Northumbria University
NEWCASTLE

CONCEPT DRIFT DETECTION USING MACHINE LEARNING IN DATA STREAM

PINGFAN WANG

PhD

2023

CONCEPT DRIFT DETECTION USING MACHINE LEARNING IN DATA STREAM

PINGFAN WANG

A thesis submitted in partial fulfilment of the requirements of the University of Northumbria at Newcastle for the degree of Doctor of Philosophy

Faculty of Faculty of Engineering and Environment

December 2023

Abstract

Machine learning applications in streaming data often grapple with dynamic changes in data distribution, particularly concept drift, where shifts in classification boundaries undermine the model's performance. This challenge is further complicated by the inherent complexity of data streams, the underutilization of deep neural networks in addressing the issue, and a lack of comprehensive understanding of the concept drift.

Data streams are non-stationary and complex by nature, which poses significant challenges to concept drift detection. Deep neural networks, despite their immense predictive power, are rarely employed in this context due to their high computational costs. Current detection methods typically concentrate on pinpointing when a concept drift occurs, neglecting to explore the detailed information about the concept drift. This dearth of information, such as the concept drift's onset, duration, severity, or endpoint, could be invaluable for a more nuanced understanding of the phenomenon.

To mitigate these issues, this thesis introduces several innovative methods:

Drift Detection Method with False Positive rate for Multi-label classification (DDM-FP-M): This novel approach extends the existing Drift Detection Method (DDM) to multi-label classification data streams. It incorporates a unique mechanism to adjust for false positives, enhancing the adaptability and accuracy of drift detection in complex data stream scenarios.

Noise Tolerant Drift Detection Method (NTDDM): NTDDM introduces a two-step process to discern true drifts from noise-induced false positives. It refines drift detection by filtering out misleading signals through subsampling and statistical detection methods, improving the reliability of drift detection in noisy data environments. The efficacy of this method is further validated through three newly proposed performance metrics specifically designed for concept drift detection.

Incremental Weighted Performance Drift Detection Method (IWPDDM): This method employs prediction confidence derived from the incremental learning of ensemble models to detect concept drift. It represents a shift in focus towards the model's own response to concept drift, rather than solely relying on the model's output. It creates an indicator using weighted prediction confidence from these models, ensuring stable and accurate drift detection, a significant improvement over traditional methods.

Model-centric Transfer Learning (MCDD) Framework: Recognizing the limited use of deep neural networks in concept drift detection due to computational constraints, this thesis proposes the MCDD framework. This approach relies solely on the model's intrinsic changes to detect concept drift, making it a model-centric method. Our experiments demonstrate that mere changes in the model itself can accurately reflect concept drift. This framework strategically utilizes transfer learning to freeze parts of the network, significantly reducing computational needs while enhancing drift detection performance.

Quadruple-based Approach for Understanding Concept Drift in Data Streams (QuadCDD): The QuadCDD framework aims to provide a holistic understanding of concept drift. Most existing methods focus only on detecting the start point of concept drift, yet there is much information about concept drift that remains unexplored, such as its endpoint, severity, and type. This lack of information greatly reduces the specificity of adaptation strategies. The QuadCDD framework goes beyond merely identifying the onset of drift, equipping models with comprehensive information for more effective adjustments and a deeper understanding of the drift dynamics.

In conclusion, this thesis addresses a critical issue in machine learning on data streams. It provides practical and innovative concept drift detection algorithms, contributing significantly to both scientific research and practical applications in the field.

Contents

List of Figures

List of Tables

Acronyms

ADWIN	ADaptive WINdowing 28, 34
AI	Artificial Intelligence 27
AUC	Area Under the Receiver Operating Characteristic Curve 71
CDANs	Conditional Domain Adversarial Networks 34
CI-CUSUM	Computational Intelligence-based CUSUM test 32
CM	Competence Model-based Drift Detection 30
DAOD	Distribution Alignment with Open Difference 34
DDM	Drift Detection Method 8, 11, 20, 28, 36
DDM-FP-M	Drift Detection method with False Positive rate for multi label classification 36
DEML	Dynamic Extreme Learning Machine 28
e-Detector	Ensemble of Detectors 32
EDDM	Early Drift Detection Method 20, 28
EDDR	Effective Detected Drift Rate 58, 60
EDE	Equal Density Estimation 30
EDR	Effective detected drift rate 58
EFS	Evolving Fuzzy System 30
eGAUSS+	Evolving Gaussian Clustering 31
ELM	Extreme Learning Machine 28
eT2RFNN	Evolving Type-2 Recurrent Fuzzy Neural Network 31

FEDP First Effective Detected Point 58, 62

FP-Growth Frequent Pattern Growth 19

FPR False Positive rate 36

HBOS Histogram-based Outlier Score 19

HCDTs Hierarchical Change-Detection Tests 32

HDDM Heoffding's inequality based Drift Detection Method 28

HHT-AG Hierarchical Hypothesis Testing with Attribute-wise "Goodness-of-fit" 33

HHT-CU Hierarchical Hypothesis Testing with Classification Uncertainty 33

HLFR Hierarchical Linear Four Rate 32

IoT Internet of Things 15, 35, 36

ITA Information-Theoretic Approach 30

IV-Jac Information Value and Jaccard similarity 32

IWPDDM Incremental weighted performance drift detection method 71

JIT Just-In-Time Adaptive Classifiers 32

KS Kolmogorov–Smirnov Test 29

KS test Kolmogorov–Smirnov two-sample test 54

LDD-DSDA Local Drift Degree-based Density Synchronized Drift Adaptation 30

LDP Last Detected Point 58

LFR Linear Four Rate drift detection 32, 33

LLDD Learning with Local Drift Detection 28

LOF Local Outlier Factor 19

LSDD-CDT Least Squares Density Difference-based Change Detection Test 30

LSDD-INC Incremental version of Least Squares Density Difference-based Change Detection Test 30

MCDD Model-centric framework for concept drift detection 79, 81

Meta-ADD	Active Drift Detection based on Meta learning 34
NTDDM	Noise Tolerant Drift Detection Method 45, 50
OB-EFS	Online Bagged EFS 30
OOD	Out-of-distribution 34
PAC	Probably Approximately Correct 34
PCA-CD	Principal Component Analysis-based Change Detection Framework 30
PH	Page-Hinkley 28
PS	Prediction stability 73
QuadCDD	Quadruple-based Approach for Understanding Concept Drift 109
RD	Relativized Discrepancy 29
ROC	Receiver Operating Characteristic 106
SAND	Semi-Supervised Adaptive Novel Class Detection 29
SCD	Statistical Change Detection 30
SEOA	Selective ensemble-based online adaptive deep neural network 33
SGD	Stochastic gradient descent 95
TSMSD-EWMA	Two-Stage Multivariate Shift-Detection based on EWMA 33
UOSDA	Unsupervised Open Set Domain Adaptation 34

Acknowledgements

My PhD journey has been quite unusual. In the second term of my PhD, around the Spring Festival of 2020, the COVID-19 pandemic swept the world. For the following one to two years, I was mostly at home, hardly even remembering where my office desk was. I feel extremely fortunate to have successfully completed my PhD thesis and to be now sitting in the office, typing these words.

I would like to express my deepest gratitude to my two advisors, Professor Nanlin Jin and Professor Wai Lok Woo. Professor Nanlin Jin was the one who initially recruited me as a PhD student and provided substantial help during the early stages of my research, including various academic guidance and invaluable suggestions. This has greatly facilitated my research journey, enabling me to fully delve into the topics I'm passionate about. After leaving Northumbria University about two years later, Professor Nanlin Jin continued to serve as my external supervisor and provided precious advice on my research. During the latter part of my PhD research, Professor Wai Lok Woo acted as the main advisor and offered substantial assistance. His expertise in the field of deep learning facilitated professional guidance in an unfamiliar area. I am also very thankful to Professor Wai Lok Woo for letting me deeply participate in several collaborative research projects, which not only enhanced my academic resume but also alleviated my financial constraints during the latter part of my PhD journey. I can say that without these two professional and incredibly supportive advisors, it would have been very difficult for me to graduate at this point in time.

Secondly, I am very grateful to my colleagues at the Department of Computer and Information Sciences at Northumbria University. I have gained a lot of research experience and advice from them, for which I am very grateful. I would like to express my special thanks to Dr. Jeremy Ellman, who served as a member of my annual progress review panel and collaborated as a course teacher for three years. Even after his retirement, he was still willing to review my PhD thesis, for which I am extremely grateful. During my PhD research, I received help from colleagues in the same field, including Professor Hang Yu from Shanghai University, Dr. Yue Zhao from the University of Southern California, and Dr. Yiliao Song from the University of Technology Sydney. I also want to express my gratitude to Dr. Haixun Xie, Dr. Shan Shan, Mr. Jialou Wang, Ms. Manli Zhu, and Mr. Anthony Ashwin Peter Chazhoor for our daily discussions. I am also very grateful for the enthusiastic help from Dr. Kay Rogage and Dr. Baqar Rizvi in my teaching

work at Northumbria University. I would like to thank all friends in ”Ke Ji Yu Tang” for their advice and experience sharing in the same field, which greatly alleviated my mental stress during my PhD journey.

