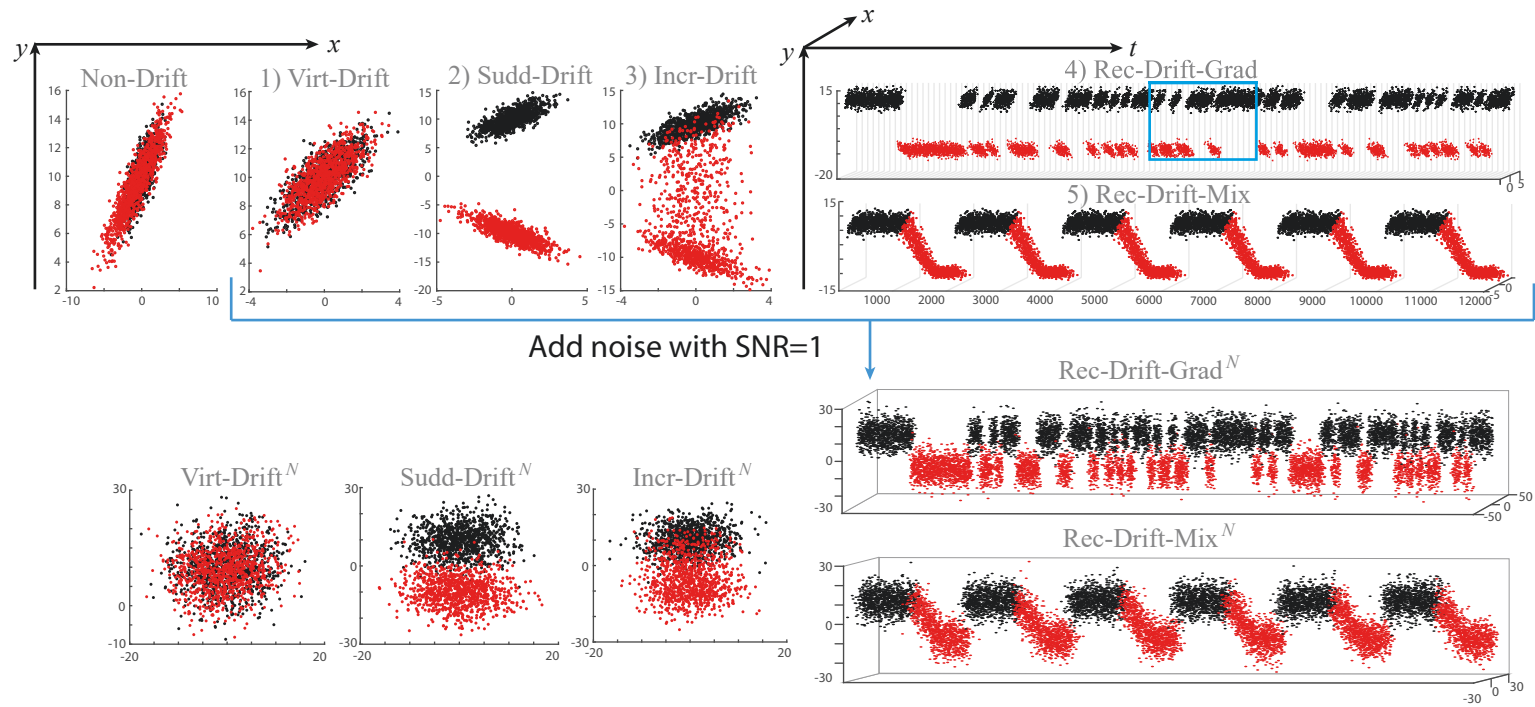


Figure 6.1: Synthetic noisy data stream with different types of drift.

Evaluation results. To validate our proposed method NoA solves drift and noise problem at the same time, the accuracy results of NoA is compared to four baseline methods: the Plain method directly uses the regression predictor in NoA to predict the data stream without involving the drift adaptation process or the denoise process; the Adapt method only contains the drift adaptation process but does not embed the denoise process; the Denoise method only contains denoise process; the DA-com is a simple combination of the drift adaptation and denoise processes where the updated training set is first denoised by the denoise technique and then used to build the regression predictor. It does not specially design a mechanism to enable the denoise process to suit the drift adaptation well; NoA is our noise-tolerant drift adaptation method.

The accuracy results (Table 6.3) indicates the following conclusions:

- 1) **Virtual drift does not affect prediction accuracy in a regression task.** For four Virt-Drift^N data, Denoise and NoA have more accurate prediction than the others. However, adaptation process does not help to get better accuracy.
- 2) **Adaptation process can effectively handle the drift problem in a data stream.** This is concluded because for all Sudd-Drift^N, Incr-Drift^N, Rec-Drift-Grad^N and Rec-Drift-Mix^N data, Adapt and NoA perform better than the other methods.
- 3) **Denoise process is probably invalid in a noisy data stream that has concept drift problem.** Can denoise technique only solve the noisy problem in a data stream? The answer is no according to the experimental

Table 6.3: Evaluation results on synthetic data.

MAE	(SNR)	Plain	Adapt	Denoise	DA-com	NoA
Virt-Drift ^N	20	1.47	1.43	1.65	1.82	1.42
	10	2.16	2.14	2.21	2.20	1.99
	5	3.15	3.12	3.37	3.35	2.87
	1	4.51	4.51	7.01	6.34	4.22
Sudd-Drift ^N	20	11.83	5.07	12.74	4.71	5.00
	10	11.92	5.36	9.11	6.11	5.22
	5	11.80	6.18	12.38	5.40	5.72
	1	12.17	7.52	8.95	6.64	6.48
Incre-Drift ^N	20	8.60	5.46	8.69	5.23	5.38
	10	8.80	5.57	8.54	5.21	5.38
	5	9.07	6.33	8.92	6.79	5.92
	1	9.59	6.74	10.41	7.29	6.44
Rec-Drift-Grad ^N	20	9.00	2.33	83.54	213.72	2.27
	10	9.05	2.98	314.87	125.32	2.79
	5	9.27	4.12	67.19	247.65	3.60
	1	9.63	5.90	121.53	197.10	5.17
Rec-Drift-Mix ^N	20	7.67	3.00	154.27	140.62	2.88
	10	7.85	3.46	76.72	44.70	3.21
	5	8.11	4.30	131.32	84.24	3.81
	1	8.59	5.69	21.25	31.06	4.79

results. The Denoise method is not steadily better than the Plain method, such as the instability in Rec-Drift-Grad^N and Rec-Drift-Mix^N.

- 4) **A simple combination of drift adaptation and denoise methods is not a solution for data stream with both concept drift and noise.** When the drift adaptation and noise processes are simply combined, the accuracy is as low as the Plain method, which reflects the complex dependency between drift adaptation and noise reduction.
- 5) **Our NoA method can solve the concept drift and noise problem**

when they simultaneously appear. The proposed NoA gets the best performance comparing to other baseline methods.

6.4.2 A case study: solar radiation prediction

This is a set of 30-minute interval environment observation data collected from the SMEAR II station which contains 140, 576 data examples of 43 variables from 00:15, on 15 April 2005 to 23:45 on 14 April 2013. The regression task is to predict solar-radiation using 43 variables. Six of the 43 variables are time labels that are not considered prediction features in the model. The remaining environmental features have been introduced in (Žliobaitė et al., 2014b). There are many missing values in this data and we eliminate them in the same way as (Žliobaitė et al., 2014b).

Several popular drift adaptation method specially for regression tasks are introduced as the benchmark. The benchmarks comprise two tree models—FIMT-DD (Ikonomovska et al., 2011a) and ORTO (Ikonomovska et al., 2011b), two rule models—AMR and its ensemble version, metaAMR (Duarte et al., 2016) and the ALL method in (Žliobaitė et al., 2014b). All benchmarks except ALL are implemented by MOA (Bifet et al., 2010a)(<https://moa.cms.waikato.ac.nz/>). The result of ALL is from (Žliobaitė et al., 2014b). The prediction results are shown in Table 6.4. It can be seen that our NoA can not only defeat the general drift adaptation regression method but also defeat the method that is specially designed for SMEAR data.

Table 6.4: Evaluation on the SMEAR data

	metaAMR	FIMTDD	ORTO	AMR	ALL	NoA
RMSE	22.7	364.0	828.1	22.1	19.0	15.6

6.5 Summary

Most of current drift adaptation methods in literature are designed in an assumption that the data stream only has changing data distributions. They omit the fact that problems in a batch-learning setting also exist in a data stream, such as the noisy data problem.

To fill the gap of prediction for noisy data streams with concept drift, we propose a noise-tolerant drift adaptation method (NoA) to simultaneously solve concept drift and noisy data problems in a data stream. We propose to filter the noisy information of data instances before using them to train or update the predictor. This denoise process is designed in an incremental way which enables it to be perfectly embedded in an incremental drift adaptation method.

We test NoA on all types of drift and their mixture with different levels of signal noise. The experimental evaluation result shows good effectiveness of our method on the data stream with both concept drift and noise problems.

DRIFT ADAPTATION WITH TEMPORAL DEPENDENCY

In the regression case of a data stream, the target variable is a time series that is probably autocorrelated, which leads to the temporal dependency problem. However, how a drift adaptation method can be applied for data streams that are neither independent nor identically distributed is rarely discussed in the literature. In this chapter, we discuss the concept drift adaptation in the scenario that a data stream contains both concept drift and temporal dependency. In this chapter, the scenario is introduced in Section 7.1. Section 7.2 lists the definitions and notations. Section 7.3 presents a drift-adapted regression framework for data streams with drift and temporal dependency. The proposed framework is validated by experimental evaluations in Section 7.4. Section 7.5 concludes this chapter.

7.1 Introduction

In the regression case of a data stream, the target variable is now a time series process that is probably autocorrelated, which leads to the temporal dependency problem. How to deal with the problems of temporal dependency and concept drift simultaneously is a big challenge and very few studies could be found that can solve these two problems at the same time.

To fill this research gap, this chapter discusses the concept drift problem in the data stream that have temporal dependency. A novel drift-adapted framework is proposed in regression cases, named DAR. In DAR, we propose to train the predictor on a reconstructed feature space to solve the temporal dependency problem, and an informed adaptation method is proposed to update the trained predictor when each new data instance arrives. The feature space is reconstructed based on the time series identification. In this section, it is proved that testing error decreases faster in a linear predictor trained on the reconstructed space when data streams with concept drift and temporal dependency problems. An local drift degree (LDD^+) statistic is proposed in DAR to select the most relevant data instances to the latest pattern in the training set, and the predictor is continuously updated with the data instances that are most relevant to the latest pattern.

The novelty of the proposed DAR framework are as follows:

- It fills the research gap of handling a data stream with concept drift and temporal dependency problem from the aspect of the time series processes;
- It proves that testing errors decreases faster in a linear predictor trained on the reconstructed space for data streams with concept drift and temporal

dependency. We compare the error decreasing speed of linear predictors trained on the original and reconstructed feature spaces;

- The linear predictor case is generalized to non-linear cases by introducing locally weighted regression;
- We develop a new statistic called LDD^+ to measure the distance from an data instance to a high-dimensional data distribution and apply it to make drift adaptation when every new instance arrives;
- Combining the above aspects, a drift-adapted regression framework (DAR) is proposed for data streams with concept drift and temporal dependency.

7.2 Definitions and Notations

In this chapter, the definition of concept drift is the same to that in Definition 3.3. The data stream will be redefined to present the characteristics of temporal dependency, and learning aim is also redefined based on the new definition of data streams. In this chapter, the data stream is considered to consist of $d + 1$ time series processes.

Definition 7.1 (Time Series). A time series process S_t is a sequence taken at successive equally spaced points in time.

Definition 7.2 (Autocovariance). Autocovariance is an important property of a time series process S_t , which is

$$(7.1) \quad \gamma(\tau) = E[(S_t - \mu_t)(S_{t+\tau} - \mu_{t+\tau})].$$

where μ_t and $\mu_{t+\tau}$ are the expectation of S_t and $S_{t+\tau}$. The autocovariance is the covariance if S_t and $S_{t+\tau}$ are two random variables.

If the time series process is autoregressive, γ depends on τ rather than being a fixed constant (Hamilton, 1994). This means that another basic assumption in conventional machine learning, *independence is also invalid*.

Definition 7.3 (Covariance-stationary and Ergodic for the Mean (Hamilton, 1994)). A time series process S_t is *covariance-stationary* and *ergodic for the mean* if

$$(7.2) \quad \begin{cases} E(S_t) = \mu & \text{for all } t \\ E[(S_t - \mu)(S_{t-\tau} - \mu)] = \gamma_\tau & \text{for all } t \text{ and } \tau \\ (1/T) \sum_{t=1}^T S_t \xrightarrow{P} E(S_t) & \text{as } T \rightarrow \infty. \end{cases}$$

Remark 7.1. *The first two equations denote that neither the mean μ_t nor the autocovariance γ_τ depends on the time t ¹. The third equation guarantees that the time average will eventually converge to the expectation $E(S_t)$.*

Definition 7.4 (Time Series Decomposing). A data stream consists of time series processes. Each time series process in the data stream is considered as weighted sum of two time series process, V_t and S_t .

$$(7.3) \quad T_t = (1 - s)V_t + sS_t, s \in [0, 1]$$

where V_t is a time series process with unknown characteristics which represents the uncertainty aspect of this variable, and S_t process is *covariance-stationary*

¹This does not conflict with (2), γ_τ depends on τ but is independent on t

and ergodic for the mean (see Definition 7.3) which represents the certain temporal dependency. V_t and S_t are assumed to be independent.

Definition 7.5 (Mixed Time Series). Given that $T_t^1, T_t^2, \dots, T_t^n$ are n time series processes (Definition 7.4), and ω_t is the weighting vector satisfying $\sum_{i=1}^n \omega_t^{(i)} = 1$, a mixed time series MT_t is defined as

$$\begin{aligned}
 (7.4) \quad MT_t &= \sum_{i=1}^n \omega_t^{(i)} \times T_t^{(i)} \\
 &= (1-s) \sum_{i=1}^n \omega_t^{(i)} V_t^{(i)} + s \sum_{i=1}^n \omega_t^{(i)} S_t^{(i)} \\
 &= (1-s)MV_t + sMS_t.
 \end{aligned}$$

To describe data streams with drift and temporal-dependence, the data stream is considered to consist of $d + 1$ time series processes, and one variable's values between two consecutive drift points are considered to be a realization of special case of MT_t .