In the past few months, in my teaching work at Coventry University, I am very grateful to Dr. Shan Shan for introducing me to this job opportunity, and I also very much appreciate Professor Muhammad Kamal for his support of my work. I also want to express my special thanks to Dr. Selman Karagoz and Dr. Umar Farooq for their advice and guidance in teaching.

Finally, I would like to thank my family who have always supported me. It is their decades of selfless love and support that have allowed me to remain in the campus engaged in pure study and research. Of course, I must thank my wife, who has witnessed my entire PhD journey, my first academic conference, my first rejected paper, my first accepted paper, and all kinds of ”firsts”. You have changed my attitude towards life and study, making me better. In the future, I hope my first time wearing a doctoral gown, the first time holding a doctoral degree, and many more firsts will have your participation!

Declaration

I declare that the work contained in this thesis has not been submitted for any other award and that it is all my own work. I also confirm that this work fully acknowledges opinions, ideas and contributions from the work of others.

Any ethical clearance for the research presented in this thesis has been approved. Approval has been sought and granted by the Faculty Ethics Committee on 04/02/2020.

I declare that the Word Count of this thesis is 35132 words.

Name: Pingfan Wang

Date: 5 December 2023

List of publications

P. Wang, H. Yu, N. Jin, and W. Woo, QuadCDD: A Quadruple-based Approach for Understanding Concept Drift in Data Streams, *Expert Systems with Applications*, Volume 238, Part E, 2024, 122114, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.122114.

P. Wang, N. Jin, D. Davies, and W. L. Woo, "Model-centric transfer learning framework for concept drift detection," *Knowledge-Based Systems*, vol. 275, pp. 110705, 2023. doi: 10.1016/j.knosys.2023.110705.

P. Wang, N. Jin, W. L. Woo, J. R. Woodward, and D. Davies, "Noise tolerant drift detection method for data stream mining," *Information Sciences*, vol. 609, pp. 1318-1333, 2022. doi: 10.1016/j.ins.2022.07.065.

P. Wang, W. Woo, N. Jin, and D. Davies, "Concept Drift Detection by Tracking Weighted Prediction Confidence of Incremental Learning," in *Proceedings of the 2022 4th International Conference on Image, Video and Signal Processing (IVSP '22)*, New York, NY, USA, 2022, pp. 218-223, doi: 10.1145/3531232.3531264.

P. Wang, N. Jin and G. Fehringer, "Concept drift detection with False Positive rate for multi-label classification in IoT data stream," *2020 International Conference on UK-China Emerging Technologies (UCET)*, Glasgow, UK, 2020, pp. 1-4, doi: 10.1109/UCET51115.2020.9205421.

P. Wang, S. McGrath, and N. Jin, "A hybrid decision-making system using image analysis by deep learning and IoT sensor data to detect human falls," in Proc. *1st International 'Alan Turing' Conference on Decision Support and Recommender Systems (DSRS-Turing'19)*, London, United Kingdom, 21-22nd November 2019, ISBN: 978-1-5262-0820-0.

Chapter 1

Introduction

1.1 Background

As our society becomes increasingly digital, governments and corporations are grappling with the challenge of handling massive volumes of data that continuously stream from various sources. The insights gleaned from such data are often instrumental in making strategic decisions and predicting future trends. However, the dynamic nature of this data, due to the constant emergence of new markets, customer behaviors, and products, presents a significant challenge known as concept drift.

Concept drift refers to the unpredictable changes in the statistical properties of the target variable that a predictive model is designed to estimate (Iwashita and Papa, 2019). When concept drift occurs, patterns that the model learned from past data may no longer apply to new data, leading to a decline in predictive accuracy and the potential for misinformed decisions, as shown in Figure 1.1. Concept drift has been implicated as a primary factor causing decreased effectiveness in data-driven information systems, such as decision support systems and early warning systems (Xu and Wilson, 2021). As a result, ensuring the reliability of data-driven predictions in an ever-changing big data landscape has become a paramount concern.

The omnipresence of concept drift in real-world scenarios is exemplified by changing behavior patterns in mobile phone usage. Over the past two decades, the primary use of mobile phones has shifted from simple voice communication to a variety of applications including internet browsing,

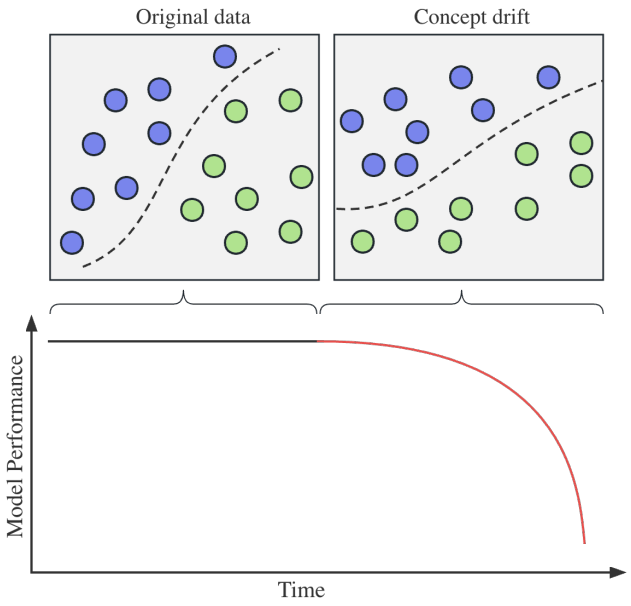


Figure 1.1: The impact of data with concept drift on the performance of the model trained with historical data deteriorates over time.

photography, and social networking, as illustrated in Figure 1.2.

The modification of behavioral patterns in real-world settings is invariably dynamic and continuous. Decision-making informed by concept drift detection results could yield substantial benefits for industry leaders or furnish critical insights for decision-makers. Proactive response to the potential implications arising from concept drift can establish a dominant position in the intensely competitive business landscape. For instance, drifts in user preferences towards folding screen or 5G mobile phones could be interpreted as instances of concept drift. Recognizing and leveraging these shifts could serve as an effective marketing strategy in the design and production of mobile

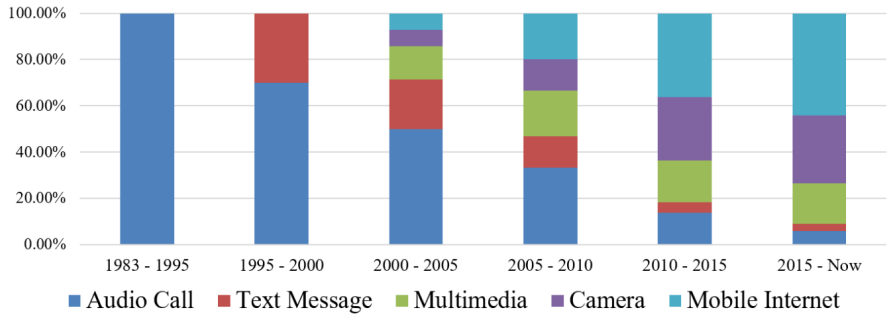


Figure 1.2: Concept drift in mobile phone usage (data used in figure are for demonstration only). (Lu et al., 2019)

devices.

Recent advancements in concept drift research have addressed complex issues such as accurately detecting concept drift in noisy and unstructured datasets, quantitatively understanding concept drift in an explainable manner, and effectively adapting to drift through the application of relevant knowledge (Dongre and Malik, 2014). These endeavors aim to equip prediction and decision-making systems with the ability to adapt to the uncertainties of a perpetually changing environment. The incorporation of concept drift techniques into fields such as data science, artificial intelligence, pattern recognition, and data stream mining has significantly elevated the capabilities of traditional machine learning approaches (Iwashita and Papa, 2019). These advancements enhance the efficacy of analogy and knowledge reasoning, paving the way for adaptive data-driven prediction and decision-making systems. In the era of big data, where data types and distributions inherently involve uncertainty, managing concept drift represents a significant challenge.

Traditional machine learning consists of two main stages: training/learning and prediction. However, research in learning under concept drift introduces three additional components: concept drift detection (identifying when drift occurs), drift understanding (comprehending the nature and scope of the drift), and drift adaptation (responding to the occurrence of drift) (Lu et al., 2019). The accurate detection of drift is pivotal for these methods to be effective. Insensitive drift detectors may fail to recognize and respond to drift in a timely manner, thus hindering the adaptation process (Liu, Song, Zhang and Lu, 2017). Conversely, overly sensitive detectors may trigger too many unnecessary model updates, leading to computational overhead and potentially compromising the model's generalization capability.

The challenge of concept drift detection is further complicated by the presence of noise. Noise can trigger false reports from the concept drift detection method, leading to unnecessary adaptations and computational costs (Harel et al., 2014). Existing drift detection methods often assume that the input is a noise-free data stream. However, in real-world applications, for example, data streams generating from the internet of things are normally contaminated with noise.

Furthermore, despite the competitive performance of deep neural networks in complex machine learning tasks compared to conventional machine learning models, their application in the field of concept drift detection remains largely unexplored (Ryan et al., 2019). This study, therefore,

aims to address the detection of concept drift in complex data streams, contributing to the broader goal of developing more robust and competitive concept drift detection methods with improved accuracy, reduced delay, and enhanced robustness.

1.2 Research Questions and Objectives

The domain of data stream mining is characterized by an incessant flow of data, inherently posing the challenge of concept drift - a phenomenon where the statistical properties of the target variable, which the model aims to predict, evolve over time. This is a critical aspect to consider in real-world dynamic environments where change is the only constant. This thesis addresses two primary challenges in understanding concept drift in data streams: the temporal identification of concept drift (When) and the characterization of its specifics (What).

The ”When” component emphasizes the ability of a drift detection method to pinpoint the exact time of drift occurrence, a fundamental function of any drift detection method.

The ”What” component delves into the nature of the concept drift, providing a more comprehensive picture. It focuses not only on the time of drift occurrence but also on its endpoint, duration, and severity. By achieving a more profound understanding of the concept drift, we can develop more appropriate responses to it.

Based on these two research challenges, this research aims to develop a set of concept drift detection algorithms for learning with streaming data and will answer the following five research questions.

Research Question 1: *How can we detect concept drift in multi-label data streams and enhance its drift detection performance?*

Concept drift inherently occurs in data streams, yet the drift detection performance is often constrained by the properties of these data streams. A significant portion of existing literature concentrates on concept drift detection within the context of binary classification problems. How can we extend and adapt the indicators designed for binary classification problems to serve as effective indicators for multi-label classification problems?

Research Question 2: *How can we accurately detect concept drift in the presence of noise-*

induced false drift within data streams?

Concept drift detection methods identify changes or ”drifts” in data streams. Existing detection methods often assume that the input is a noise-free data stream. However, real-world applications, such as those involving data streams generating from the Internet of Things, are typically contaminated with noise (i.e., class noise and/or attribute noise). Furthermore, no current drift detection method provides quantitative information about the impact of different types of noise and strategies to filter out false drift caused by such noise.

Research Question 3: *How can changes within the machine learning model itself serve as an indicator for drift detection, as opposed to relying on model performance?*

The extraction of meaningful information from data streams holds considerable significance in a wide array of real-world applications, especially in the field of big data. Nonetheless, conventional machine learning models often encounter unique obstacles when processing data streams. These challenges stem from the inherent attributes of data streams, such as their potential for limitless real-time data volume and the changing nature of data, often termed as concept drift. Existing concept drift detection methods largely depend on the model's performance as an indicator, neglecting the changes within the model itself. A more beneficial approach could entail deepening the exploration and understanding of changes within the model, which may provide more insightful indicators for drift detection.

Research Question 4: *How to apply deep neural networks to concept drift detection while avoiding excessive computational requirements?*

Several concept drift detection methods have been proposed to address issues caused by real concept drift. The goal is to detect drift after its occurrence and then adjust the model based on the detection results to adapt to the new data stream. However, existing methods usually monitor only the model's output performance changes to determine whether drift has occurred. These models are usually based on conventional machine learning. But contemporary data streams often have more complex structures. Deep learning models, which are robust and have strong fitting abilities, haven't been widely applied in the field of concept drift detection. However, the challenge is that deep learning typically requires more computational resources and time, which could lead to delays that are unfavorable for timely drift detection.

Research Question 5: *How can we gain a more comprehensive understanding of concept drift beyond simply determining its start time?*

Concept drift, a common phenomenon in data streams, requires both detection and deep understanding, as it implies that the statistical properties of a target variable, which the model aims to predict, change over time in unpredictable ways. Current detection methods predominantly aim to identify the drift start time but often lack a comprehensive understanding of data streams. This results in a loss of essential drift information, such as drift end time, drift duration, and drift severity. Therefore, how can we develop methods that provide a more holistic understanding of concept drift?

Objective 1: *To develop a novel concept drift detection method which extends the indicator design for binary classification to broader scenarios for drift detection.*

This objective corresponds to research question 1. Existing concept drift detection methods mainly rely on the error of the model as an indicator for drift detection. For those indicators, designed for binary classification problems are rarely used. The major limitation of these methods is that they less discover potential of not design for multi-label indicator for concept drift detection.

Objective 2: *To develop a novel noise-tolerant method for concept drift detection.*

This objective corresponds to research question 2. Existing drift detection methods often assume that the input is a noise-free data stream. However, in real-world applications, for example, data streams generating from the Internet of Things are normally contaminated with noise (noise, i.e., class noise and/or attribute noise). Noise can significantly hinder the performance of data mining models. Noise has an adverse impact on both the interpretation of the data, and reduction in the performance of the machine learning model. However, the majority of existing drift detection methods assume no noise.

Objective 3: *To develop an indicator based on prediction stability of the incremental learning model for concept drift.*

This objective corresponds to research question 3. Traditional drift detection methodologies predominantly concentrate on monitoring the output of models trained on historical data, typically represented by error rates. However, before the change happens on the back-end of the model,

which is the output of the model, the model itself will change before the output. However, few works have been done to utilize the model itself for drift detection. Instead of detecting concept drift by monitoring the output of concept drift, this study proposes a model prediction stability-based concept drift detection method, which could detect the drift by monitoring the change in the model itself.

Objective 4: *To develop a deep neural network-based drift detection method, where the computational restriction is solved by transfer learning.*

This objective corresponds to research question 4. Existing works usually only monitor the output performance changes of the model to judge whether drift occurs. These models are usually conventional machine learning models, but today's data streams have more complex structures. Compared with traditional machine learning, the deep learning model with stronger fitting ability and robustness has not been applied in the field of concept drift detection as far as we know. However, the problem is that deep learning usually requires more computing resources and time. Under the same computing resources, the processing time is slower than that of conventional shallow models with simple structures. Excessive delay is not conducive to drift detection.

Objective 5: *To develop a novel quadruple-based method for understanding concept drift.*

This objective corresponds to research question 5. Existing detection methods predominantly aim to identify the drift start time, which lack comprehensive understanding of data streams, leading to a loss of drift information. The proposed method should offer a more detailed analysis of concept drift through the use of quadruples, encompassing drift start, drift end, drift severity, and drift type. In addition, the concept drift detection method is largely reliant on conventional machine learning methods, however, the ability of the deep neural network is rarely discovered. The main reason is that the timing requirement for the concept drift is important, however, the deep neural network usually requires significant computational resources, therefore minimizing the consumption of computational resources and processing time is critical for concept drift detection.

1.3 Research Contributions

This thesis presents a comprehensive analysis of concept drift detection problems from multiple perspectives. The main contributions of this study are succinctly summarized as follows:

1) A novel drift detection method for multi-label data stream classification.

A novel drift detection method is proposed for handling multi-label data streams. This method extends the Drift Detection Method (DDM) with an additional false positive rate indicator. The proposed method effectively detects concept drift in multi-label classification data streams and offers approximately 50% performance improvement compared to the widely used DDM. (Chapter 3)

2) Introduction of three performance indicators to evaluate drift detection performance.

Three performance indicators are proposed to evaluate the effectiveness of drift detection. These indicators are designed to determine whether drift detection is conducted within a reasonable time frame and to estimate the duration until the known drift starting point. This approach provides a more comprehensive evaluation of drift detection performance, as opposed to relying solely on drift delay, which only provides a narrow view of the performance. By using these three indicators, we can gain a better understanding of the overall performance of drift detection. (Chapter 4)

3) A Novel Noise-Tolerant Concept Drift Detection Method for Data Stream Mining.

A novel noise-tolerant drift detection method is proposed to identify concept drift within noisy data streams. The study begins by providing a quantitative definition and evaluation of noise. To address the issue of false concept drift, the method employs a two-step detection and validation function. This function identifies drifts and filters out false drifts induced by noise. In the validation step, a sparse sliding time window is introduced to further eliminate false drift and expedite the detection process. The proposed method demonstrates superior detection performance compared to existing methods. (Chapter 4)

4) A Concept Drift Detection Method Based on Tracking Weighted Prediction Confidence of Incremental Learning.

A novel model prediction stability-based indicator is proposed. Instead of relying solely on the performance of the model as a drift detection indicator, this indicator combines model performance with model prediction stability to detect drift. Additionally, an incremental ensemble learning method based on a voting mechanism is used to continuously update the value of the indicator. This method exhibits significant capabilities for drift detection. (Chapter 5)

5) A Model-Centric Transfer Learning Framework for Concept Drift Detection.

A novel model-centric transfer learning framework is proposed for drift detection. This framework uses a deep neural network to detect drift by monitoring changes in the network parameters. It applies transfer learning to speed up the drift detection process and reduce computational complexity by freezing parts of the network. To further minimize false positive detections, a method using long and short time windows is introduced to distinguish actual drift from potential detected drift. Experiments on real-world and artificial datasets demonstrate the effectiveness of the proposed framework. (Chapter 6)

6) A Quadruple-Based Approach for Understanding Concept Drift in Data Streams.