Definition 7.6 (Drift Point). Drift point is defined as the time point when a new concept starts. Namely the $t_{d(i)}$ in Definition 3.3.

Definition 7.7 (Data Stream with Drift and Temporal-dependence).

$$(7.5) \quad DS_t^{(X,y)} = \{MT_t^1, \dots, MT_t^m, MT_t | s, \omega, V, S\}$$

For example, a data stream contains two different patterns in which $y_t = \beta_1 X_t$ before time point t_d and $y_t = \beta_2 X_t$ after t_d , where X_t is a time series. Given

$T_t^{x,1} = T_t^{x,2} = X_t$, $T_t^{y,1} = \beta_1 T_t^{x,1}$, $T_t^{y,2} = \beta_2 T_t^{x,1}$ four time series, $\omega_t^{x,1}$, $\omega_t^{y,1}$ equals 1 when $t < t_d$ while 0 when $t \geq t_d$, and $\omega_t^{x,2}$, $\omega_t^{y,2}$ equals 0 when $t < t_d$ while 1 when $t \geq t_d$. Given $MT_t^x = \omega_t^{x,1} T_t^{x,1} + \omega_t^{x,2} T_t^{x,2}$ and $MT_t^y = \omega_t^{y,1} T_t^{y,1} + \omega_t^{y,2} T_t^{y,2}$, this data stream is $\{MT_t^x, MT_t^y\}$.

Remark 7.2. Clearly, X_t and y_t themselves are time series but here we use a linear combination of time series to represent them because if they are considered as a single time series, their statistical properties are totally unknown. The definition of mixed time series helps to abstract the regular components in X_t and y_t as MS_t and leave the chaos components as MV_t .

So far, we have given the definition of a data stream with temporal dependency and concept drift. The drift is presented in a data stream $DS_t^{(X,y)}$ if $\omega_t \neq \omega_{t+1}$ for some t , and the temporal dependency is presented by the time series T . In this study, we provide a solution for finding the unbiased estimation for MS_t . However, estimation on MV_t will not be discussed, because the characteristics of MV_t are unknown in our assumption. Therefore, for an arbitrary data stream $DS_t^{(X,y)}(s, \cdot)$, a bigger s means a better estimation by our method. In the following discussion, the problem will be simplified to a special case of $s = 1$, namely $MT_t^{(X,y)} = MS_t^{(X,y)}$.

For a data stream with drift and temporal dependency, each time series process $MS_t^{(i)}$ is assumed to be a p th-order autoregressive process denoted by $AR(p)$.

Definition 7.8 (p th-order Autoregressive Process). A p th-order autoregressive process X_t satisfies

$$(7.6) \quad X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \dots + \phi_p X_{t-p} + \epsilon_t,$$

where X_{t-1}, \dots, X_{t-p} are lag orders (earlier observations) of X_t and ϵ_t is the white noise sequence (Definition 7.9).

Definition 7.9 (White Noise Sequence). A white noise sequence is a sequence $\{\epsilon_t\}_{t=-\infty}^{\infty}$ satisfying:

$$(7.7) \quad E(\epsilon_t) = 0, E(\epsilon_t^2) = \sigma^2 \text{ and } E(\epsilon_t, \epsilon_\tau) = 0 \text{ for } t \neq \tau.$$

Definition 7.8 can be rewritten by $\sum_{i=0}^p \mathcal{L}^{(i)} X_t$ by introducing the *Lag operator* \mathcal{L} that for any integer k , $\mathcal{L}^k X_t = X_{t-k}$. Referring to Definition 3.5, the learning process of the data stream $DS_t^{(\mathbf{X}, y)}$ will be as Definition 7.10.

Definition 7.10 (Learning Aim for Data Streams with Drift and Temporal-dependence).

$$(7.8) \quad \min_{h_1, h_2, \dots, h_t, \dots} \sum_t \ell(h_t, \mathbf{X}, y, \mathcal{L} | (\mathbf{X}, y) \sim p_t(\mathbf{X}, y)).$$

7.3 Drift Adaptation with Temporal Dependency

In this section, we will discuss how to find a better solution to each h_t in (7.8) step by step (Section 7.3.1). After that, a drift-adapted regression framework is described in details (Section 7.3.2).

7.3.1 Analysis of testing error when the real drift exists

In this section, we will discuss how to find a better solution to each h_t in (7.8) step by step. When the learning aim is to obtain less error and a squared loss is

considered in the regression task, the concept drift problem focuses on real drift. To prove this, we introduce Theorem 7.1.

Theorem 7.1. *The estimation with smallest squared loss is the expectation of y conditional on \mathbf{X} : $\hat{y} = h(\mathbf{X}) = E(y|\mathbf{X})$ where $h = \underset{h \in \mathcal{H}_{reg}}{\operatorname{argmin}} \ell(h, \mathbf{X}, y)$.*

Proof. The data stream $DS_t^{(\mathbf{X}, y)}$ consists of the attribute variable \mathbf{X} and the label variable y . Consider basing the estimated y (denoted by \hat{y}) on any predictor in the hypothesis set is $h(\mathbf{X})$ other than the conditional expectation, the loss would be

$$\begin{aligned} E[y - h(\mathbf{X})]^2 &= E[y - E(y|\mathbf{X}) + E(y|\mathbf{X}) - h(\mathbf{X})]^2 \\ &= E[y - E(y|\mathbf{X})]^2 + E[E(y|\mathbf{x}) - h(\mathbf{X})]^2 \\ &\quad + 2E\{[y - E(y|\mathbf{X})][E(y|\mathbf{X}) - h(\mathbf{X})]\}. \end{aligned}$$

Let $\eta \equiv [y - E(y|\mathbf{X})][E(y|\mathbf{X}) - h(\mathbf{X})]$. As the terms $E(y|\mathbf{X})$ and $h(\mathbf{X})$ are known constants under the condition of \mathbf{X} , $E([E(y|\mathbf{X}) - h(\mathbf{X})]|\mathbf{X}) = E(y|\mathbf{X}) - h(\mathbf{X})$. The expectation of η conditional on \mathbf{X} can be written into:

$$\begin{aligned} E(\eta|\mathbf{X}) &= [E(y|\mathbf{X}) - h(\mathbf{X})]E([y - E(y|\mathbf{X})]|\mathbf{X}) \\ &= [E(y|\mathbf{X}) - h(\mathbf{X})] \times 0 = 0. \end{aligned}$$

According to the law of iterated expectations, $E[\eta] = E(E[\eta|\mathbf{X}]) = 0$. Substituting

$E[\eta]$ back into $E[y - h(\mathbf{X})]^2$ gives

$$E[y - h(\mathbf{X})]^2 = E[y - E(y|\mathbf{X})]^2 + E([E(y|\mathbf{x}) - h(\mathbf{X})]^2).$$

On the right side of the above equation, the first term does not depend on $h(\mathbf{X})$, and the second term cannot be made smaller than 0. Therefore the predictor that makes $E[y - h(\mathbf{X})]^2$ as small as possible satisfies $E([E(y|\mathbf{x}) - h(\mathbf{X})]^2) = 0$, namely $h(\mathbf{X}) = E(y|\mathbf{X})$. ■

Assuming the hypothesis set contains the optimal predictor $E(y|\mathbf{X})$, if $p(\mathbf{X})$ (virtual drift) changes but $p(y|\mathbf{X})$ (real drift) stays the same, the current predictor can be the same as the previous predictor because $E(y|\mathbf{X})$ has not changed. Therefore, the virtual drift is omitted in the following discussion. Next, we will discuss how to effectively build and update the predictor when real drift ($E(y|\mathbf{X})$ changes) occurs.

7.3.1.1 The linear case

We first consider the simplest case there is only one attribute variable denoted by X_t , and y_t is the label variable; y_t and X_t are linearly correlated, and X_t is a first-order autoregressive process. The sudden drift occurs at time point t_d .

In this linear case, $X_t = \beta_0 + \beta_1 X_{t-1} + \epsilon_t$, $|\beta_1| < 1$, and $y_t = \theta_0 + \theta_1 X_t + \epsilon_t$ (pattern 1) before time point t_d , $y_t = \theta'_0 + \theta'_1 X_t + \epsilon_t$ (pattern 2) after t_d where $\theta'_0 \neq \theta_0$ and $\theta'_1 \neq \theta_1$. Clearly, $\{X_t, y_t\}$ is a data stream with drift and temporal dependency in which a real drift occurring at t_d , and X_t is a first-order autoregressive process. According to Definition. 7.7, this data stream can be rewritten as $DS_t^{(X,y)} =$

$\{MT_t^X, MT_t^Y\}$, where MT_t^x is:

$$(7.9) \quad \begin{aligned} MT_t^x &= S_t^x \\ S_t^x : X_t &= \beta_0 + \beta_1 X_{t-1} + \epsilon_t \end{aligned}$$

and MT_t^y is:

$$(7.10) \quad \begin{aligned} MT_t^y &= \omega_t^1 S_t^{y,1} + \omega_t^2 S_t^{y,2} + \omega_t^3 S_t^{y,3} \\ S_t^{y,1} : y_t &= \theta_0 + \theta_1 \beta_0 - \theta_0 \beta_1 + \beta_1 y_{t-1} + \epsilon_t \\ S_t^{y,2} : y_t &= \theta'_0 + \theta'_1 \beta_0 - \theta'_0 \beta_1 - \frac{\theta'_0 \theta'_1 \beta_1}{\theta_1} y_{t-1} + \epsilon_t \\ S_t^{y,3} : y_t &= \theta'_0 + \theta'_1 \beta_0 - \theta'_0 \beta_1 + \beta_1 y_{t-1} + \epsilon_t \end{aligned}$$

$$\omega_t = \begin{cases} (1, 0, 0) & t < t_d \\ (0, 1, 0) & t = t_d \\ (0, 0, 1) & t > t_d \end{cases}$$

There are two ways to estimate y_t : a) The traditional approach is to find the optimal predictor in the hypothesis set $\mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$; b) the optimal hypothesis can also be found in the hypothesis set $\mathcal{H} : \mathcal{Y} \rightarrow \mathcal{Y}$ to estimate y_t . Next, we will discuss the difference between these two approaches when drift occurs.

The estimations of θ or β and y_t computed by a) and b) before t_d are unbiased, consistent, and efficient estimations if the ordinary least squares (OLS) method is used. Similarly, the observations after viewing enough instances of the new pattern comprise the new training set, and the estimations of θ' and y_t by a) or b) are also unbiased, consistent, and efficient. *The difference between a) and b) is represented when the training set mixes data from the old pattern (pattern 1) and the new pattern (pattern 2).* Assume that the training set contains n observations of (X_t, y_t) , where $n_1 = n - n_2$ of them are from the old pattern, and n_2 of them are from the new pattern. A predictor is trained with this training set to estimate future y_t which follows the new pattern.

Theorem 7.2. *Given a data stream $DS_t^{(X,y)} = \{MT_t^X, MT_t^y\}$ where MT_t^X and MT_t^y are as presented in (7.9) and (7.10) separately, the testing error linearly decreases to 0 if using $\mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$ as the hypothesis set.*

Proof. According to OLS, the estimation of θ' by the n -size training set is as follows:

$$(7.11) \quad \begin{cases} \hat{\theta}'_1 = \frac{(1-r)\overline{Xy_1} + r\overline{Xy_2}}{\overline{X^2}} \\ \hat{\theta}'_0 = (1-r)\overline{y_1} + r\overline{y_2} - \hat{\theta}'_1\overline{X} \end{cases}$$

where $\overline{Xy_1} = \frac{1}{n_1} \sum_{t < t_d} Xy$, $\overline{Xy_2} = \frac{1}{n_2} \sum_{t \geq t_d} Xy$ and $r = n_2/n$ (the inference of estimating θ' is given in A.2). Based on $\hat{\theta}'$, the estimation of an unknown y_u

is:

$$(7.12) \quad \hat{y}_u = \hat{\theta}'_0 + \hat{\theta}'_1 X_u$$

An unbiased estimation of θ is obtained by n_2 observations from the new pattern, that is:

$$(7.13) \quad \tilde{\theta}'_1 = \frac{\overline{X} y_2}{\overline{X^2}}, \tilde{\theta}'_0 = \overline{y_2} - \tilde{\theta}'_1 \overline{X}$$

Based on $\tilde{\theta}$, the unbiased estimation of an unknown y_u is:

$$(7.14) \quad \tilde{y}_u = \tilde{\theta}'_0 + \tilde{\theta}'_1 X_u, \text{ and } E(\tilde{y}_u - y_u) = 0$$

Therefore, the testing error of \hat{y}_u is:

$$\begin{aligned} e_u &= \hat{y}_u - \tilde{y}_u \\ &= (1-r)(\overline{y_1} - \overline{y_2}) + \frac{(1-r)(\overline{X} y_1 - \overline{X} y_2)}{\overline{X^2}} (X_u - \overline{X}) \\ (7.15) \quad &= (1-r) \left[(\overline{y_1} - \overline{y_2}) + \frac{\overline{X} y_1 - \overline{X} y_2}{\overline{X^2}} (X_u - \overline{X}) \right] \\ &= (1-r) \left[(\tilde{\theta}_0 - \tilde{\theta}'_0) + (\tilde{\theta}_1 - \tilde{\theta}'_1) X_u \right] \end{aligned}$$

$$(7.16) \quad E_{X,y}(e_u) = (1-r)E \left[(\tilde{\theta}_0 - \tilde{\theta}'_0) + (\tilde{\theta}_1 - \tilde{\theta}'_1) X_u \right]$$

As the coefficient of $1-r$ is considered to be constant, $\lim_{r \rightarrow 1} E(e_u) = 0$, and $E(e_u)$ linearly decreases to 0 as more instances from the new pattern are included in the training set (r goes to 1). ■

Theorem 7.3. *Given a data stream $DS_t^{(\mathbf{X}, \mathcal{Y})} = MS_t^{(\mathbf{X}, \mathcal{Y})}$, the testing error exponentially decreases to 0 if using $\mathcal{H} : \mathcal{Y} \rightarrow \mathcal{Y}$ as the hypothesis set.*

Proof. The relationship between Y_{t-1} and Y_t given $t \in (t_d, n]$ can be rewritten by the following difference equation:

$$(7.17) \quad y_t = c + \phi y_{t-1} + \epsilon_t,$$

where ϵ_t is a white noise sequence defined in (7.9). As y_t is covariance-stationary, $|\phi| < 1$, and the stable solution to (7.17) is:

$$(7.18) \quad y_t = c \frac{1 - \phi^\infty}{1 - \phi} + \sum_{\tau=0}^t \phi^{t-\tau} \epsilon_{t-\tau} + \phi^\infty y_{-\infty}.$$

Similarly, $y_{t-1} = c \frac{1 - \phi^\infty}{1 - \phi} + \sum_{\tau=0}^{t-1} \phi^{t-1-\tau} \epsilon_{t-1-\tau} + \phi^\infty y_{-\infty}$, and $y_t - y_{t-1} = \phi^t \epsilon_t$. Given $e_u = y_t - y_{t-1} = \phi^{nr} \epsilon_{nr}$, it exponentially decreases to 0 as r increases. ■

As the new instances arrive and are included in the training set, the r will finally increase to 1. Clearly, the error decreases faster under the condition of Theorem 7.3 (exponentially decreases to 0) than that under the condition of Theorem 7.2 (linearly decreases to 0).