A novel Quadruple-based Approach for Understanding Concept Drift in Data Streams (QuadCDD) framework is proposed. This framework not only detects and predicts the starting point of concept drift, but also provides a detailed analysis of concept drift using quadruples, which encompass drift start, drift end, drift severity, and drift type. The framework employs quadruples to facilitate informed decision-making and appropriate actions to handle various concept drifts, thereby effectively maintaining high and stable performance in data streams with concept drift. Experimental results validate the effectiveness of the QuadCDD framework in accurately detecting and understanding concept drifts, as well as in preserving the stability and performance of models in the presence of these drifts. (Chapter 7)

1.4 Thesis Structure

The logical structure of this thesis (the chapters and the corresponding research questions) and the relationship between the chapters are shown in Figure 1.3. The main contents of each chapter are summarized as follows:

CHAPTER 2 reviews the existing literature on data stream mining and concept drift detection methods, thereby identifying notable gaps in the current body of research. This chapter introduces the fundamental problem of concept drift and lays out the foundational procedures for adapting to concept drift. It further provides a comprehensive categorization of existing algorithms, organized according to their unique implementation details. The chapter culminates in an insightful discussion on the limitations inherent to the algorithms reviewed, setting the stage for the innovative

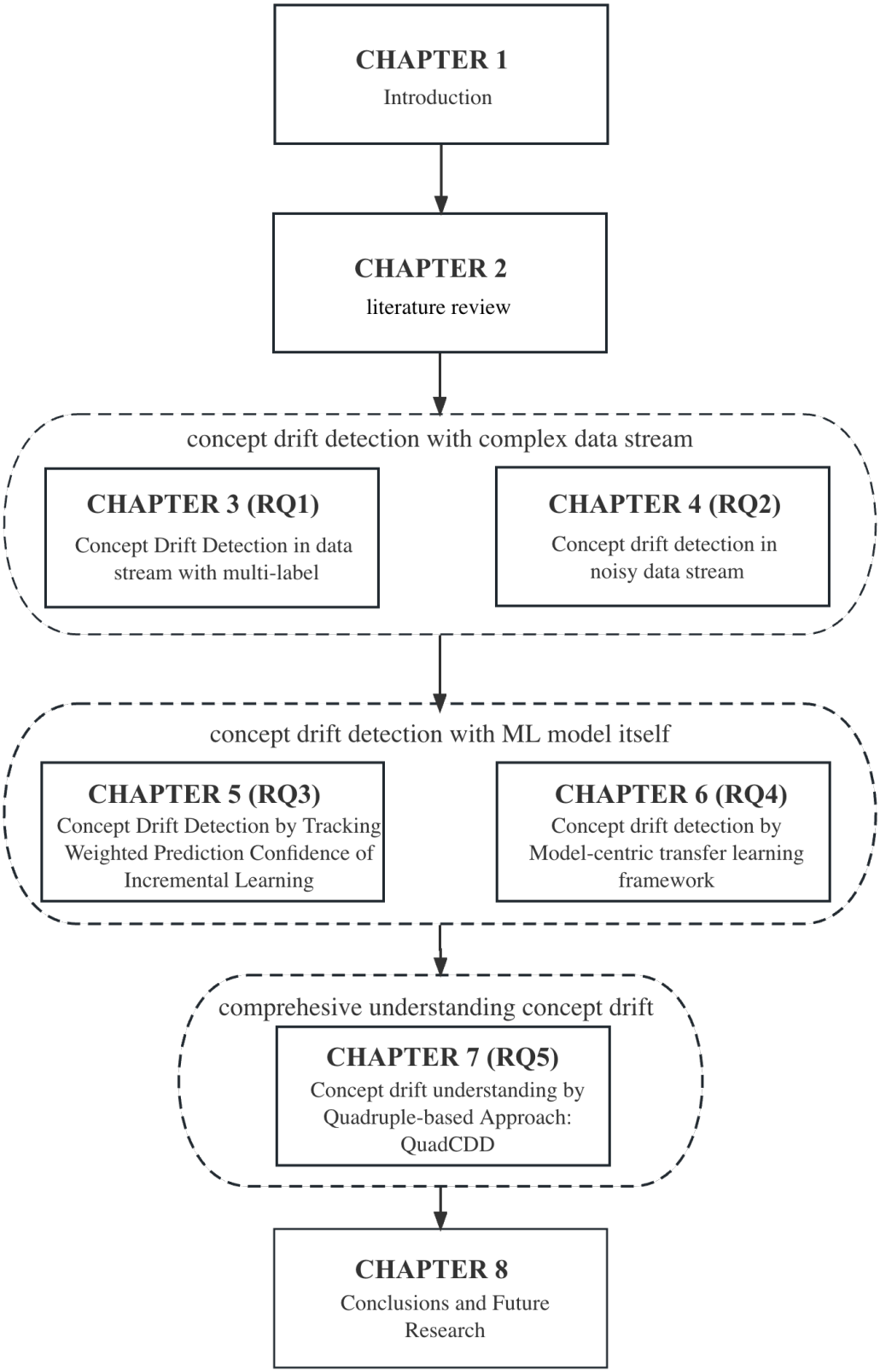


Figure 1.3: Thesis structure and relationship between chapters

solutions presented in subsequent chapters.

CHAPTER 3 introduces a method for detecting concept drift in multi-label data streams. This method enhances the capabilities of the DDM, enabling it to perform concept drift detection in multi-label data streams while accounting for the false positive rate. This innovative approach directly addresses the *Research Question 1* and achieves the *Objective 1*. The primary contribution lies in broadening the application of the DDM to more complex scenarios, specifically in contexts where data streams are multi-labeled. As a result, this method achieves a higher degree of accuracy in drift detection.

CHAPTER 4 presents a concept drift detection method for noisy data streams. Initially, the types and quantitative definitions of noise are introduced. Subsequently, three metrics for evaluating the performance of concept drift detection are proposed. The proposed drift detection method is then tested on data streams that contain both concept drift and varying levels of noise. Finally, the performance of drift detection is evaluated using the proposed metrics. This innovative approach directly addresses the *Research Question 2* and achieves the *Objective 2*. The principal contribution is the enhancement of the concept drift detection method, enabling it to effectively identify drifts within a noisy environment. This refined approach significantly reduces the impact of false drifts triggered by noise, thereby improving the overall accuracy and reliability of the detection process.

CHAPTER 5 introduces a concept drift detection method that employs a conventional machine learning approach for classification models. This method detects drift when there is a significant change in prediction confidence. The change in prediction confidence in the machine learning model is monitored and used as an indicator of concept drift. This innovative approach directly addresses the *Research Question 3* and achieves the *Objective 3*. The key advancement is exploiting the inherent characteristics of the model for drift detection, rather than predominantly relying on the model's outputs. This method provides a more comprehensive and potentially more precise approach to identifying shifts in data patterns.

CHAPTER 6 presents a model-centric framework for concept drift detection. This framework utilizes a deep neural network to detect drift by monitoring changes within the network, as reflected by the parameters within the nodes. Additionally, a transfer learning strategy is adopted to reduce

computational costs and maintain a low delay in drift detection. This innovative approach directly addresses the *Research Question 4* and achieves the *Objective 4*. The primary contribution of this research is the pioneering application of deep neural networks to concept drift detection. The utilization of transfer learning results in a substantial reduction of computational overhead, underscoring the efficiency of this approach. Furthermore, the improved detection performance attests to the potency of deep neural networks in managing complex data dynamics.

CHAPTER 7 presents a novel quadruple-based Approach for Understanding Concept Drift in Data Streams (QuadCDD) framework that not only detects and predicts the concept drift start point but also offers a more detailed analysis of concept drift through the use of quadruples, encompassing drift start, drift end, drift severity, and drift type. Our framework employs quadruples to enable informed decision-making and adopt appropriate actions to handle various concept drifts, effectively maintaining high and stable performance in data streams with concept drift. This innovative approach directly addresses the *Research Question 5* and achieves the *Objective 5*. The principal contribution of this research lies in fostering a more comprehensive understanding of concept drift. This enriched perspective enables further refinements to be more targeted and efficient, paving the way for more effective strategies in handling data stream dynamics and changes over time.

CHAPTER 8 summarizes the findings of this thesis and points to directions for future work.

Chapter 2

Literature Review

This chapter focuses on using concept drift detection methods to detect concept drift 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 data stream mining. Then, the concept drift problem is introduced including its definition, types, and applications in Section 2.2. Concept drift detection techniques are comprehensively reviewed in Section 2.3.

2.1 Data Stream Mining

This part provides a comprehensive overview of data stream mining, which serves as the foundation for developing an in-depth understanding of the concept drift problem.

2.1.1 Evolving Data Stream

Data streams, characterized by a relentless flow of information that can often occur in real time, mark a departure from traditional static datasets (Krawczyk et al., 2015). Where traditional data processing relies on storing and then analyzing data, data streams necessitate on-the-fly analysis. This fundamental shift is emblematic of our age, where information flows without pause, reflecting various real-world scenarios such as financial transactions, social media interactions, sensor data in industrial applications, and intricate web server logs.

Within the realm of computer science, particularly in data stream mining, the endeavor to dis-

till knowledge structures represented in models and patterns from this continuous data is both ambitious and complex. It necessitates the creation of algorithms capable of processing and scrutinizing data as it emerges, often without the ability to store all of it or even to process it more than once.

This nature of data streams presents formidable challenges that define the cutting edge of data science (Wares et al., 2019):

Efficiency: Algorithms must be robust and lean, adept at keeping pace with the rapid flow of data, and economical in their use of memory. The limits imposed by a data size that could possibly be infinite further complicate this situation, making it necessary to combine mathematical sophistication with computational prowess.