The conclusion of the unary case above is also suitable in the multiple linear case. If no collinearity exist in the multiple case, θ_i for $X^{(i)}$ (i th dimension of \mathbf{X}) is similarly computed to (7.11) but Xy needs to subtract $\mathbf{X}y$ of other dimensions. The y_t process still converges to a constant by constraint (7.2). Therefore, after drift occurs, as the training set contains more instances of the new pattern, the testing error of the predictor trained with the training set on the reconstructed space $(\mathcal{L}, \mathcal{Y})$ decreases faster than that of the predictor trained on the original space $(\mathcal{X}, \mathcal{Y})$.

7.3.1.2 A general case

The conclusion of the linear case can be applied in a general case by introducing locally weighted regression that the predictor trained on the reconstructed space is better than the predictor trained on the original space even \mathbf{X} and y have non-linear correlation. *The locally weighted regression is an existing method, which is not the contribution in this study. However, it helps us to widen the conclusion in the linear case to a general case.*

The learning goal of locally weighted regression at each target point \mathbf{X}_u is represented as follows:

$$(7.19) \quad \min_{\boldsymbol{\theta}(\mathbf{X}_u)} \sum_{t=1}^N K_k(\mathbf{X}_u, \mathbf{X}_t) (y_t - \boldsymbol{\theta}^T \mathbf{X}_u)^2,$$

where

$$(7.20) \quad K_k(\mathbf{X}_u, \mathbf{X}_t) = D \left(\frac{|\mathbf{X}_t - \mathbf{X}_u|}{b_k(\mathbf{X}_u)} \right).$$

$K_k(\mathbf{X}_u, \mathbf{X}_t)$ is the weight determined by the distance from \mathbf{X}_u to the k -nearest neighborhoods of \mathbf{X}_u , $b_k(\mathbf{X}_u)$, and D is defined as:

$$(7.21) \quad D(d) = \begin{cases} 1 & d \leq b_k(\mathbf{X}_0) \\ 0 & d < b_k(\mathbf{X}_0) \end{cases}$$

To build a predictor based on the current training set (\mathbf{X}_t, y_t) to estimate y_u , we first find the k -nearest neighborhoods of (\mathbf{X}_u, y_u) , and learn a linear predictor on these neighborhoods. y_u is estimated by this local predictor. This transforms the general case to a sum of linear cases which have been discussed in Section 7.3.1.1.

Remark. For the data stream with concept drift and temporal dependency $DS_t^{(\mathcal{X}, \mathcal{Y})}$, this subsection gives a theoretical conclusion that when the training set mixes data from two patterns, the error of a predictor built on the reconstructed space $(\mathcal{L}, \mathcal{Y})$ decreases faster than that of a predictor built on the original space $(\mathcal{X}, \mathcal{Y})$ as the training set contains more instances from the new pattern.

7.3.2 Drift adaptation procedure in DAR

Section 7.3.1 concludes that the predictor built on the reconstructed space is better than the predictor built on the original space for the data stream with concept drift and temporal dependency. Once the space is reconstructed, the next problem is how to build and update the predictor to adapt the newest pattern. This section will discuss this the drift adaptation procedure used in DAR.

When the training set mixes with instances of different patterns, we consider the pattern of the upcoming future instances to be the new pattern. According to the conclusion in Section 7.3.1, the more new pattern instances are included, the more accurate it is to use a predictor built on this training set to estimate future instances. Therefore, updating the training set is also an important process for data streams with concept drift. If none of the instances in the training set follows the new pattern, it is impossible for any method to train an effective predictor to estimate future values. In this paper, we propose to update the training set based on the local drift degree.

Local drift degree (LDD), proposed by Liu et al., is a statistic to quantify regional discrepancies between two different sample sets Liu et al. (2017a). Given Δ_1 and Δ_2 two $m + 1$ -dimension populations from space \mathbb{R}^{m+1} , two samples of

Δ_1 and Δ_2 , δ_1 and δ_2 , consist of instances from Δ_1 and Δ_2 respectively. As it is impossible to acquire all instances in Δ_1 and Δ_2 , δ_1 and δ_2 are used to infer by statistical theories whether Δ_1 and Δ_2 have the same distribution. If Δ_1 and Δ_2 have the same distribution, the number of instances belonging to δ_1 and δ_2 in any arbitrary subspace $\omega \subset \mathbb{R}^{m+1}$ are theoretically the same, which leads to an insignificant discrepancy between the number of instances belonging to δ_1 and δ_2 . If Δ_1 and Δ_2 have different distributions, uneven density exists in at least one subspace. Based on the above idea, LDD is defined in (7.22):

$$(7.22) \quad d_\omega = \frac{|\delta_{\omega_1}|/n_{\delta_1}}{|\delta_{\omega_2}|/n_{\delta_2}} - 1$$

where $|\delta_{\omega_1}|$ and $|\delta_{\omega_2}|$ represents the number of instances in ω belonging to δ_1 and δ_2 , and n_{δ_1} and n_{δ_2} are the sample size of δ_1 and δ_2 respectively.

When δ_1 is the current training set, and δ_2 is the newly arrived data instances, LDD can be used to measure the distribution difference of the training set and the newly arrived batch. If LDD is larger than a statistical threshold, the newly arrived instances are considered to have a different distribution from the training set, which denotes a drift. The original LDD assumes that n_{δ_2} is very large so that $|\delta_{\omega_2}|/n_{\delta_2}$ can be considered as a constant. However, this is not always true. To overcome this drawback, we use LDD^+ instead of LDD. LDD^+ is computed as follows:

$$(7.23) \quad d_\omega^+ = \frac{|\delta_{\omega_1}|}{n_{\delta_1}} - \frac{|\delta_{\omega_2}|}{n_{\delta_2}}$$

where $|\delta_{\omega_1}|$, $|\delta_{\omega_2}|$, n_{δ_1} and n_{δ_2} have the same meaning as in (7.22).

Theorem 7.4. *Given δ_1 and δ_2 have the same distribution, $d_\omega^+ \sim N(0, S_{\delta_1}^2/n_{\delta_1} + S_{\delta_2}^2/n_{\delta_2})$, where $S_{\delta_1}^2$ and $S_{\delta_2}^2$ are the sample variances, and n_{δ_1} and n_{δ_2} are the sample size.*

Proof. Define $\delta_1^{(i)}$ as (7.24), and define $\delta_2^{(i)}$, $\Delta_1^{(i)}$ and $\Delta_2^{(i)}$ in the same way.

$$(7.24) \quad \delta_1^{(i)} = \begin{cases} 1 & \text{ith point of } \delta_1 \text{ is in } \omega \\ 0 & \text{otherwise} \end{cases}$$

d_ω^+ can be written as $d_\omega^+ = \sum_i \delta_1^{(i)} - \sum_j \delta_2^{(j)} = \bar{\delta}_1 - \bar{\delta}_2$. In Liu et al. (2017a), it has been proved that $E(\bar{\delta}_1) = \bar{\Delta}_1$ and $E(\bar{\delta}_2) = \bar{\Delta}_2$. Therefore, when no drift occurs, $E(d_\omega^+) = 0$. As data points from δ_1 and points from δ_2 are independent, $var(d_\omega^+) = var(\bar{\delta}_1) + var(\bar{\delta}_2)$. To compute $var(\delta_1)$ and $var(\delta_2)$, we introduce a random variable I_i ,

$$(7.25) \quad I_i = \begin{cases} 1 & \Delta_1^{(i)} \in \delta_1 \\ 0 & \text{otherwise} \end{cases}.$$

$var(\delta_1)$ can be rewritten as:

$$(7.26) \quad \begin{aligned} var(\delta_1) &= var\left(\frac{1}{n_{\delta_1}} \sum_i I_i \Delta_1^{(i)}\right) \\ &= \frac{1}{n_{\delta_1}^2} \left(\sum_i (\Delta_1^{(i)})^2 var(I_i) + 2 \sum_{i \neq j} \Delta_1^{(i)} \Delta_1^{(j)} cov(I_i, I_j) \right) \end{aligned}$$

Considering select n units from N units, the probability that each unit will be selected in n draws is $C(N-1)^{(n-1)}/C_N^n = n/N$ and the probability that two units

will be selected in n draws is $n(n-1)/N(N-1)$. Under this condition, I_i satisfies the following equations:

$$(7.27) \quad \begin{cases} E(I_i) = \frac{n}{N} = f, \\ \text{var}(I_i) = \frac{n}{N} \frac{N-n}{N} = f(1-f), \\ \text{cov}(I_i, I_j) = E(I_i I_j) - E(I_i)E(I_j) = -\frac{f(1-f)}{N-1}. \end{cases}$$

Based on this, $\text{var}(\delta_1)$ is computed as:

$$(7.28) \quad \begin{aligned} \text{var}(\delta_1) &= \frac{f(1-f)}{n_{\delta_1}^2} \left[\sum_i (\Delta_1^{(i)})^2 - \frac{2 \sum_{i \neq j} \Delta_1^{(i)} \Delta_1^{(j)}}{n_{\Delta_1} - 1} \right] \\ &= \frac{f(1-f)}{n_{\delta_1}^2} \left[\sum_i (\Delta_1^{(i)})^2 + \frac{\sum_i (\Delta_1^{(i)})^2}{n_{\Delta_1} - 1} - \frac{(\sum_i \Delta_1^{(i)})^2}{n_{\Delta_1} - 1} \right] \\ &= \frac{1-f}{n_{\delta_1} n_{\Delta_1}} \left(\frac{n_{\Delta_1}}{n_{\Delta_1} - 1} \sum_i (\Delta_1^{(i)})^2 - \frac{(\sum_i \Delta_1^{(i)})^2}{n_{\Delta_1} - 1} \right) \\ &= \frac{1-f}{n_{\delta_1} (n_{\Delta_1} - 1)} \sum_i (\Delta_1^{(i)} - \bar{\Delta}_1)^2 = \frac{S_{\Delta_1}^2}{n_{\delta_1}} (1-f) = \frac{S_{\delta_1}^2}{n_{\delta_1}}. \end{aligned}$$

So far, the expectation and variance of d_{ω}^+ has been obtained. As ω is an arbitrary

Algorithm 7.1: Computation of LDD^+

Input : DS : the current training set
 DS_w : the newly arrived batch of data
 k : the number of nearest neighbors

Output: ldd^+

```

1 for  $t = 1$  to  $|DS_w|$  do
2    $B = [DS, DS_w]$ ;
3    $(knn_1, \dots, knn_{k+1}) = knnsearch(DS_w, B, k + 1)$ ; %find  $k + 1$  nearest
   neighbors of  $DS_w$ ,  $|\cdot|$  computes the cardinality
4    $n_0 = |n_{i \in (1, k+1)} \in DS|$ ; %the number of neighbors in  $DS$ 
5    $n_1 = |n_{i \in (1, k+1)} \notin DS|$ ; %the number of neighbors in  $DS_w$ 
6    $ldd^+(t) = n_0/|DS| - n_1/|DS_w|$ 
7 end
8 return  $ldd^+$ 

```

subset in \mathbb{R}^{m+1} , each d_{ω}^+ can be seen as an element from a simple random sample. Therefore, $var(d_{\omega}^+)$ obeys a normal distribution based on the central limit theorem. ■

The original version of LDD is applied in a classification task where $|\delta_{\omega}|$ is computed based on the L2 norm of feature vectors given the same label that is $d(k) = \|\mathbf{X}_k - \mathbf{X}_0\|_2^2$ when $y_k = y_0$ and $d(k) = \infty$ when $y_k \neq y_0$. In the regression task, KNNs are computed based on the distance of $\|\mathbf{Z}_k - \mathbf{Z}_0\|_2^2$ where $\mathbf{Z} = (\mathbf{X}, y)$ in this paper ². The computation process of LDD^+ is given in Algorithm 8. The returned LDD^+ values represent the relevance of instances in DS to the distribution of DS_w . A larger LDD^+ denotes that this instance is less similar to the current pattern.

- The general procedure and pseudocode of DAR

² \mathbf{Z} can also be in other forms of combinations of \mathbf{X} and y , which may improve the prediction accuracy.

Algorithm 7.2: The drift-adapted regression framework (DAR framework)

Input : DS_0 : the historical data, the initial training set
 DS_w : the newly arrived batch of data
 k : the number of nearest neighbors
 θ : required parameters in *LWR*
 w : $w = |DS_w|$

Output: $\hat{y}_t, t = T + 1, \dots, T + w$ %the estimated value

```

1 Process 1 begin
2   for  $i = 1 : d + 1$  do
3     for  $j=1:3$  % the max lag order is 3 do
4       | compute  $AIC(j, DS_0)$ 
5     end
6      $p(i) = \underset{j}{\operatorname{argmin}} AIC(j, DS_0)$  % determine the best lag order
7   end
8 end
9 % now the input contains the lag orders Process 3 begin
10   $\mathbf{X} = []; y = []$ 
11  for  $i = T : T + w$  do
12    |  $learnset = trainset$ 
13    | if  $isempty(ldd^+) \neq True$  then
14      |  $DeleteIndex = ldd^+(ldd^+(1 : w) > w)$ 
15      |  $learnset>DeleteIndex = []$ 
16    | end
17    | Process 2 begin
18      |  $\hat{y}_i = LWR(learnset, \mathbf{X}_i, \theta, k)$ 
19      |  $\mathbf{X} = [\mathbf{X}; \mathbf{X}_i]; y = [y; y_i]$ 
20    | end
21  end
22   $driftinsts = [\mathbf{X}, y]$ 
23   $driftbase = trainset$ 
24   $ldd^+ = LDD^+(driftinsts, driftbase)$ 
25   $trainset = [trainset(w : end); (\mathbf{X}, y)]$ 
26 end
27 return  $\hat{y}_t, t = T + 1, \dots, T + w$ 

```

According to the conclusion in Section 7.3.1, training a predictor on the reconstructed space is more effective than training it on the original space. One key problem is how to reconstruct the feature space, namely how to identify the p in (7.8). A larger p means more lag orders are involved in the predictor. If a large p is applied, the predictor will be complex and may induce the sparsity problem. However, valuable information is ignored if p is too small. The principle of p identification is to include orders that strongly affect the current state under the condition that the predictor will not be too complex.

p could be a pre-assigned parameter, and tuning can be introduced to determine the value of p . In this paper, we use the Akaike information criterion (AIC) Sakamoto et al. (1986) to identify the lag order p . For each data stream, the identification of p is conducted on the historical data at the beginning. After that, the determined p will be used for this data stream without change.

The AIC of a p -th autoregressive predictor trained on a N -size data sample is computed as:

$$(7.29) \quad AIC(p) = 2k - 2\ln(\hat{\ell}(p)),$$

where k is the number of free parameters to be estimated (for example, here it is $p+1$), and ℓ is the likelihood function to estimate the parameter vector $\boldsymbol{\phi} = (c, \phi_1, \dots, \phi_p)$ in (7.8).

$$(7.30) \quad \ell(p) = f_{Y_p, \dots, Y_1}(y_p, \dots, y_1; \boldsymbol{\phi}) \times \prod_{t=p+1}^N f(y_t | y_{t-1}, \dots, y_{t-p}; \boldsymbol{\phi}).$$

For an autoregressive process as in (7.8) where p is unknown but assumed to be less than a pre-assigned value P , using AIC to determine p is to find $p \in (1, P)$

with minimum AIC from a collection of predictor $AR(1), \dots, AR(P)$.

$$(7.31) \quad \tilde{p} = \underset{p \in (1, P)}{\operatorname{argmin}} \operatorname{AIC}(p)$$

The time series identification of each feature is conducted on the historical data. Once the value of p is determined, it will not change. The maximum p is 3 in this paper. We reconstruct the original space to $(\mathcal{X}, \mathcal{Y}, \mathcal{L})$ using this process.

The DAR framework is summarized in Algorithm 9. Before a new batch of instances arrives, the historical data $\{DS_t^{(X, Y)} | t = 1, \dots, T\}$ is the initial training set. During Process 1, the lag orders, X_{t-p}, y_{t-p} are added to the system as new attributes. Process 2 conducts the locally weighted regression algorithm as given in Algorithm A.1 in A.3. The rules for adaptation are as follows: 1. The new incoming batch of w instances represents the newest pattern, so they will be added to the input set during updating and the oldest w instances in the input set will be deleted; 2. The relevance of old instances to the new pattern is measured by ldd^+ . 3. If the remaining $(T - w)$ old instances in the input set have larger ldd^+ than the w th largest ldd^+ , they will also be deleted from the training set. This process is realized in Process 3 in the DAR algorithm.

7.4 Experimental Evaluations

In this section, the effectiveness of the proposed DAR will be proved on both synthetic data and real data. DAR is validated on three aspects: 1) DAR on 1-dimensional linear case. This corresponds to the theoretical conclusion in Section 7.3.1.1; As 1-dimensional case is available for graphic presentation, the error will be given for every tested instance to show that the error decreases faster on

the reconstructed space and therefore improve the adaptation performance; 2) DAR on multi-dimensional non-linear cases. This corresponds to the theoretical conclusion in Section 7.3.1.2. It also validate our assumption that DAR performs better if the data stream $DS_t^{(X,y)}$ (Definition 7.7) has larger s ; 3) DAR is compared to other drift adaptation methods. DAR uses different parameters for synthetic data and real-world data. They will be specified in each subsection. The organization of the experiments are listed in Table 7.1. Two criteria are used for evaluation, mean absolute error (MAE) and mean absolute percentage error (MAPE).

Table 7.1: Experimental Design

Section	Data	Experimental aim	Main Results
Section 7.4.1	d1	Validate the error decreases faster on a restructured space on simple linear cases (corresponding to Section 7.3.1.1)	Table 7.3, Figure 7.1
Section 7.4.1	d2	Validate DAR on multi-dimensional non-linear cases (corresponding to Section 7.3.1.2)	Table 7.4, Table 7.5, Figure 7.2
Section 7.4.2	d3	Compare DAR with the-state-of-the-art drift adaptation methods	Table 7.6
Section 7.4.3	d3	Statistical test for comparison between methods	Table 7.7

d1 contains three data generated in Section 7.4.1: 1-dimensional linear data
d2 contains six data generated in Section 7.4.1: multi-dimensional non-linear data
d3 real-world data

7.4.1 Experiments on synthetic data

Data Description.

- **Simple linear data (the feature variable is 1-dimensional)**

Three data were generated by several parameter-changing linear models. The data is generated in the same manner as is in Section 3.4.1. *Non-Drift* is a data stream with no drift; *Sudd-Drift* contains a real sudden drift; *Incr-Drift* means that real incremental drift occurs over a period.

- **Multi-dimensional non-linear data (only considering sudden drift)**

The multi-dimensional non-linear data is generated by referring to the python package scikit-learn package (2007) and the paper Friedman (1991). The original data is not used for validate concept drift problem. In this paper, we use the similar mapping functions to the mapping functions used in scikit-learn package (2007). The drift is added by use a negative label variable after the drift point, and the time dependency is added by using autoregressive feature variables instead of the temporally independent feature variables in the original version. Six data streams are generated, and the details of generation process are explained below:

Step 1: Generate five feature variables $X_{1,t}, X_{2,t}, \dots, X_{5,t}$, and each of the feature variable is generated in the same way as X_t in *Non-Drift*. Namely, we have five AR(2) time series.

Step 2: Normalize each feature variable generated in Step 1. This is because the mapping function in scikit-learn package (2007) requires the feature variables on the interval $[0, 1]$.

Step 3: Generate the label variable using the mapping function in (7.32). Six groups of parameters are used for generating six data. They are listed in Table 7.2.

$$(7.32) \quad y_t = \theta_1 \sin(\pi X_{1,t} X_{2,t}) + \theta_2 (X_{3,t} - 0.5)^2 + \theta_3 X_{4,t} + \theta_4 X_{5,t} + \theta_5 \epsilon_t$$

Step 4: Drift is added by letting $y_{t>1000} = -y_{t>1000}$. Clearly, a sudden drift occurs at $t = 1001$ for all these six data.

Table 7.2: Parameters for generating multi-dimensional non-linear data in (7.32)

	θ_1	θ_2	θ_3	θ_4	θ_5
Para0	10	20	10	5	1
Para1	10	10	15	10	40/35
Para2	5	10	15	15	40/35
Para3	1	2	15	15	32/35
Para-1	10	20	2	1	23/35
Para-2	10	20	0	0	20/35

Remark. This mapping function presents a non-linear relationship of polynomial and sine transforms where θ_1 and θ_2 control their weights separately. In addition, it contains linear relationships where θ_3 and θ_4 control their weights separately. Para0 is the parameters used in scikit-learn package (2007). Compared to Para0, Para1, Para2 and Para3 have more weights on the linear relationship while Para-1 and Para-2 have less weights on the linear relationship. Para-2 does not have the term of a regular temporally dependency as θ_3 and θ_4 are 0. We design these parameters to validate the claim that our method will perform better with a larger s in $DS_t^{(X,y)}$ in Definition 7.7. The value of θ_5 is not 1 except for Para0. This is because the max absolute values of y in other data are not the same to that in Para0, which is 35. Therefore, θ_5 changes to alleviate

the difference caused by the disturbance when the estimation results on these six data are compared.

Analysis of Results on Synthetic Data. *Preassigned parameters.* In the experiments of synthetic data. In the experiments, the first 500 instances were set as the historical data for the first training, namely $N = 500$. The max lag order P is 3, the length of the windows (w) is 100, and the number of neighbors (K) is 50. The parameters of SVR and tree models are the default in MATLAB.

- **Simple linear cases (X_t is 1-dimensional)**

Table 7.3 lists the experiment results of different methods. In the conventional learning process, the data distribution is unchanged, so once a predictor has been trained on the training set, it is used to estimate all the upcoming data without adaptation. In the **LWR-NoAdapt** column, the results are computed in this non-adaptation way. The first column in **LWR-NoAdapt** presents the MAEs of models trained on the original space $(\mathcal{X}, \mathcal{Y})$, and the second column presents the MAEs of the reconstructed space $(\mathcal{X}, \mathcal{Y}, \mathcal{L})$. The **LWR-LDD⁺** column presents the results of adaptation by LDD⁺ in which the predictor is trained on the learning set updated by LDD⁺. There are four columns for LDD⁺: the linear column shows the results of training and updating a linear predictor on the space $(\mathcal{X}, \mathcal{Y})$, and the other three columns denote for the linear, SVR and tree models on the space $(\mathcal{X}, \mathcal{Y}, \mathcal{L})$, namely models of our DAR framework.

The accuracy results in Table 7.3 indicate that:1) LDD⁺-based adaptation can solve the drift problem in the data stream; 2) Reconstructed space helps to improve the drift adaptation process. The accuracy of **LWR-NoAdapt** before reconstruction is the same bad as that after reconstruction but the MAE of DAR-

Table 7.3: Comparisons between NoAdapt and adaptation by LDD⁺ approaches on different spaces, and DAR with different predictors.

MAE	LWR-NoAdapt		LWR-LDD ⁺ ($w=100$)			
	$(\mathcal{X}, \mathcal{Y})$	$(\mathcal{X}, \mathcal{Y}, \mathcal{L})$	$(\mathcal{X}, \mathcal{Y})$	$(\mathcal{X}, \mathcal{Y}, \mathcal{L})$		
Predictor	linear	linear	linear	DAR-linear	DAR-SVR	DAR-tree
NonDrift	0.8017	0.8029	0.8055	0.8079	0.8231	1.0638
SuddDrift	13.0991	13.2057	4.7669	1.7392	1.9061	1.8015
IncrDrift	9.542	9.315	4.5388	1.4387	1.2346	1.8967

linear is 1.7392 for SuddDrift and 1.4387 for IncrDrift which is much less than the MAE of linear. This verifies the discussion in Section 7.3.1 that reconstructed space improves drift adaptation for time-dependent data when the training set mixes samples from different patterns.

Next, we verify that reconstructed space improves drift adaptation because the error decreases faster. Figure 7.1 shows the complete estimation process of LWR-LDD⁺ and DAR-linear. In general, DAR-linear chases the various trends faster than LWR-LDD⁺. Subplot A is drawn before drift occurs, where the estimation of both methods is accurate. In contrast, when drift starts to appear in subplot B, DAR-linear gradually chases the new trend while LWR-LDD⁺ remains in the old pattern, as all the black dots are still near to the circles denoting the old pattern. In subplot C, DAR-linear has already adapted to the new concept after the 1101st point, but LWR-LDD⁺ has only just started to adapt to the new concept. Subplot D shows that LWR-LDD⁺ finally chases the new concept after the 1501st instance which has 400 points delay than DAR-linear.

The testing error of each instance from 950th to 1150th is given in subplot E. As the 950th-1000th instances are estimated by the predictor trained by instances before the 950th instance, there is no instance from the new pattern in the training set, and the testing error for the 999th and 1000th instance is

very high. After the arrival of the 1000th instance, the training set is updated and contains two instances from the new pattern, namely the 999th and 1000th instance. The training set is now a mixture of instances from the old and new patterns. If LWR-LDD⁺ is applied, the effectiveness of the adaptation is subtle and the testing error of the 1001st-1100th instances is still as high as in the non-adaptation predictor. However, when DAR-linear is applied, the testing error clearly decreases. After the 1100th instance has been obtained, the training set is updated again. The testing error of DAR-linear is now as low as it was before the drift occurred, which means that DAR-linear already adapted to the new pattern. In contrast, LWR-LDD has only just started to adapt.

So far, we have validate DAR on a simple linear regression for data stream with drift and temporal dependence. We have shown the estimated value of the label variable at each time point before and after drift occurs because the simple linear regression is suitable for a directly graphic presentation. In the next section, a general case of multi-dimensional non-linear data will be presented, where the average accuracy will be used for validation.