Adaptability: The notion of concept drift is central to the understanding of data streams. As underlying patterns in data shift over time, algorithms must not just detect but adapt to these changes. This requires a nuanced understanding of statistical variations and an ability to recalibrate models dynamically.

Noise and Outliers Handling: The inherent noise and outliers within data streams can distort the quality of extracted knowledge. Building mechanisms to identify and minimize the impact of these anomalous elements is a multifaceted problem that combines aspects of statistical theory, machine learning, and practical engineering.

Despite these complexities, the exploration of data stream mining is far from an academic curiosity. It is a critical component of modern data analysis. In a world that increasingly operates in real-time, the capacity to extract immediate insights can translate into tangible advantages. Whether in financial markets, where milliseconds can mean millions, or in healthcare, where real-time analysis of patient data can save lives, the applications are as profound as they are pervasive.

Furthermore, the alignment of data stream mining with emerging technologies like the Internet of Things (IoT) opens new horizons. Whether in smart cities, where sensor data can be harnessed for more efficient urban planning, or in manufacturing, where real-time analytics can optimize production, the applications are virtually limitless.

In academia, the study of data stream mining serves as a rich platform for research, inspiring

new methodologies, theories, and applications. It offers fertile ground for multidisciplinary collaboration, linking computer science with mathematics, engineering, business, and social sciences.

In conclusion, data stream mining represents a vibrant intersection of theoretical complexity and practical applicability. Its challenges are formidable, yet the pursuit of solutions to these challenges is unlocking new frontiers in real-time data analysis, transforming industries, and enriching our understanding of the very nature of information. As scholars, practitioners, and innovators, our continued exploration of this field promises to shape not just the future of computer science, but the fabric of our data-driven world.

2.1.2 Data Stream Mining

Data stream mining has recently emerged as a crucial research domain within computer science, encapsulating the evolving demands and complexities of the modern, data-driven technological landscape (Gaber, 2012). With an exponential growth in data generation from various sources like sensors, social media, financial systems, and more, traditional data mining methods have shown limitations. In response, data stream mining has positioned itself at the intersection of real-time analysis and Big Data processing, making it essential for many contemporary applications such as fraud detection, trend analysis, and Internet of Things (IoT) monitoring.

The principle underlying data stream mining is the continuous analysis of data as it is generated. Unlike traditional data processing methods where information is collected, stored, and then analyzed, data stream mining requires handling data sequentially and continuously. This necessitates a new paradigm in algorithm design, as data must be processed in the exact order it arrives, often without the possibility of revisiting past elements.

Two central classifications within data streams are static (or stationary) and evolving streams (Masud et al., 2009). Static streams are characterized by a uniform underlying pattern, consistent over time. Evolving streams, on the other hand, can undergo changes, altering how the data is analyzed and interpreted. This dynamic behavior is often referred to as concept drift, and addressing it effectively is a foundational challenge in data stream mining.

Adapting existing machine learning and data mining techniques to the demands of data streams

is complex. Traditional algorithms are typically designed for batch processing, treating data as a whole. However, data streams, due to their inherent nature, require what's often termed one-pass learning. It demands the creation of specialized algorithms, which can work with ever-changing data without losing sight of shifts in underlying patterns. This area opens up fertile ground for research, with ongoing developments in areas like online learning and adaptive modeling.

The practical constraints imposed by data streams further complicate matters. The sheer volume and velocity of data can be overwhelming, leading to potential hardware limitations. Memory management becomes particularly challenging, as the infinite nature of data streams makes continuous storage impractical. Algorithms must thus be crafted with computational efficiency in mind, balancing speed with memory constraints.

In conclusion, data stream mining stands as an intricate, multifaceted discipline within computer science. It is a field marked by continuous innovation, driven by the ever-increasing complexity of our data-centric world. The theoretical challenges, coupled with the practical considerations, make it a dynamic area of study, rich in academic exploration, and vital in real-world application. Its role in shaping our understanding of real-time data processing is central to our ongoing journey towards harnessing the power of information in an increasingly interconnected and fast-paced world.

2.1.3 Background and Concepts

Data stream mining is an intricate field that expands the boundaries of the conventional data mining concept, which is traditionally focused on static databases (Dubuc et al., 2021). Unlike its traditional counterpart, data stream mining deals with data that is continuously produced at a rapid rate, often necessitating real-time processing and analysis. In this complex field, there are several key concepts including data streams, concept drift, real-time analytics, among others, that offer an understanding of its foundational theory and unique challenges.

A data stream is defined as a sequence of digitally encoded coherent signals used to transmit or receive information that is currently being transmitted. These data streams are ubiquitous in our digital age and originate from a myriad of sources including online transactions, IoT devices, social media, network logs, sensors and many more. Their inherent nature—high-speed, infinite length, and sequential order—introduces a set of unique challenges. Data stream mining algo-

rithms must not only manage these characteristics, but also make intelligent decisions on the fly without compromising the speed and accuracy of their predictions.

Concept drift is a key concept that is inextricably linked to data stream mining (Lu et al., 2019). It refers to the phenomenon where the statistical properties of the target variable, which the predictive model is attempting to estimate, evolve over time in unexpected ways. In a real-world setting, it's common for the underlying concept of data to shift due to various factors such as changing consumer behaviors, market conditions, seasonal variations, among others. These changes often result in the model's predictive performance degrading over time, as the assumptions upon which the model was originally trained no longer hold. The ability to detect and swiftly adapt to concept drift is crucial for maintaining the relevance and accuracy of the data stream mining models.

Real-time analytics is another critical concept within this field. It pertains to the process of using various tools and techniques to extract actionable insights from data as it enters the system, without any significant delay. In today's fast-paced digital economy where data is continually generated, the ability to analyze and interpret data in real-time can provide businesses and organizations with a significant competitive advantage. It allows them to react quickly to changing circumstances, anticipate future trends, make timely decisions, and ultimately deliver better outcomes.

In addition to the aforementioned key concepts, there are other notions that are worth understanding in the context of data stream mining. One such concept is data pre-processing, which includes techniques used to clean and prepare the incoming stream data for analysis. Given the noisy and unpredictable nature of real-time data, pre-processing can be a vital step to enhance the quality of the data and thus, the reliability of the analytics.

Another relevant concept is the sliding window model. It represents a specific strategy to manage the potentially infinite length of data streams. The idea is to only consider a window of the most recent data items for analysis, effectively discarding older information (Yu et al., 2019). The size of the window and the mechanism of its movement (whether it's sliding or expanding) can significantly impact the performance of the mining process.

Also, the use of summary statistics and data structures to capture relevant information about the data stream is another noteworthy approach (Lu et al., 2019). The continuous and rapid generation of data in streams makes it impractical to store all the data points. Summary statistics like count,

sum, mean, variance, min, max, etc., can provide concise and computationally efficient summaries of the data stream.

In the context of concept drift, it's important to understand the different types of drifts such as sudden, incremental, gradual, and recurring. Each type of drift represents a different way in which the underlying concept can change, and they each require different strategies for detection and adaptation.

Lastly, ensemble methods (Krawczyk et al., 2017) are often used in data stream mining to improve predictive performance and robustness against concept drift. Ensemble methods combine the predictions of multiple base learners to make a final prediction, and they have been found to be particularly effective at dealing with concept drift since different learners can specialize in different regions or periods of the concept space.

In conclusion, a comprehensive understanding of these concepts forms the backbone of data stream mining and provides the essential framework to tackle the inherent complexities and evolving nature of continuous data streams. The rapid advancements in technology and data generation only heighten the relevance and necessity for continued exploration and research within this field.

2.1.4 Data Stream Mining Techniques

Data stream mining techniques span a diverse spectrum of methods that are specifically devised to manage continuous, voluminous, and high-speed data. These techniques, each having its distinct features and applications, are designed to extract actionable insights and make predictions using streaming data. They include Classification, Clustering, Frequent Pattern Mining, Anomaly Detection, and Concept Drift Detection among others.

Classification is a fundamental technique in data stream mining that involves assigning data elements to pre-specified classes or categories (Zyblewski et al., 2021). Notable for its usage in various fields such as spam detection, sentiment analysis, and customer segmentation, it requires robust algorithms capable of handling immense volumes of data and adapt to concept drift. One such notable algorithm is the Hoeffding Tree algorithm, admired for its efficiency in managing large datasets with limited computational resources. The classification in data stream mining gen-

erally follows a two-step process: model construction using training data and model usage to classify new data. Handling the continuous influx of data and adapting to its changing nature over time are core necessities for these classification algorithms.

Clustering, another pivotal technique, involves grouping data elements based on their similarity (Liu, Song, Zhang and Lu, 2017). This process is indispensable in various applications such as customer segmentation, image segmentation, anomaly detection, and social network analysis. Given the dynamic nature of streaming data, clustering poses unique challenges, necessitating the use of techniques that can continuously update and adapt the cluster formation as new data comes in. One of the well-known clustering algorithms for data streams is StreamKM++ (Ackermann et al., 2010).

Frequent Pattern Mining, a critical process in data stream mining, entails identifying patterns or itemsets that appear frequently within the data stream. These could include sub-sequences or substructures. Given the high-speed, continuous nature of data streams, it becomes crucial for the algorithms to not only identify these frequent patterns but also to update them efficiently as new data enters the stream. Frequent Pattern Growth (FP-Growth) and Apriori are some of the widely-used algorithms for frequent pattern mining in data streams (Kaur and Jagdev, 2017).