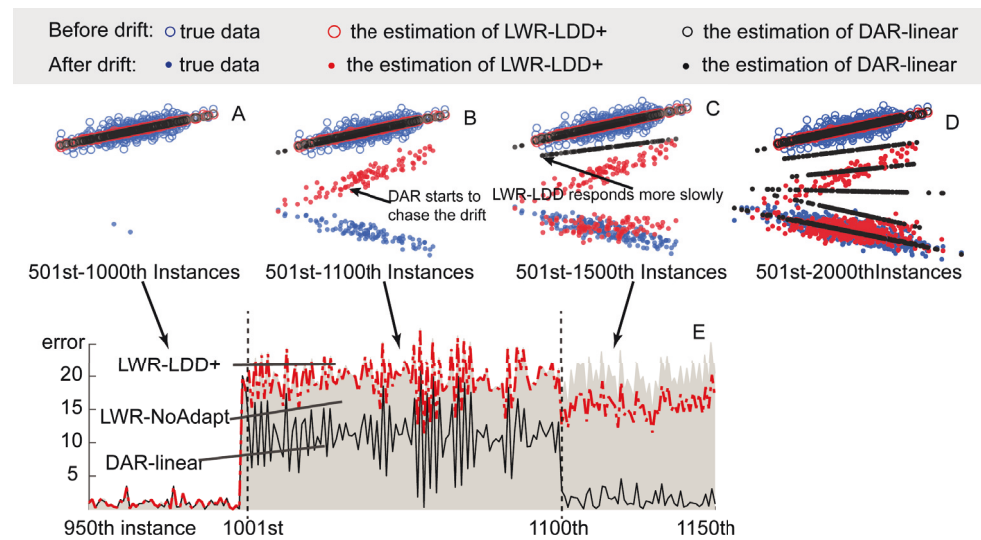


Figure 7.1: Error decreasing process. Circles represent the scatter plots of inputs and outputs before drift occurs, while dots show the results after drift has occurred. Colors denote the different estimation results: blue is for real values, red is for LWR-LDD⁺, and black is for DAR-linear. The testing errors of 950th-1150th instances are given in the subplot E, where the gray shadow represents LWR-NoAdapt, the red dotted line represents LWR-LDD⁺ and the black line represents DAR-linear.

- **Multi-dimensional non-linear cases**

In this section, instead of listing every predicted value and corresponding testing error, the average accuracy is used for validating DAR in six multi-dimensional non-linear cases. The results of predicting accuracy are listed in Table 7.4 (MAE) and Table 7.5 (MAPE). According to Table 7.4 and Table 7.5, we can obtain similar conclusion that DAR can handle data stream with drift and temporal dependency by the proposed LDD⁺-based adaptation on the reconstructed space. In addition, we found that replacing the linear predictor by other non-linear predictor such as SVR and tree in DAR may have better prediction results for the data with large weights on the non-linear terms. For example, in Table 7.4, the MAE of DAR-tree is much smaller than the MAE of DAR-linear on data Para-2 which only contains non-linear terms.

Table 7.4: Comparisons of MAE between NoAdapt and adaptation by LDD⁺ approaches on different spaces, and DAR with different predictors (multi-dimensional non-linear cases).

MAE	LWR-NoAdapt		LWR-LDD ⁺ ($w=100$)			
	$(\mathcal{X}, \mathcal{Y})$	$(\mathcal{X}, \mathcal{Y}, \mathcal{L})$	$(\mathcal{X}, \mathcal{Y})$	$(\mathcal{X}, \mathcal{Y}, \mathcal{L})$		
Predictor	linear	linear	linear	DAR-linear	DAR-SVR	DAR-tree
Para0	20.1489	18.4213	8.2653	5.3583	4.7753	3.5539
Para1	24.2418	24.606	9.0504	5.66	5.2406	4.7441
Para2	25.1195	24.6822	8.7878	4.3891	4.6255	4.3482
Para3	20.8337	19.5359	6.8099	2.8586	3.9276	2.8313
Para-1	11.9175	11.8405	5.5357	4.4129	3.9263	2.5419
Para-2	10.2588	10.5276	4.8849	4.1026	3.6865	2.2622

We also found that compared to adaptation on the original feature space, the improvement of DAR increases if the data generated with larger weights of the linear term. This validates our claim that ‘for an arbitrary data stream $DS_t^{(\mathbf{X}, \mathbf{y})}(s, \cdot)$ (Definition 7.7), a bigger s means a better estimation by DAR’. To

Table 7.5: Comparisons of MAPE between NoAdapt and adaptation by LDD⁺ approaches on different spaces, and DAR with different predictors (multi-dimensional non-linear cases).

MAPE	LWR-NoAdapt		LWR-LDD ⁺ ($w=100$)			
	$(\mathcal{X}, \mathcal{Y})$	$(\mathcal{X}, \mathcal{Y}, \mathcal{L})$	$(\mathcal{X}, \mathcal{Y})$	$(\mathcal{X}, \mathcal{Y}, \mathcal{L})$		
Predictor	linear	linear	linear	DAR-linear	DAR-SVR	DAR-tree
Para0	142.53%	129.45%	61.41%	42.68%	36.85%	25.98%
Para1	138.59%	140.90%	52.96%	35.55%	32.16%	29.03%
Para2	133.36%	130.66%	48.71%	25.41%	26.13%	25.30%
Para3	130.28%	119.73%	43.77%	24.73%	28.08%	19.01%
Para-1	175.37%	173.89%	88.64%	77.18%	62.29%	34.93%
Para-2	312.37%	323.73%	205.97%	215.42%	161.17%	42.14%

clearly present this, we compute the improvement percentage of DAR-linear compared to LWR-LDD⁺-linear, and draw the results in Figure 7.2. The x -axis starts from Para-2 and ends with Para3 which is arranged in an increased order of s . Each dot in Figure 7.2 is computed by subtracting the accuracy of DAR-linear from that of LWR-LDD⁺-linear and then divided by the accuracy of LWR-LDD⁺-linear. For example, the value of first blue dot is computed as $(4.8849 - 4.1026)/4.8849$, which is 16.01%.

It should be *noticed* both LWR-LDD⁺-linear and DAR-linear use linear predictors. Therefore, the difference between these two methods is not caused by the increased linear relationship between \mathbf{X} and y . The improvement clearly exists because there is a larger proportion of regular temporal dependency in the data stream. During the process of generating these six data, the temporal dependency on \mathbf{X} disappears when this term involves non-linear transforms. Therefore, using larger weights on the linear terms in (7.32) corresponds to the case of bigger s in Definition 7.7. These experimental results explain why we define the mixed time series for data stream.

Discussion: According to the synthetic data experiments, we conclude that:

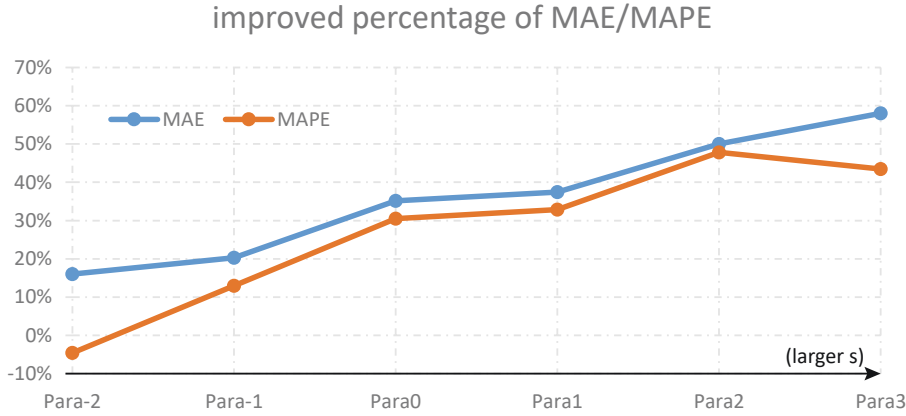


Figure 7.2: The improved percentage of MAE/MAPE of DAR-linear from LWR-LDD⁺(linear). Both methods have drift adaptation process. The difference is that DAR-linear is implemented on the reconstructed space.

1) feature space reconstruction will not obtain a worse estimation when no drift in the data; 2) the predictor trained on reconstructed feature space ties with the predictor on the original space if the training set only contains a single pattern; 3) when the data has drift and time-dependence, and the training set mixes data examples from different patterns, space reconstruction enables faster adaptation to the new pattern; 4) adaptation by LDD⁺ is effective for data streams with concept drift; 5) the effectiveness of reconstruction and adaptation by LDD⁺ is not affected by the type of predictors; 6) DAR performs better for the data streams with strong temporal dependency.

7.4.2 Experiments on real data streams

In this subsection, we test DAR framework on 7 real-world data streams. Seven techniques to tackle regression drift are introduced as benchmarks. One benchmark is LWR-NoAdapt which is the non adaptation version of DAR-linear, and

the rest six are: FIMT-DD Ikonomovska et al. (2011a), ORTO Ikonomovska et al. (2011b), AMR and its ensemble version, metaAMR Duarte et al. (2016), Perceptron Bifet et al. (2010c) and FUZZ-CARE Song et al. (2019a). The previous five benchmarks are implemented by MOA Bifet et al. (2010a)

Data description. There are seven data streams which have been introduced in Section 3.4.2.

Analysis of results on real-world data streams.

There are four preassigned parameters: length of training set, max lag order, length of windows, and the number of neighbors. All the experiments use a parameter combination as follows: the length of the training set (N) is 2000, the max lag order (P) is 3, the length of the windows (w) is 200, and the number of neighbors (K) is 50. The parameters of SVR and Tree models are the default in MATLAB.

The MAE and corresponding rank of DAR framework and other methods are listed in Table 7.6. The accuracy rank of each method shows that DAR-based methods perform better than other methods. The variables are very likely to exhibit the temporal dependency problem, and the overwhelming good performance of DAR-based methods demonstrates that our proposed DAR framework is an effective adaptation framework for data streams with drift and temporal dependency.

Table 7.6: MAE comparisons between different methods on real-world data streams. The DAR methods outperform the compared methods on data streams where concept drift occurs.

MAE/Rank	CCPP	Sensor3	Sensor8	Sensor20	Sensor46	SMEAR	Solar	Ave Rank
LWR-NoAdapt	3.6E+00 5	2.9E-01 10	6.5E-02 9	1.9E-01 7	2.2E-01 9	1.8E+01 6	1.9E+02 9	7.86
FIMT-DD	3.6E+00 4	7.1E-03 4	9.7E-03 7	8.0E-01 9	1.6E-01 6	2.3E+01 8	1.1E+02 7	6.43
ORTO	4.5E+02 10	6.6E-02 9	1.7E-01 10	9.6E-01 10	4.0E-01 10	3.4E+01 9	2.3E+02 10	9.71
AMR	3.4E+00 3	7.7E-03 5	6.6E-03 3	8.2E-03 5	2.0E-01 8	1.4E+01 5	9.5E+01 6	5.00
metaAMR	3.3E+00 1	1.6E-02 8	7.3E-03 4	1.1E-02 6	1.7E-01 7	2.0E+01 7	9.4E+01 5	5.43
Perceptron	3.7E+00 7	6.9E-03 3	6.0E-03 1	7.9E-01 8	1.6E-01 5	3.8E+01 10	1.3E+02 8	6.00
FUZZ-CARE	5.6E+00 9	1.6E-02 7	1.7E-02 8	7.9E-03 4	5.3E-02 4	1.0E+01 3	8.7E+01 4	5.57
DAR-linear	3.6E+00 6	5.1E-03 1	6.4E-03 2	3.9E-03 1	5.0E-03 1	9.9E+00 2	4.0E+01 2	2.14
DAR-SVR	3.3E+00 2	5.9E-03 2	7.8E-03 5	4.6E-03 2	5.8E-03 2	8.9E+00 1	3.1E+01 1	2.14
DAR-tree	4.7E+00 8	7.7E-03 6	9.4E-03 6	6.9E-03 3	7.7E-03 3	1.1E+01 4	4.6E+01 3	4.71

7.4.3 Statistical test on data Streams

In this section, we use Friedman test and its Post-hoc test after Conover to validate whether our proposed method is significantly better than those benchmarks. These two statistical test is conducted on the average MAE of different methods among all the data streams. The χ_R^2 statistic for Friedman test is computed as

$$(7.33) \quad \chi_R^2 = \frac{12}{nM(M+1)} \sum_{m=1}^M R_m^2 - 3n(M+1),$$

where M is the number of dependent treatment groups which is the number of methods in our case, n is the number of blocks, which is the number of data streams, R_i^2 is the squared rank sum of the i -th group.³ If the null hypothesis of the Friedman test is rejected, we use a post-hoc test after Conover for pairwise comparisons. The absolute difference between two group rank sums is significantly different if the following inequality is satisfied:

$$(7.34) \quad |R_i - R_j| > t_{1-\frac{\alpha}{2};(n-1)(M-1)} \times \sqrt{\frac{2M(1 - \frac{\chi_R^2}{n(M-1)})(\sum_{i=1}^n \sum_{m=1}^M R_{i,m}^2 - \frac{nM(M+1)^2}{4})}{(M-1)(n-1)}}.$$

When we conduct Friedman test and the post-hoc test, the methods concludes all three versions of DAR. The test results are shown in Table 7.7. While in Table 7.7, we only present the result of DAR-linear in post-hoc test part. $R_i - R_{DAR}$ is the difference between rank sums of other benchmarks and DAR-linear, and the p-value tests the significance of this difference.

The test results shows that the prediction accuracy of the various methods is different and DAR-linear is significantly better than other benchmarks.

³Please note that it has been wrongly written by R_i in Pohlert (2014) if you refer to this citation.

Table 7.7: Friedman test and its post-hoc test of all the methods over all seven data streams, where “Friedman Test” is the result for Friedman test and “Friedman - post-hoc test after Conover” is for the pairwise comparison. “*”, “***”, and “****” means this value is significant at the level of 0.05, 0.01 and 0.001 respectively. “df” denotes the freedom degree.

Friedman Test		χ_R^2	χ_{pvalue}^2	df			
		36.5376	3.18E-05****	7			
Friedman - post-hoc test after Conover							
DAR-linear	LWR-NoAdapt	FIMTDD	ORTO	AMR	metaAMR	Perceptron	FUZZ-CARE
$R_i - R_{DAR}$	40	30	53	20	23	27	24
diff _{pvalue}	0.000***	0.0013**	0.000***	0.020*	0.009**	0.003**	0.007**

7.5 Summary

In this chapter, we discuss drift adaptation for data streams that have both concept drift and temporal dependency problems. We conduct a theoretical study on the convergence of error when these two problems occur. Based on that, a drift-adapted regression (DAR) framework is proposed to predict the continuous target variable in non-stationary environments. The DAR framework is able to deal with drift and the temporal dependency problems simultaneously as it conducts the drift-adaptation process on a temporal reconstructed feature space. The drift-adaptation embedded in DAR framework is able to discard outdated samples in every batch or every instance by a developed statistic, LDD⁺.

The experimental evaluation on synthetic data verifies the theoretical aspect, and the experiments on real data streams illustrate several advantages of the proposed DAR framework: 1) effectiveness in handling abrupt and incremental drift in regression cases, 2) a fast response to drift, which makes it accurate in its prediction, and 3) good suitability for different predictors.