Anomaly Detection, an important method in data stream mining, identifies data elements that deviate significantly from the normal behavior. This technique finds its applications in various domains such as credit card fraud detection, network intrusion detection, and health monitoring systems. The high-speed, high-volume characteristics of data streams make anomaly detection a complex task. Algorithms need to identify anomalies in real-time and be capable of adapting to changes in the data. Local Outlier Factor (LOF) and Histogram-based Outlier Score (HBOS) are among the popular anomaly detection algorithms in the realm of data stream mining (Alghushairy et al., 2020).

Lastly, Concept Drift Detection plays an integral role in data stream mining. This technique focuses on identifying and adapting to the changes in the underlying data distribution over time - a common occurrence in data streams. The process becomes crucial as changes or 'drifts' can significantly impact the predictive performance of the mining models. Algorithms designed for concept drift detection need to not only identify these changes in real-time but also adapt the min-

ing process accordingly to maintain the accuracy and relevance of the model. DDM and Early Drift Detection Method (EDDM) are among the commonly-used algorithms for concept drift detection (Lu et al., 2019).

2.1.5 Challenges

Despite the transformative potential of data stream mining, it is riddled with a variety of challenges. Concept drift, one of the main challenges, refers to changes in the underlying patterns in the data over time. This phenomenon necessitates the development and deployment of adaptive algorithms capable of adjusting to the ever-changing nature of data streams.

High-speed data streams are another significant challenge. The vast volume and high velocity of data streams require scalable frameworks and sophisticated algorithms capable of processing these streams without causing a bottleneck in the system. This implies that the algorithms should not only be able to handle the fast-paced nature of the data streams but also provide accurate results in real-time.

Additionally, the potentially infinite length of data streams presents a considerable challenge. Given that data streams can theoretically extend indefinitely, algorithms and techniques designed for data stream mining must be equipped to handle the unbounded size and varying speed of arriving instances from a data stream. Efficient data summarization techniques such as sampling, sketching, or windowing can be used to circumvent this problem by providing a manageable snapshot of the data stream at any given point in time.

Resource constraints further compound these challenges. The continuous nature of data streams imposes strict restrictions on the learning algorithms, particularly in terms of memory usage and processing time. Algorithms are expected to induce a model incrementally, keep up with the speed of arriving instances, and use only a constant amount of memory, even as the volume of data increases. To mitigate these issues, incremental learning methods like batch-incremental learning, and distributed processing frameworks are often employed.

Other challenges include handling the noisy nature of data streams, dealing with missing or delayed data, and maintaining privacy and security in the face of real-time data processing.

The complexity of these challenges underscores the need for continued research and innovation in

the field of data stream mining. While many strategies have been proposed and are currently in use, the evolving nature of data streams and their applications continue to pose new problems and require innovative solutions. Researchers are continuously striving to develop new methods and enhance existing approaches to keep up with these dynamic challenges and maximize the potential benefits of data stream mining.

2.2 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 Definition of Concept Drift

Concept drift refers to the phenomenon of changing data distribution. The problem of concept drift arises in a data stream when newly arrived data instances may exhibit a different pattern from the previous data. 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.

A widely accepted definition of concept drift is $p_{t+1}(X, y) \neq p_t(X, y)$. According to the properties of probability, concept drift can be decomposed into two parts whereby $p(X, y) = p(y|X) \times p(X)$. Real drift refers to the changes in the posterior probabilities, i.e., changes in $p(y|X)$, and virtual drift denotes the changes in $p(X)$. Existing research mainly focuses on the real concept drift problem.

However, limited research explains the term "concept". A concept is formally defined as a many-to-one mapping function $X \rightarrow y$. Many machine learning algorithms require a many-to-many mapping from 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. Unlike noise series, concept drift is influenced by hidden changes in the context. 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 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.

Online prediction tasks become particularly difficult if real concept drift occurs, i.e., the changes in $P(y|X)$. 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 X is independent of the model parameters.

The notion of concept drift is multifaceted, with several terms and definitions attributed to it in the literature. Some authors have referred to this phenomenon as "dataset shift" or "concept shift," and various related terminologies have been proposed. In certain research works, concept drift has been designated as one specific subcategory of dataset shift. Dataset shift itself has been subdivided into three primary components: covariate shift, prior probability shift, and concept shift. While these definitions have helped delineate the scope of various research topics, there is a recognized association of concept drift with covariate shift and prior probability shift.

An increasing body of literature has gravitated towards defining the problem of concept drift using the mathematical expression: $\exists t : P_t(X, y) \neq P_{t+1}(X, y)$. This definition succinctly captures the essence of concept drift, encompassing the temporal variations in the joint probability distribution of the features X and the target variable y. The present survey also adheres to this definition, allowing for a consistent interpretation across different contexts.

Concept drift at time t signifies a shift in the joint probability distribution of X and y at that particular time. The expression $P_t(X, y)$ can be broken down into two components: $P_t(X, y) = P_t(X) \times P_t(y|X)$. Consequently, three sources of drift can trigger concept drift:

Source I: Characterized by $P_t(X) \neq P_{t+1}(X)$ while $P_t(y|X) = P_{t+1}(y|X)$. In this scenario, the focus lies on the drift in $P_t(X)$, while $P_t(y|X)$ remains static. Often termed as "virtual drift," this form does not induce changes to the decision boundary.

Source II: Defined by $P_t(y|X) \neq P_{t+1}(y|X)$ while $P_t(X) = P_{t+1}(X)$. Here, the drift causes alterations to the decision boundary, potentially leading to decrements in learning accuracy. This is commonly referred to as "real drift."

Source III: This represents a fusion of Source I and Source II, where both $P_t(X) \neq P_{t+1}(X)$ and $P_t(y|X) \neq P_{t+1}(y|X)$. It embodies a multifaceted drift encompassing changes in both $P_t(y|X)$ and $P_t(X)$, reflecting intricate dynamics within the learning environment.

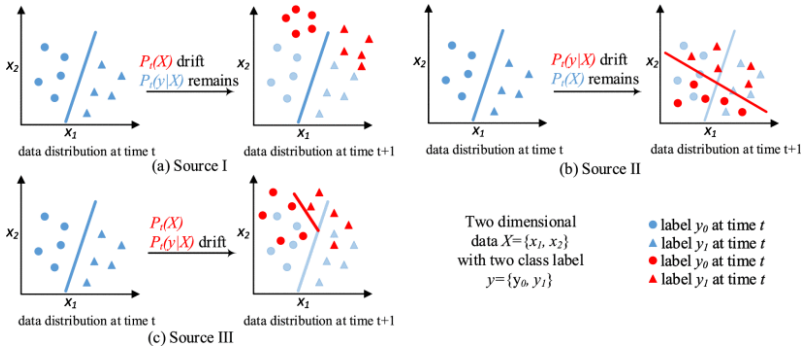


Figure 2.1: Illustration of the three sources of concept drift. (Lu et al., 2019)

The visual representation in Figure 2.1 elucidates the distinctions among these three sources within a two-dimensional feature space. Source I pertains to the drift in feature space, while Source II involves changes in the decision boundary. In practical applications, it is not uncommon for Source I and Source II to occur in conjunction, resulting in the more complex Source III.

These distinctions offer critical insights into the dynamic nature of learning in non-stationary environments. They also pave the way for the design of robust algorithms that can recognize and adapt to these different forms of drift, fostering resilience and accuracy in predictive modeling across various domains.

2.2.2 Types of Concept Drift

Concept drift is a crucial factor to consider in data stream mining, as it refers to the dynamic changes in the underlying relationships or distributions within the data over time. The concept drift in data streams can be typically categorized into four principal types as illustrated in Figure 2.2. Each type has its unique characteristics, manifestations, and requires specific techniques to handle effectively. These types include Abrupt (or Sudden) Drift, Incremental Drift, Gradual Drift, and Recurring Drift (Lu et al., 2019).

Abrupt(or Sudden) Drift: As the name suggests, sudden drift is characterized by an abrupt shift from one concept to another, effectively transforming the underlying pattern. This type of drift represents an instantaneous change, with the former pattern being replaced completely by a new

one without any transitional phase. A prominent example of sudden drift could be observed in the stock market, where sudden and unpredictable events, such as policy changes or major announcements from influential companies, could trigger a swift change in the market trends. For instance, a major breakthrough in battery technology could lead to an unexpected surge in the demand for electric vehicles, thus causing a sudden shift in the auto industry's market trends.

Incremental Drift: In contrast to the sudden drift, incremental drift reflects a gradual transition from one concept or pattern to another. This type of drift is often marked by a series of intermediary patterns that appear during the transition, effectively representing a phase of constant change. An illustration of incremental drift could be the slow transformation of a user's online browsing behavior over time due to evolving interests, growing knowledge, or changing needs. Such a gradual shift might be influenced by various factors, including age, lifestyle changes, or exposure to new information or technologies.

Gradual Drift: Gradual drift signifies a scenario where the emergence of a new pattern is frequently interspersed with reversion to the previous pattern. However, over time, the frequency of reverting back to the old pattern decreases, thereby indicating a slow but definitive change in the data concept. A pertinent example of gradual drift can be seen in public health, where the prevalence of a certain disease in a population might gradually decrease over time due to improved treatment methods or health policies. Nonetheless, there might be intermittent spikes in the disease prevalence due to seasonal factors, unexpected outbreaks, or emergence of new strains, which epitomize the fluctuations characterizing the gradual drift.