CONCLUSION AND FUTURE RESEARCH

This chapter concludes the entire thesis and provides some further research directions for this topic.

8.1 Conclusions

The development of the Internet of Things and Big data generates an increasing demand of real-time prediction in modern life. Conventional batch-based machine learning systems are built on a static assumption of independent and identically distributed (i.i.d) data and therefore are not suitable to make real-time prediction for data streams. Many subsequent studies have proved that the concept drift detection and adaptation techniques are effective approaches to solve the problem of distribution changes.

Recent research of concept drift poses several unsolved and challenging problems in this area, i.e., 1) how to effectively understand concept drift to help improve adaption; 2) how to effectively react to drift by adapting related knowledge; 3) the lack of theoretical frameworks for learning in the non-stationary environment; 4) how to solve the transient concept drift and limited data problem; and how to solve the concept drift problem if the data has other uncertainties such as 5) noise and 6) temporal dependencies.

To solve the above-mentioned challenges, this thesis proposes five concrete research questions and corresponding research objectives. The findings of this study are summarized as follows:

- *The proposed FUZZ-CARE method can identify concepts during the learning process. (to achieve objective 1)*

As it is unknown when concept drift occurs, the training set used to train the predictor probably contains several patterns, leading to the situation that the same input value maps to different outputs. To overcome this problem, this research introduces fuzzy techniques into the learning process and develops a drift adaptation method based on the fuzzy clustering process, named FUZZ-CARE. FUZZ-CARE is able to learn how many patterns exist in the observed data instances, i.e., the training set, and recognize the membership degree of each instance belonging to each pattern during the process of learning the parameters for the predictor.

- *The proposed SEGA method can sequentially select the most relevant data for training. (to achieve RQ2)*

Existing informed drift adaptation methods need to wait for an entire batch (time window) of data to detect drift and then update the predictor (if drift is detected), which causes adaptation delay. To overcome the adaptation delay and select the most relevant data to the new pattern, this research proposes a sequentially updated statistic (drift-gradient). Based on drift-gradient, this research develops a drift adaptation method (SEGA) to update the predictor with the most reliable data when every new instance arrives.

- *The proposed AFN method can generate samples of new concepts. (to achieve RQ3)*

Identifying a new concept in its early stages means there are few available instances of the new concept. It is difficult to learn a precise predictor for the new concept if there are not enough instances. To solve this problem, this research develops a drift adaptation method, named AFN to generate samples of new concept through previous data.

- *The proposed NoA method can be used for noisy data streams with concept drift. (to achieve RQ4)*

Drift adaptation for noisy data is a complex problem because concept drift and noise are not always independent in a data stream, and they have some overlaps sometimes. This research analyzes the data stream as a collection of time series and develops a drift adaptation method, called NoA, for data streams with signal noise.

- *The proposed DAR framework is able to handle data streams with concept drift and temporal dependency. (to achieve RQ5)*

The continuous variable in data streams is a time series process that is probably autocorrelated, which leads to the temporal dependency problem. This research discusses the concept drift problem in the data stream that have temporal dependency. A drift adaptation framework (DAR) is developed for data streams with concept drift and temporal dependency in regression cases.

8.2 Future Study

This thesis identifies the following directions as future work:

- *Adaptation efficiency.* This study assumes that we can store a fixed number of instances in the system. In most of the discussed cases, we store no more than 2000 instances in the buffer. When new instances arrive, the old instances may be replaced by new instances. However, the maximum storage is of 2000 instances. In some tasks, drift adaptation methods need to learn in a one-pass manner, without frequent or even no access to the previous data instances. Therefore, how to update predictors in a one-pass manner could be another meaningful direction of further work.
- *Concept drift adaptation for scarce data.* Scarce data may manifest in different ways such as insufficient data instances, imbalanced data, data without true labels, data streams with uneven time stamps. Oversampling is one of the promising techniques to solve this problem.
- *Concept drift adaptation in noisy data.* The noisy data, especially the feature noise is still a challenge in the research field of noise. Considering

the intrinsic relation between drift and noise, it is promising to use drift handling techniques to solve noise problems and vice versa.

- *Concept drift adaptation for data with temporal dependency.* Drift adaptation with temporal dependency has been a challenging problem since the idea of concept drift is proposed, but has rarely been solved. This problem is one of the most difficult challenges in this area because it needs solid theoretical guarantees other than the learning theory.
- *Concept drift detection and adaptation for multiple streams.* Real-world applications such as online decision making system often require a method for handling multiple streams at the same time. Therefore, a drift detection and adaptation framework for multiple data streams is needed.
- *A comprehensive concept drift adaptation framework.* This thesis has discussed the drift adaptation from five aspects including three root causes and two new scenarios. However, each aspect is discussed independently. The real applications normally need to consider two or more aspects simultaneously. For example, a data stream contains both signal noise and temporal dependency. Therefore, a comprehensive concept drift adaptation framework is needed to handle more realistic problems.



A.1 Derivation step from Symmetric Degree (SD) to Segmented Symmetric Degree(SSD)

According to (4.2), we have

$$\begin{aligned}
 \bar{d}_{P,Q}(k) &= \frac{1}{N_P} \sum_{u \in P_1} \left(\frac{|K_{u,P}(k)|}{N_P} - \frac{|K_{u,Q}(k)|}{N_Q} \right) + \frac{1}{N_P} \sum_{u \in P_2} \left(\frac{|K_{u,P}(k)|}{N_P} - \frac{|K_{u,Q}(k)|}{N_Q} \right) \\
 &+ \frac{1}{N_Q} \sum_{v \in Q} \left(\frac{|K_{v,Q}(k)|}{N_Q} - \frac{|K_{v,P}(k)|}{N_P} \right).
 \end{aligned}
 \tag{A.1}$$

As $|K_{u,P}(k)| + |K_{u,Q}(k)| = k$, the first two terms in $\bar{d}_{P,Q}(k)$ can be written by

$$(A.2) \quad \frac{1}{N_P} \sum_{u \in P_1} \left(\frac{|K_{u,P}(k)|}{N_P} - \frac{|K_{u,Q}(k)|}{N_Q} \right) = \frac{1}{N_P} \sum_{u \in P_1} \left(\frac{k - |K_{u,Q}(k)|}{N_P} - \frac{|K_{u,Q}(k)|}{N_Q} \right),$$

$$(A.3) \quad \frac{1}{N_P} \sum_{u \in P_2} \left(\frac{|K_{u,P}(k)|}{N_P} - \frac{|K_{u,Q}(k)|}{N_Q} \right) = \frac{1}{N_P} \sum_{u \in P_2} \left(\frac{k - |K_{u,Q}(k)|}{N_P} - \frac{|K_{u,Q}(k)|}{N_Q} \right).$$

The third term in (A.1) is

$$(A.4) \quad \frac{1}{N_Q} \sum_{v \in Q} \left(\frac{|K_{v,Q}(k)|}{N_Q} - \frac{|K_{v,P}(k)|}{N_P} \right) = 2 \times \frac{1}{2N_Q} \sum_{v \in Q} \frac{|K_{v,Q}(k)|}{N_Q} - \frac{1}{N_Q} \sum_{v \in Q} \frac{|K_{v,P}(k)|}{N_P}.$$

The term $|K_{v,P}(k)|$ represents the number of v 's k nearest neighbors that are from P . As $P = P_1 \cup P_2$ and $P_1 \cap P_2 = \emptyset$, $|K_{v,P}(k)| = |K_{v,P_1}(k)| + |K_{v,P_2}(k)|$, we have

$$(A.5) \quad \begin{aligned} \frac{1}{N_Q} \sum_{v \in Q} \left(\frac{|K_{v,Q}(k)|}{N_Q} - \frac{|K_{v,P}(k)|}{N_P} \right) &= \frac{1}{2N_Q} \sum_{v \in Q} \frac{|K_{v,Q}(k)|}{N_Q} + \frac{1}{2N_Q} \sum_{v \in Q} \frac{|K_{v,Q}(k)|}{N_Q} \\ &\quad - \frac{1}{N_Q} \sum_{v \in Q} \frac{|K_{v,P_1}(k)|}{N_P} - \frac{1}{N_Q} \sum_{v \in Q} \frac{|K_{v,P_2}(k)|}{N_P}. \end{aligned}$$

Substituting (A.2), (A.3) and (A.5) into (A.1), we have $\bar{d}_{P,Q}(k) = \text{ssd}_{P_1,Q}(k) + \text{ssd}_{P_2,Q}(k)$.

A.2 Estimation of model parameters for the mixed training set

The estimation of θ_1 by OLS is:

$$\begin{aligned} \hat{\theta}_1 &= \frac{\sum_{t=1}^n X_t y_t}{\sum_{t=1}^n X_t^2} = \frac{\sum_{t=1}^{n_1} X_t y_t + \sum_{t=1}^{n_2} X_t y_t}{\sum_{t=1}^n X_t^2} \\ &= \frac{n_1 \overline{X y_1} + n_2 \overline{X y_2}}{n \overline{X^2}} = \frac{(1-r) \overline{X y_1} + r \overline{X y_2}}{\overline{X^2}} \end{aligned} \tag{A.6}$$

$$\begin{aligned} \hat{\theta}_0 &= \frac{\sum_{t=1}^n y_t}{n} - \hat{\theta}_1 \overline{X} = \frac{\sum_{t=1}^{n_1} y_t + \sum_{t=n_1+1}^n y_t}{n} - \hat{\theta}_1 \overline{X} \\ &= \frac{n_1 \overline{y_1} + n_2 \overline{y_2}}{n} - \hat{\theta}_1 \overline{X} = (1-r) \overline{y_1} + r \overline{y_2} - \hat{\theta}_1 \overline{X} \end{aligned}$$

\overline{X} needs not to split because the distribution of X is unchanged.

A.3 The locally weighted regression

Details of the locally weighted regression algorithm can be found in Algorithm A.1

Algorithm A.1: The locally weighted regression

Input : $learnset$: data used to train the model, it contains $learninput$ and $learnoutput$;
 \mathbf{X}_q : the value of features for the query point;
 k : the number of neighbors;
 $model$: it can be linear, SVR or Tree model;
 θ : the other parameters needed in $model$.

Output: \hat{y}_q : the estimation of the label variable.

```
1  $Distance = knn(\mathbf{X}_q, learnsetinput)$   
2  $I = sort(Distance, 'ascend')$   
3  $index = I(1:k)$  % find the index of  $\mathbf{X}_q$ 's  $k$  nearest neighbors in  $learnset$   
    $(\mathbf{X}, y) = learnset(index)$   
4  $\hat{y}_q = model(\mathbf{X}, y, \theta)$   
5 return  $\hat{y}_q$ 
```

BIBLIOGRAPHY

- Abdallah, Z. S., Gaber, M. M., Srinivasan, B. & Krishnaswamy, S., 2018, 'Activity recognition with evolving data streams: A review', *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36.
- Aggarwal, C. C., Philip, S. Y., Han, J. & Wang, J., 2003, 'A framework for clustering evolving data streams', *Proceedings 2003 VLDB conference*, Elsevier, Berlin, Germany, Sep. 9-12, pp. 81–92.
- Aha, D. W., Kibler, D. & Albert, M. K., 1991, 'Instance-based learning algorithms', *Machine learning*, vol. 6, no. 1, pp. 37–66.
- Alippi, C., Boracchi, G. & Roveri, M., 2011, 'A just-in-time adaptive classification system based on the intersection of confidence intervals rule', *Neural Networks*, vol. 24, no. 8, pp. 791–800.
- Alippi, C., Boracchi, G. & Roveri, M., 2012, 'Just-in-time ensemble of classifiers', *The 2012 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Brisbane, Australia, Jun.10-15, pp. 1–8.
- Alippi, C., Boracchi, G. & Roveri, M., 2013, 'Just-in-time classifiers for recurrent concepts', *IEEE transactions on neural networks and learning systems*, vol. 24, no. 4, pp. 620–634.

- Alippi, C., Boracchi, G. & Roveri, M., 2017, 'Hierarchical change-detection tests', *IEEE transactions on neural networks and learning systems*, vol. 28, no. 2, pp. 246–258.
- Alippi, C. & Roveri, M., 2008a, 'Just-in-time adaptive classifiers,Äîpart i: Detecting nonstationary changes', *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1145–1153.
- Alippi, C. & Roveri, M., 2008b, 'Just-in-time adaptive classifiers,Äîpart ii: Designing the classifier', *IEEE Transactions on Neural Networks*, vol. 19, no. 12, pp. 2053–2064.
- Baena-Garcia, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R. & Morales-Bueno, R., 2006, 'Early drift detection method', *Fourth international workshop on knowledge discovery from data streams*, , vol. 6Berlin, Germany, Sep 18-22, pp. 77–86.
- Bakirov, R., Gabrys, B. & Fay, D., 2015, 'On sequences of different adaptive mechanisms in non-stationary regression problems', *28th International Joint Conference on Neural Networks*, IEEE, Killarney, Ireland, Jul.12-17, pp. 1–8.
- Bakker, J., Pechenizkiy, M., Žliobaitė, I., Ivannikov, A. & Kärkkäinen, T., 2009, 'Handling outliers and concept drift in online mass flow prediction in cfb boilers', *the 3rd International Workshop on Knowledge Discovery from Sensor Data*, ACM, Paris, France, Jun. 28, pp. 13–22.
- Bifet, A., de Francisci Morales, G., Read, J., Holmes, G. & Pfahringer, B., 2015, 'Efficient online evaluation of big data stream classifiers', *the 21th ACM SIGKDD*

- international conference on knowledge discovery and data mining*, ACM, Sydney, NSW, Australia, Aug.10-13, pp. 59–68.
- Bifet, A., Fan, W., He, C., Jianfeng, Q., Jianfeng, Z. & Holmes, G., 2017, ‘Extremely fast decision tree mining for evolving data streams’, *the 23rd SIGKDD Conference on Knowledge Discovery and Data Mining*, Halifax, NS, Canada, Aug. 13–17., pp. 1733–1742.
- Bifet, A., Gavalda, R. & Gavaldà, R., 2007, ‘Learning from time-changing data with adaptive windowing’, *the 7th SIAM International Conference on Data Mining*, Minneapolis Minnesota, USA, Apr. 26-28, pp. 443–448.
- Bifet, A., Holmes, G., Kirkby, R. & Pfahringer, B., 2010a, ‘MOA: Massive online analysis’, *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1601–1604.
- Bifet, A., Holmes, G. & Pfahringer, B., 2010b, ‘Leveraging bagging for evolving data streams’, *Joint European conference on machine learning and knowledge discovery in databases*, Springer, Barcelona, Spain, Sep. 19-23, pp. 135–150.
- Bifet, A., Holmes, G., Pfahringer, B. & Frank, E., 2010c, ‘Fast perceptron decision tree learning from evolving data streams’, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Hyderabad, India, Jun. 21-24, pp. 299–310.
- Boracchi, G., Carrera, D., Cervellera, C. & Maccio, D., 2018, ‘Quanttree: Histograms for change detection in multivariate data streams’, *the 35th International Conference on Machine Learning*, Stockholm, Sweden, Jul.10-15, pp. 638–647.