Recurring Drift: Recurring drift represents a unique case where previously observed patterns or concepts re-emerge in the data. This type of drift is common in cyclical or seasonal data, where the same trends or patterns are expected to occur periodically. Examples of recurring drift can be frequently observed in various domains such as economics, transportation, and meteorology. For instance, seasonal employment trends where certain industries hire more employees during specific periods (e.g., retail industry during holiday seasons), daily rush hour traffic patterns in urban areas, and seasonal weather patterns (e.g., patterns of rainfall or temperature variations across different seasons) all illustrate recurring drifts.

It's worth highlighting the nuanced difference between incremental drift and gradual drift. Incre-

mental drift specifically refers to scenarios where individual data instances progressively modify their values over time, leading to an overall shift in the data concept. Conversely, gradual drift signifies changes that involve fluctuations in the class distribution of the preceding data, causing an overall pattern shift interspersed with occasional reversions to the previous pattern.

Each of these types of concept drift poses its unique set of challenges and complexities in data stream mining. Accordingly, they require specialized handling strategies and tailored algorithmic approaches to effectively analyze the data streams under the influence of these drifts. A deep understanding of the type of concept drift occurring in a data stream is pivotal to the selection of the most fitting data mining techniques and algorithms, ensuring that the analytical results are reliable, relevant, and valuable.

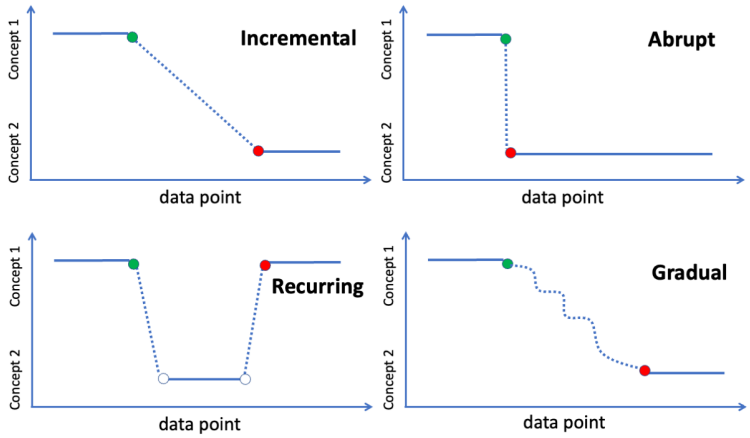


Figure 2.2: Four types of concept drift

Existing concept drift-related studies handle the 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. summarize 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.

And most of the existing concept drift detection methods only focus on the start of the concept drift. But actually the drift start point is only part of the information of different type of concept drift. The length, end and severity of drift is rarely discovered.

2.2.3 Concept Drift Applications

In recent years, the field of artificial intelligence (AI) has witnessed a fascinating and complex phenomenon known as concept drift. This intricate process, marked by alterations in decision boundaries within a data stream over a period, has emerged as a critical element in modern machine learning and predictive modeling. It mirrors the shifts in underlying data patterns, reflecting the dynamism and complexity of the world's ever-changing informational landscape.

Concept drift detection is not a mere academic exercise; it's a vital tool for applications where data changes dynamically, and where real-time or near-real-time analysis is needed. Such analysis plays a pivotal role in various domains, ranging from machine learning, data mining, predictive analytics to healthcare, finance, and beyond. The applications are as diverse as they are impactful, demonstrating the universality and importance of concept drift detection in modern data science.

Several methods are employed to identify concept drift, each with its distinctive characteristics and applications. These include statistical approaches that apply mathematical rigor, statistical process control methods that ensure quality and consistency, temporal window-based strategies that offer flexibility in monitoring data streams, and deep neural network-constructed methodologies that bring the power of deep learning into the fold. Each technique has its unique merits, capable of managing specific types of drift and model conditions. These methods' diversity underscores the multifaceted nature of concept drift and the need for a tailored approach in addressing it.

Understanding and implementing these techniques require a deep dive into the theoretical foundations and practical applications of concept drift detection. Academics, researchers, and practitioners must familiarize themselves with these detection methods to adopt the best-suited measure for each specific drift type and model. Expanding knowledge through careful study, hands-on experimentation, and engagement with relevant literature will aid in this endeavor, fostering a more nuanced understanding of the subject.

Real-world applications of concept drift detection are numerous and growing. In the fast-paced world of mobile application stores, concept drift detection plays a vital role. Through the use of classification models and specialized detectors, it's possible to monitor shifts in underlying data patterns in real or near-real time. This continuous monitoring is essential in today's world, where

trends shift rapidly, user behavior changes, and technology evolves at a breakneck pace.

But the significance of concept drift detection extends well beyond the digital realm. In traditional industries such as finance, where market trends, customer behavior, and economic landscapes are in constant flux, concept drift detection stands as a bulwark against uncertainty. It enables financial models to adapt to changes swiftly and efficiently, ensuring accuracy and reliability.

In healthcare, another domain that experiences constant change, concept drift detection holds great promise. As patient information changes due to the progression of diseases, the effects of treatment, or changes in medical understanding, the ability to detect and adapt to these changes is paramount. Concept drift detection provides a method to ensure that medical diagnostic and prognostic models maintain their accuracy, even as the landscape of medical knowledge and patient care evolves.

The educational field, too, can benefit from concept drift detection, where personalized learning paths and dynamic curriculum adjustments can be tailored based on individual student progress and changes in educational standards. This demonstrates the wide-ranging impact and potential applications that extend across various domains of human endeavor.

In summary, concept drift detection is more than a technical term; it's a foundational principle that ensures the continuous accuracy and efficiency of machine learning models amidst shifting decision boundaries. Its importance resonates across various industries, reflecting the ever-changing nature of data and the need for models that can adapt accordingly. As the field of Artificial Intelligence (AI) continues to advance and expand, the study and implementation of concept drift detection will undoubtedly remain a vital area of research and application, reinforcing its position as an indispensable tool in modern data science. Its multifaceted applications and intricate complexities make it a compelling subject for continued exploration and innovation.

2.3 Concept Drift Detection

The concept drift detection refers to methods for determining the occurrence of drift. Drifts are frequently caused by changes in the environment, which might lower the accuracy of a machine learning model that had previously been fitted. Usually, these drifts affect the way the machine learning model performs or the statistical characteristic both before and after the drift. There has

been some works done for learning with drift in review (Lu et al., 2019) and (Krawczyk et al., 2017). These approaches may be broadly split into four categories: performance-based, data distribution-based, evolving system and multiple hypothesis drift detection methods. In addition, we looked into a variety of current studies of data stream mining on neural networks.

2.3.1 Performance-based

The performance-based drift detection approach focuses on monitoring changes in a machine learning model's performance. If there is a statistically significant rise or reduction, concept drift is detected. The most popular technique is the DDM (Gama et al., 2004), which measures the total forecast error rate to determine if drift has occurred. In contrast to the DDM, the EDDM (Baena-Garcıa et al., 2006) detects drift by measuring the distance between two adjacent prediction mistakes. It cannot function until 30 prediction mistakes have occurred, which is problematic when drift develops rapidly.

Similar applications have been made to drift detection using Heoffding's inequality based Drift Detection Method (HDDM) (Frías-Blanco et al., 2015), which uses Hoeffding's inequality to determine the crucial area of a drift. In addition, the ADaptive WINdowing (ADWIN) (Bifet and Gavalda, 2007) approach detects drift by measuring the difference between two adaptive size windows. Page-Hinkley (PH) (Li et al., 2020) finds the drift by comparing the observed values to their historical mean.

Dynamic Extreme Learning Machine (DEML) does not change the DDM detection algorithm but uses a novel base learner, which is a single hidden layer feedback neural network called Extreme Learning Machine (ELM) to improve the adaptation process after a drift has been confirmed.

Learning with Local Drift Detection (LLDD) method is an error rate-based drift detection algorithm that has been adopted and applied in concept drift detection. It modifies the overall drift detection problem by dividing it into a set of decision tree node-based drift detection problems. This means that LLDD monitors the performance of the learning system by tracking changes in the online error rate of base classifiers at the decision tree node level. If an increase or decrease of the error rate is proven to be statistically significant, an upgrade process (drift alarm) will be triggered.

2.3.2 Distribution-based Drift Detection

Distribution-based drift detection constitutes a significant category of drift detection algorithms. These algorithms employ a distance function or metric to quantify the dissimilarity between the distribution of historical data and new data. If this dissimilarity is statistically significant, the system triggers a process to upgrade the learning model.

Similar to performance-based drift detection methods, distribution-based methods discover drift when a statistically significant dissimilarity occurs. A key advantage of distribution-based methods is their ability to process both labelled and unlabelled data.

The Kolmogorov–Smirnov Test (KS) , commonly used in this context, determines if two distributions derived from two sample sets are equal. When employed as a static drift detector (Sobolewski and Wozniak, 2013), the KS test is applicable to unlabelled data. A modified version of the KS test has been proposed to detect drift progressively, enabling more effective processing of large volumes of stream data (dos Reis et al., 2016).