- Bose, R. P. J. C., Aalst, W. M. P. V. D., Žliobaitė, I. & Pechenizkiy, M., 2011, 'Handling concept drift in process mining', *23rd International Conference on Advanced Information Systems Engineering*, London, England, Jun. 20-24, pp. 391–405.
- Brzezinski, D. & Stefanowski, J., 2014, 'Reacting to different types of concept drift: The accuracy updated ensemble algorithm', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94.
- Bu, L., Alippi, C. & Zhao, D., 2018, 'A pdf-free change detection test based on density difference estimation', *IEEE transactions on neural networks and learning systems*, vol. 29, no. 2, pp. 324–334.
- Bu, L., Zhao, D. & Alippi, C., 2017, 'An incremental change detection test based on density difference estimation', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2714–2726.
- Carmona, J. & Gavaldà, R., 2012, 'Online techniques for dealing with concept drift in process mining', *the 11th International Symposium on Intelligent Data Analysis*, , vol. 7619 LNCSHelsinki, Finland, Oct. 25-27, pp. 90–102.
- Cavalcante, R. C., Minku, L. L. & Oliveira, A. L., 2016, 'FEDD: Feature extraction for explicit concept drift detection in time series', *2016 International Joint Conference on Neural Networks*, IEEE, Vancouver, British Columbia, Canada, Jul. 24-29, pp. 740–747.
- Chandola, V., Banerjee, A. & Kumar, V., 2009, 'Anomaly detection: A survey', *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58.

- Dasu, T. & Krishnan, S., 2006, 'An information-theoretic approach to detecting changes in multi-dimensional data streams', *the 38th Symposium on the Interface of Statistics, Computing Science, and Applications*, Pasadena California, USA, May 24-27, pp. 1–24.
- De Francisci Morales, G., Bifet, A., Khan, L., Gama, J. & Fan, W., 2016, 'IoT big data stream mining', *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, San Francisco, California, USA, Aug. 13-17, pp. 2119–2120.
- Ditzler, G. & Polikar, R., 2013, 'Incremental learning of concept drift from streaming imbalanced data', *IEEE transactions on knowledge and data engineering*, vol. 25, no. 10, pp. 2283–2301.
- Ditzler, G., Roveri, M., Alippi, C. & Polikar, R., 2015, 'Learning in nonstationary environments: A survey', *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25.
- dos Reis, D. M., Flach, P., Matwin, S. & Batista, G., 2016, 'Fast unsupervised online drift detection using incremental kolmogorov-smirnov test', *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, San Francisco California, USA, Aug. 13-17, pp. 1545–1554.
- Drosou, M., Jagadish, H. V., Pitoura, E. & Stoyanovich, J., 2017, 'Diversity in big data: A review', *Big Data*, vol. 5, no. 2, pp. 73–84.
- Duarte, J., Gama, J. & Bifet, A., 2016, 'Adaptive model rules from high-speed data streams', *ACM Transactions on Knowledge Discovery from Data*, vol. 10, no. 3, p. 30.

- Elgendy, N. & Elragal, A., 2014, 'Big data analytics: A literature review paper', *Proceedings of the 2014 Industrial Conference on Data Mining*, Springer, pp. 214–227.
- Elwell, R. & Polikar, R., 2011, 'Incremental learning of concept drift in nonstationary environments', *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531.
- Fan, W., 2004, 'Streamminer: A classifier ensemble-based engine to mine concept-drifting data streams', *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, Toronto, Canada, Aug 29 - Sep.3, pp. 1257–1260.
- Fan, W. & Bifet, A., 2013, 'Mining big data: Current status, and forecast to the future', *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 1–5.
- Frías-Blanco, I., del Campo-Ávila, J., Ramos-Jiménez, G., Carvalho, A. C., Ortiz-Díaz, A. & Morales-Bueno, R., 2016, 'Online adaptive decision trees based on concentration inequalities', *Knowledge-Based Systems*, vol. 104, pp. 179–194.
- Frías-Blanco, I., del Campo-Ávila, J., Ramos-Jiménez, G., Morales-Bueno, R., Ortiz-Díaz, A. & Caballero-Mota, Y., 2015, 'Online and non-parametric drift detection methods based on hoeffding's bounds', *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823.
- Friedman, J. H., 1991, 'Multivariate adaptive regression splines', *The annals of statistics*, pp. 1–67.

- Gaber, M. M., Zaslavsky, A. & Krishnaswamy, S., 2005, 'Mining data streams: a review', *ACM Sigmod Record*, vol. 34, no. 2, pp. 18–26.
- Gama, J., 2012, 'A survey on learning from data streams: current and future trends', *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 45–55.
- Gama, J., Medas, P., Castillo, G. & Rodrigues, P., 2004, 'Learning with drift detection', *Brazilian Symposium on Artificial Intelligence*, São Luís, Brazil, Sep. 29-Oct, pp. 286–295.
- Gama, J., Sebastião, R. & Rodrigues, P. P., 2013, 'On evaluating stream learning algorithms', *Machine learning*, vol. 90, no. 3, pp. 317–346.
- Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A., 2014, 'A survey on concept drift adaptation', *ACM computing surveys*, vol. 46, no. 4, p. 44.
- Gandomi, A. & Haider, M., 2015, 'Beyond the hype: Big data concepts, methods, and analytics', *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144.
- Gomes, H. M., Barddal, J. P., Enembreck, F. & Bifet, A., 2017a, 'A survey on ensemble learning for data stream classification', *ACM Computing Surveys*, vol. 50, no. 2, p. 23.
- Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G. & Abdessalem, T., 2017b, 'Adaptive random forests for evolving data stream classification', *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495.

- Gomes, H. M. & Enembreck, F., 2013, 'SAE: Social adaptive ensemble classifier for data streams', *2013 IEEE symposium on computational intelligence and data mining (CIDM)*, IEEE, Singapore, Apr.16-19, pp. 199–206.
- Gomes, H. M. & Enembreck, F., 2014, 'SAE2: advances on the social adaptive ensemble classifier for data streams', *Proceedings of the 29th annual ACM symposium on applied computing*, Gyeongju, Republic of Korea, Mar. 24-38, pp. 798–804.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y., 2014, 'Generative adversarial nets', *Advances in neural information processing systems*, Palais des Congrès de Montréal, Montréal Canada, Dec. 8-13, pp. 2672–2680.
- Gu, F., Zhang, G., Lu, J. & Lin, C. T., 2016, 'Concept drift detection based on equal density estimation', *the 29th International Joint Conference on Neural Networks, Vancouver, Canada, Jul. 24-29*, pp. 24–30.
- Guo, Z., Zhao, W., Lu, H. & Wang, J., 2012, 'Multi-step forecasting for wind speed using a modified emd-based artificial neural network model', *Renewable Energy*, vol. 37, no. 1, pp. 241–249.
- Gurjar, G. S. & Chhabria, S., 2015, 'A review on concept evolution technique on data stream', *2015 International Conference on Pervasive Computing (ICPC)*, IEEE, Pune, India, Jan. 8-10, pp. 1–3.
- Hamilton, J. D., 1994, *Time series analysis*, vol. 2, Princeton university press Princeton, NJ.

- Haque, A., Khan, L. & Baron, M., 2016a, 'SAND: Semi-supervised adaptive novel class detection and classification over data stream', *the 30th AAAI Conference on Artificial Intelligence*, Phoenix Arizona, USA, Feb. 12-17, pp. 1652–1658.
- Haque, A., Khan, L., Baron, M., Thuraisingham, B. & Aggarwal, C., 2016b, 'Efficient handling of concept drift and concept evolution over stream data', *IEEE 32nd International Conference on Data Engineering*, IEEE, Helsinki, Finland, May. 16-20, pp. 481–492.
- Harel, M., Mannor, S., El-Yaniv, R. & Crammer, K., 2014, 'Concept drift detection through resampling', *the 31st International Conference on Machine Learning, Beijing, China, Jun. 21-26*, pp. 1009–1017.
- Harries, M. B., Sammut, C. & Horn, K., 1998, 'Extracting hidden context', *Machine learning*, vol. 32, no. 2, pp. 101–126.
- Hernández, M. A. & Stolfo, S. J., 1998, 'Real-world data is dirty: Data cleansing and the merge/purge problem', *Data mining and knowledge discovery*, vol. 2, no. 1, pp. 9–37.
- Hou, B.-J., Zhang, L. & Zhou, Z.-H., 2017, 'Learning with feature evolvable streams', *Advances in Neural Information Processing Systems*, pp. 1417–1427.
- Huang, H. C., Chuang, Y. Y. & Chen, C. S., 2012, 'Multiple kernel fuzzy clustering', *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 1, pp. 120–134.
- Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., Yen, N.-C., Tung, C. C. & Liu, H. H., 1998, 'The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis',

Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences, vol. 454, no. 1971, pp. 903–995.

Hulten, G., Spencer, L. & Domingos, P., 2001, 'Mining time-changing data streams', *the 7th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, San Francisco, California, USA, Aug. 26-29, pp. 97–106.

Ikonomovska, E., Gama, J. & Džeroski, S., 2011a, 'Learning model trees from evolving data streams', *Data mining and knowledge discovery*, vol. 23, no. 1, pp. 128–168.

Ikonomovska, E., Gama, J., Ženko, B. & Džeroski, S., 2011b, 'Speeding-up hoeffding-based regression trees with options', *the 28th International Conference on Machine Learning*, Citeseer, Bellevue Washington, USA, Jun. 28- Jul. 2, pp. 537–544.

Ikonomovska, E., Jafarpour, S. & Dasdan, A., 2015, 'Real-time bid prediction using thompson sampling-based expert selection', *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Sydney, NSW, Australia Aug. 10-13, pp. 1869–1878.

Jang, J.-S., 1993, 'Anfis: adaptive-network-based fuzzy inference system', *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685.

Janssen, M., Van Der Voort, H. & Wahyudi, A., 2017, 'Factors influencing big data decision-making quality', *Journal of Business Research*, vol. 70, pp. 338–345.

- Jaworski, M., Duda, P. & Rutkowski, L., 2018, 'New splitting criteria for decision trees in stationary data streams', *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2516–2529.
- Kolter, J. Z. & Maloof, M. A., 2005, 'Using additive expert ensembles to cope with concept drift', *the 22nd international conference on Machine learning*, ACM, Bonn, Germany, Aug. 7-11, pp. 449–456.
- Kolter, J. Z. & Maloof, M. A., 2007, 'Dynamic weighted majority: An ensemble method for drifting concepts', *Journal of Machine Learning Research*, vol. 8, no. Dec, pp. 2755–2790.
- Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J. & Woźniak, M., 2017, 'Ensemble learning for data stream analysis: A survey', *Information Fusion*, vol. 37, pp. 132–156.
- Kreml, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M. et al., 2014, 'Open challenges for data stream mining research', *ACM SIGKDD explorations newsletter*, vol. 16, no. 1, pp. 1–10.
- Kubat, M., 1992, 'A machine learning-based approach to load balancing in computer networks', *Cybernetics and Systems*, vol. 23, no. 3-4, pp. 389–400.
- Labrinidis, A. & Jagadish, H. V., 2012, 'Challenges and opportunities with big data', *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 2032–2033.

- Li, C., Wei, F., Dong, W., Wang, X., Liu, Q. & Zhang, X., 2018, 'Dynamic structure embedded online multiple-output regression for streaming data', *IEEE Transactions on Pattern Analysis and Machine Intelligence*. .
- Littlestone, N., 1988, 'Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm', *Machine learning*, vol. 2, no. 4, pp. 285–318.
- Liu, A., 2019, 'Concept drift datasets', <https://github.com/Anjin-Liu/ConceptDriftDatasets/tree/master/Synthetic>.
- Liu, A., Lu, J., Liu, F. & Zhang, G., 2018a, 'Accumulating regional density dissimilarity for concept drift detection in data streams', *Pattern Recognition*, vol. 76, pp. 256–272.
- Liu, A., Song, Y., Zhang, G. & Lu, J., 2017a, 'Regional concept drift detection and density synchronized drift adaptation', *the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, Aug. 19-25, pp. 2280–2286.
- Liu, A., Zhang, G. & Lu, J., 2017b, 'Fuzzy time windowing for gradual concept drift adaptation', *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, Naples, Italy, Jul.9-12, pp. 1–6.
- Liu, D., Wu, Y. & Jiang, H., 2016, 'FP-ELM: An online sequential learning algorithm for dealing with concept drift', *Neurocomputing*, vol. 207, pp. 322–334.
- Liu, F., Lu, J. & Zhang, G., 2018b, 'Unsupervised heterogeneous domain adaptation via shared fuzzy equivalence relations', *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 6, pp. 3555–3568.

- Losing, V., Hammer, B. & Wersing, H., 2016, 'Knn classifier with self adjusting memory for heterogeneous concept drift', *IEEE 16th International Conference on Data Mining (ICDM)*, IEEE, Barcelona, Spain, Dec 12-15, pp. 291–300.
- Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. & Zhang, G., 2018, 'Learning under concept drift: A review', *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363.
- Lu, J., Liu, A., Song, Y. & Zhang, G., 2019, 'Data-driven decision support under concept drift in streamed big data', *Complex & Intelligent Systems*, pp. 1–7.
- Lu, N., 2012, *Handling concept drift in case-based reasoning systems*, Ph.D. thesis, Univerisity of Technology Sydney.
- Lu, N., Lu, J., Zhang, G. & De Mantaras, R. L., 2016, 'A concept drift-tolerant case-base editing technique', *Artificial Intelligence*, vol. 230, pp. 108–133.
- Lu, N., Zhang, G. & Lu, J., 2014, 'Concept drift detection via competence models', *Artificial Intelligence*, vol. 209, pp. 11–28.
- Lu, Y., Cheung, Y.-m. & Tang, Y. Y., 2017, 'Dynamic weighted majority for incremental learning of imbalanced data streams with concept drift.', *the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, Aug.19-25, pp. 2393–2399.
- Lughofer, E., Weigl, E., Heidl, W., Eitzinger, C. & Radauer, T., 2016, 'Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled [sic] instances', *Information Sciences*, vol. 355-356, pp. 127–151.

BIBLIOGRAPHY

- Maass, W., Parsons, J., Puro, S., Rosales, A., Storey, V. C. & Woo, C. C., 2017, 'Big data and theory', *Encyclopedia of Big Data*, pp. 1–5.
- Maloof, M. A. & Michalski, R. S., 2004, 'Incremental learning with partial instance memory', *Artificial intelligence*, vol. 154, no. 1-2, pp. 95–126.
- Mendes-Moreira, J., Soares, C., Jorge, A. M. & Sousa, J. F. D., 2012, 'Ensemble approaches for regression: A survey', *ACM Computing Surveys*, vol. 45, no. 1, p. 10.
- Minku, L. L., White, A. P. & Yao, X., 2010, 'The impact of diversity on online ensemble learning in the presence of concept drift', *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742.
- Minku, L. L. & Yao, X., 2012, 'DDD: A new ensemble approach for dealing with concept drift', *IEEE transactions on knowledge and data engineering*, vol. 24, no. 4, pp. 619–633.
- Moreira-Matias, L., Gama, J. & Mendes-Moreira, J., 2016, 'Concept neurons—handling drift issues for real-time industrial data mining', *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, Riva del Garda, Italy, Sep 19-23, pp. 96–111.
- OpenML, n.d., 'OpenML datasets', <https://www.openml.org/home>.
- Oza, N. C., 2005, 'Online bagging and boosting', *2005 IEEE international conference on systems, man and cybernetics*, , vol. 3Ieee, Waikoloa, Hawaii, USA, Oct. 10-12, pp. 2340–2345.

- Pace, R. K. & Barry, R., 1997, 'Sparse spatial autoregressions', *Statistics & Probability Letters*, vol. 33, no. 3, pp. 291–297.
- Pal, N. R. & Sarkar, K., 2014, 'What and when can we gain from the kernel versions of c-means algorithm?', *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 2, pp. 363–379.
- Parker, B. S. & Khan, L., 2015, 'Detecting and tracking concept class drift and emergence in non-stationary fast data streams', *29th AAAI Conference on Artificial Intelligence*, Austin Texas, USA, Jan 25-30, pp. 2908–2913.
- Pohlert, T., 2014, 'The pairwise multiple comparison of mean ranks package (PMCMR)', *R package*, pp. 2004–2006.
- Pratama, M., Lu, J., Lughofer, E., Zhang, G. & Anavatti, S., 2016, 'Scaffolding type-2 classifier for incremental learning under concept drifts', *Neurocomputing*, vol. 191, pp. 304–329.
- Pratama, M., Lu, J., Lughofer, E., Zhang, G. & Er, M. J., 2017, 'An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks', *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 5, pp. 1175–1192.
- Puthal, D., Nepal, S., Ranjan, R. & Chen, J., 2017, 'Dlsef: A dynamic key-length-based efficient real-time security verification model for big data stream', *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 2, p. 51.
- Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M. & Herrera, F., 2017, 'A survey on data preprocessing for data stream mining: Current status and future directions', *Neurocomputing*, vol. 239, pp. 39–57.

BIBLIOGRAPHY

- Rutkowski, L., Jaworski, M., Pietruczuk, L. & Duda, P., 2015, 'A new method for data stream mining based on the misclassification error', *IEEE transactions on neural networks and learning systems*, vol. 26, no. 5, pp. 1048–1059.
- Sakamoto, Y., Fukui, K.-I., Gama, J., Nicklas, D., Moriyama, K. & Numao, M., 2015, 'Concept drift detection with clustering via statistical change detection methods', *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE)*, IEEE, pp. 37–42.
- Sakamoto, Y., Ishiguro, M. & Kitagawa, G., 1986, *Akaike information criterion statistics*, KTK Scientific, Tokyo, with D. Reidel, Dordrecht, Holland.
- Schlimmer, J. C. & Granger, R. H., 1986a, 'Beyond incremental processing: Tracking concept drift.', *the 5th AAAI Conference on Artificial Intelligence*, Philadelphia Pennsylvania, USA, Aug. 11-15, pp. 502–507.
- Schlimmer, J. C. & Granger, R. H., 1986b, 'Incremental learning from noisy data', *Machine learning*, vol. 1, no. 3, pp. 317–354.
- scikit-learn package, P., 2007, 'sklearn.datasets.make_friedman1', <https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_friedman1.html#sklearn.datasets.make_friedman1>.
- Shaker, A. & Hüllermeier, E., 2012, 'Iblstreams: a system for instance-based classification and regression on data streams', *Evolving Systems*, vol. 3, no. 4, pp. 235–249.
- Shao, J., Ahmadi, Z. & Kramer, S., 2014, 'Prototype-based learning on concept-drifting data streams', *the 20th ACM SIGKDD international conference on*

- Knowledge discovery and data mining*, ACM, New York, USA, Aug.24-27, pp. 412–421.
- Soares, S. G. & Araújo, R., 2015, ‘An on-line weighted ensemble of regressor models to handle concept drifts’, *Engineering Applications of Artificial Intelligence*, vol. 37, pp. 392–406.
- Song, Y., 2019, ‘FUZZ-CARE’, <https://github.com/songyiliao/FUZZ-CARE>.
- Song, Y., Lu, J., Lu, H. & Zhang, G., 2019a, ‘Fuzzy clustering-based adaptive regression for drifting data streams’, *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 3, pp. 544–557.
- Song, Y., Qin, S., Qu, J. & Liu, F., 2015, ‘The forecasting research of early warning systems for atmospheric pollutants: A case in yangtze river delta region’, *Atmospheric Environment*, vol. 118, pp. 58–69.
- Song, Y., Zhang, G., Lu, H. & Lu, J., 2018, ‘A self-adaptive fuzzy network for prediction in non-stationary environments’, *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, Rio de Janeiro, Brazil, Jul.8-13, pp. 1–8.
- Song, Y., Zhang, G., Lu, H. & Lu, J., 2019b, ‘A noise-tolerant fuzzy c-means based drift adaptation method for data stream regression’, *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, New Orleans, LA, USA, Jun. 23-26, pp. 1–6.
- Song, Y., Zhang, G., Lu, J. & Lu, H., 2017, ‘A fuzzy kernel c-means clustering

- model for handling concept drift in regression', *IEEE International Conference on Fuzzy Systems*, IEEE, Naples, Italy, Jul.9-12, pp. 1–6.
- Sousa, M. R., Gama, J. & Brandão, E., 2016, 'A new dynamic modeling framework for credit risk assessment', *Expert Systems with Applications*, vol. 45, pp. 341–351.
- Stanley, K. O., 2003, 'Learning concept drift with a committee of decision trees', *Informe técnico: UT-AI-TR-03-302*, Department of Computer Sciences, University of Texas at Austin, USA.
- Stoica, I., Song, D., Popa, R. A., Patterson, D., Mahoney, M. W., Katz, R., Joseph, A. D., Jordan, M., Hellerstein, J. M., Gonzalez, J. E. et al., 2017, 'A berkeley view of systems challenges for ai', *arXiv preprint arXiv:1712.05855*.
- Street, W. N. & Kim, Y., 2001, 'A streaming ensemble algorithm (SEA) for large-scale classification', *7th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, San Francisco CA, USA, Aug. 26-29, pp. 377–382.
- Su, B., Shen, Y.-D. & Xu, W., 2008, 'Modeling concept drift from the perspective of classifiers', *2008 IEEE Conference on Cybernetics and Intelligent Systems*, IEEE, Chengdu, Chinam, Sep. 21-24, pp. 1055–1060.
- Sun, Y., Tang, K., Zhu, Z. & Yao, X., 2018, 'Concept drift adaptation by exploiting historical knowledge', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 4822–4832.

- Tennant, M., Stahl, F., Rana, O. & Gomes, J. B., 2017, 'Scalable real-time classification of data streams with concept drift', *Future Generation Computer Systems*, vol. 75, pp. 187–199.
- Tsymbal, A., 2004, 'The problem of concept drift: definitions and related work', *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58.
- Wadewale, K. & Desai, S., 2015, 'Survey on method of drift detection and classification for time varying data set', *Int. Res. J. Eng. Technol.*, vol. 2, no. 9, pp. 709–713.
- Wang, H., Fan, W., Yu, P. S. & Han, J., 2003, 'Mining concept-drifting data streams using ensemble classifiers', *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, AcM, Washington, DC, USA, Aug.24-27, pp. 226–235.
- Wang, L.-Y., Park, C., Yeon, K. & Choi, H., 2017, 'Tracking concept drift using a constrained penalized regression combiner', *Computational Statistics & Data Analysis*, vol. 108, pp. 52–69.
- Wang, S., Minku, L. L. & Yao, X., 2015, 'Resampling-based ensemble methods for online class imbalance learning', *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1356–1368.
- Wang, S., Minku, L. L. & Yao, X., 2018, 'A systematic study of online class imbalance learning with concept drift', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 4802–4821.

- Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L. & Petitjean, F., 2016, 'Characterizing concept drift', *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994.
- Widmer, G. & Kubat, M., 1996, 'Learning in the presence of concept drift and hidden contexts', *Machine learning*, vol. 23, no. 1, pp. 69–101.
- Xie, H.-B., Sivakumar, B., Boonstra, T. W. & Mengersen, K., 2018, 'Fuzzy entropy and its application for enhanced subspace filtering', *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 4, pp. 1970–1982.
- Xiong, H., Pandey, G., Steinbach, M. & Kumar, V., 2006, 'Enhancing data analysis with noise removal', *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 304–319.
- Xu, S. & Wang, J., 2017, 'Dynamic extreme learning machine for data stream classification', *Neurocomputing*, vol. 238, pp. 433–449.
- Yang, H. & Fong, S., 2015, 'Countering the concept-drift problems in big data by an incrementally optimized stream mining model', *Journal of Systems and Software*, vol. 102, pp. 158–166.
- Yang, X., Zhang, G., Lu, J. & Ma, J., 2010, 'A kernel fuzzy c-means clustering-based fuzzy support vector machine algorithm for classification problems with outliers or noises', *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 1, pp. 105–115.
- Yang, X., Zhang, G., Lu, J. & Ma, J., 2011, 'A kernel fuzzy c-means clustering-based fuzzy support vector machine algorithm for classification problems with

- outliers or noises’, *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 1, pp. 105–115.
- Yeon, K., Song, M. S., Kim, Y., Choi, H. & Park, C., 2010, ‘Model Averaging via Penalized Regression for Tracking Concept Drift’, *Journal of Computational and Graphical Statistics*, vol. 19, no. 2, pp. 457–473.
- Zambon, D., Alippi, C. & Livi, L., 2018, ‘Concept drift and anomaly detection in graph streams’, *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5592–5605.
- Zeira, G., Maimon, O., Last, M. & Rokach, L., 2004, ‘Change detection in classification models induced from time series data’, *Data mining in time series databases*, World Scientific, pp. 101–125.
- Zhang, Z., Song, Y., Liu, F. & Liu, J., 2016, ‘Daily average wind power interval forecasts based on an optimal adaptive-network-based fuzzy inference system and singular spectrum analysis’, *Sustainability*, vol. 8, no. 2, p. 125.
- Zhao, P., Cai, L.-W. & Zhou, Z.-H., 2018, ‘Handling concept drift via model reuse’, *Machine Learning*, pp. 1–36.
- Zheng, Z., Zheng, L. & Yang, Y., 2017, ‘Unlabeled samples generated by gan improve the person re-identification baseline in vitro’, *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, Oct 22-29, pp. 3754–3762.
- Zhu, X., 2010, ‘Stream data mining repository’, <http://www.cse.fau.edu/~xqzhu/stream.html>.

- Zhu, Y., Ting, K. M. & Zhou, Z.-H., 2018, 'Multi-label learning with emerging new labels', *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1901–1914.
- Žliobaitė, I., 2010a, 'Learning under concept drift: an overview', *arXiv preprint arXiv:1010.4784*.
- Žliobaitė, I., 2010b, 'Learning under concept drift: an overview', *arXiv preprint arXiv:1010.4784*.
- Žliobaitė, I., Bifet, A., Pfahringer, B. & Holmes, G., 2014a, 'Active learning with drifting streaming data', *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39.
- Žliobaitė, I., Bifet, A., Read, J., Pfahringer, B. & Holmes, G., 2015, 'Evaluation methods and decision theory for classification of streaming data with temporal dependence', *Machine Learning*, vol. 98, no. 3, pp. 455–482.
- Žliobaitė, I., Hollmén, J. & Junninen, H., 2014b, 'Regression models tolerant to massively missing data: a case study in solar-radiation nowcasting', *Atmospheric Measurement Techniques*, vol. 7, no. 12, pp. 4387–4399.
- Žliobaitė, I., Pechenizkiy, M. & Gama, J., 2016, 'An overview of concept drift applications', *Big data analysis: new algorithms for a new society*, Springer, Cham, pp. 91–114.
- Zuo, H., Zhang, G., Pedrycz, W., Behbood, V. & Lu, J., 2016, 'Fuzzy regression transfer learning in takagi–sugeno fuzzy models', *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 6, pp. 1795–1807.

Zuo, H., Zhang, G., Pedrycz, W., Behbood, V. & Lu, J., 2018, 'Granular fuzzy regression domain adaptation in takagi–sugeno fuzzy models', *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 2, pp. 847–858.