In addition to the KS test, clustering techniques are also popular in this field. For instance, the K-nearest neighbours algorithm is used to generate data clusters for density estimation (Liu et al., 2018). The Semi-Supervised Adaptive Novel Class Detection (SAND) framework (Haque et al., 2016) builds clusters and uses them to decide if incoming data belong to the original cluster; if not, the new cluster is considered a new drift. These approaches utilise unsupervised and semi-supervised learning, but they are more resource-intensive than performance-based approaches in terms of computing capacity.

These algorithms address concept drift at its root source, which is distribution drift. They can accurately identify the time of drift and provide information about the location of the drift. However, these algorithms typically incur a higher computational cost and require users to predefine the historical time window and new data window.

A common strategy is to use two sliding windows with a fixed historical time window and a moving new data window. The first formal treatment of change detection in data streams proposed the use of total variation as a natural notion of distance between distributions. This approach provided practical guidance for designing a distance function for distribution discrepancy analysis and proposed a family of distances called Relativized Discrepancy (RD). The significance level of the

distance according to the number of data instances was presented, and bounds on the probabilities of missed detections and false alarms were theoretically proven.

Another typical density-based drift detection algorithm is the Information-Theoretic Approach (ITA). This algorithm uses a *kdqTree* to partition the historical and new data into a set of bins and uses Kullback-Leibler divergence to quantify the density difference in each bin. The algorithm applies a bootstrapping hypothesis test and confirms concept drift when the estimated probability is less than a certain significance level.

Other similar distribution-based drift detection methods include Statistical Change Detection (SCD) for multi-dimensional data, Competence Model-based Drift Detection (CM), a prototype-based classification model for evolving data streams called SyncStream, Principal Component Analysis-based Change Detection Framework (PCA-CD), Equal Density Estimation (EDE), Least Squares Density Difference-based Change Detection Test (LSDD-CDT), Incremental version of LSDD-CDT (Incremental version of Least Squares Density Difference-based Change Detection Test (LSDD-INC)), and Local Drift Degree-based Density Synchronized Drift Adaptation (LDD-DSDA). Above methods are discussed in (Feng, 2019).

2.3.3 Evolving System

The processing of data streams with concept drift can be approached in several ways, primarily through explicit and implicit methods. The explicit method involves rebuilding or retraining the machine learning model after concept drift is detected and reported.

Evolving systems are capable of processing data streams with concept drift in an online and incremental manner. A wide range of work has been done with evolving systems. For instance, the Evolving Fuzzy System (EFS) with concept fuzzy rule age and gradual forgetting mechanism has been used to handle concept drift (Lughofer and Angelov, 2011). Similar works (Andonovski et al., 2021) have utilized EFS with nonlinear Wiener-Hammerstein processes, demonstrating excellent performance in data stream situations.

The Online Bagged EFS (OB-EFS) has been proposed with two variants to tackle data streams with concept drift, performing better than the single EFS model (Lughofer et al., 2021). The negative effects of gradual drifts are autonomously compensated by the generalized fuzzy rules

with an incremental rule splitting concept, capable of performing in a fully online and single pass manner (Lughofer et al., 2018).

The Evolving Type-2 Recurrent Fuzzy Neural Network (eT2RFNN) with generalized interval type-2 rule has been developed to handle concept drift and uncertainties in large real-world data streams. This approach is especially effective in tackling high dimensional scenarios with an online feature selection technique (Pratama et al., 2017).

A new merging approach based on cluster volume for incrementally Evolving Gaussian Clustering (eGAUSS+) has been proposed for achieving more efficient data stream clustering, demonstrating a great ability to process high-dimensional data sets (Škrjanc, 2020). Gaussian related incremental learning methods on data streams have shown good performance in non-stationary environments (Leite et al., 2020).

A local model network in Takagi-Sugeno form has been used in the online identification of different processes from the data stream, showing great potential in handling significant nonlinearity (Blažič and Škrjanc, 2020). For data streams with sparse labeled samples, active learning can serve as a supportive component to assign labels to unlabeled samples (Lughofer, 2017). In (Lughofer et al., 2016), semi-supervised and fully unsupervised approaches are proposed for drift detection in classification issues of data stream mining.

Recent surveys on evolving systems and data stream learning can be found in (Leite and Gomide, 2020), which contains a wide range of system components, real-time applications, and application examples. Furthermore, (Škrjanc, Iglesias, Sanchis, Leite, Lughofer and Gomide, 2019) provides a systematic overview of evolving systems, focusing on rule-based and neuro-fuzzy based systems in machine learning on data streams. These surveys provide a comprehensive understanding of the current state of the art in evolving systems and their application in data stream mining.

In conclusion, evolving systems offer a promising approach to handle concept drift in data stream mining. They provide a flexible and adaptive framework that can adjust to changes in the data distribution, making them suitable for dealing with the dynamic nature of data streams. However, more research is needed to further improve their performance and adaptability, especially in the face of complex and high-dimensional data streams.

2.3.4 Multiple Hypothesis Test Drift Detection Method

Multiple hypothesis test drift detection algorithms employ techniques similar to those mentioned in the previous categories. The novelty of these algorithms lies in their use of multiple hypothesis tests to detect concept drift in different ways. These algorithms can be categorized into two groups: parallel multiple hypothesis tests and hierarchical multiple hypothesis tests.

Parallel Multiple Hypothesis Tests

Parallel multiple hypothesis drift detection algorithms utilize multiple hypothesis tests in parallel to detect concept drift. The Just-In-Time Adaptive Classifiers (JIT) (Alippi et al., 2013) is one of the first algorithms that set multiple drift detection hypotheses in this manner. JIT extends the CUSUM chart, known as the Computational Intelligence-based CUSUM test (CI-CUSUM), to detect changes in the mean of the features of interest by learning systems.

Other implementations of parallel multiple hypothesis tests include the Linear Four Rate drift detection (LFR) (Yu et al., 2019), which tracks changes in True Positive rate (TP), True Negative rate (TN), False Positive rate (FP), and False Negative rate (FN) in an online manner. The three-layer drift detection based on Information Value and Jaccard similarity (IV-Jac) aims to address label drift, feature space drift, and decision boundary drift individually. The Ensemble of Detectors (e-Detector) detects concept drift via an ensemble of heterogeneous drift detectors.

Hierarchical Multiple Hypothesis Tests

Hierarchical drift detection is a newer category of drift detection that employs a multiple verification schema. Algorithms in this category typically detect drift using an existing method, known as the detection layer, and then apply an additional hypothesis test, known as the validation layer, to obtain a second validation of the detected drift in a hierarchical manner.

The Hierarchical Change-Detection Tests (HCDTs) (Alippi et al., 2017) is one of the first attempts to address concept drift using a hierarchical architecture. The detection layer can be any existing drift detection method that has a low drift delay rate and low computational burden. The validation layer is activated and deactivated based on the results returned by the detection layer.

Other hierarchical drift detection algorithms include the Hierarchical Linear Four Rate (HLFR)

(Yu and Abraham, 2017), which applies the LFR (Wang and Abraham, 2015) algorithm as the detection layer, and the Two-Stage Multivariate Shift-Detection based on EWMA (TSMSD-EWMA). The Hierarchical Hypothesis Testing with Classification Uncertainty (HHT-CU) (Yu et al., 2018) and Hierarchical Hypothesis Testing with Attribute-wise "Goodness-of-fit" (HHT-AG) (Yu et al., 2018) are two drift detection algorithms based on a request and reverify strategy.

In conclusion, multiple hypothesis test drift detection algorithms provide a flexible and robust approach to detect concept drift in data streams. They offer the ability to detect drift in different ways, which can be beneficial in dealing with complex and dynamic data streams. However, more research is needed to further improve their performance and adaptability, especially in the face of high-dimensional and non-stationary data streams.

2.3.5 Deep Neural Network

To the best of our knowledge, there has been no explicit utilization of deep neural networks for concept drift detection in the existing literature. Few research works explore deep neural network learning under concept drift but not on drift detection. These are limited to discover the causes of a drift and its properties, such as start time and duration. In the paper (Korycki and Krawczyk, 2021), the author employs a reactive subspace buffer to track virtual drift induced by class drift, which may be regarded as a cluster-based method for learning under concept drift. The research paper (Liu et al., 2019) introduces a recurrent neural network to data stream learning, and incorporates the modified DDM approach to deal with concept drift, resulting in improved prediction performance.

The idea of the paper (Guo et al., 2021) is the proposed selective ensemble-based online adaptive deep neural network (SEOA), which monitors the fluctuation of the base learner to identify when a certain level of fluctuation has been reached. It demonstrates robustness and generalizability to the unstable data stream learning. In (Kauschke et al., 2019), neural network patching is applied to deep neural network to adapt non-stationary environment with concept drift. The idea is to leverage the inner layers of the previously trained network and its output to train a patch. It could adapt quickly as the concept drifts, while preserving long-term learning potential. In (Priya and Uthra, 2021), author seeks to address both class imbalance and concept drift in data streams. It

incorporates the ADWIN. Among various datasets, its precision is higher. Using meta learning theory and a prototype neural network, the method Active Drift Detection based on Meta learning (Meta-ADD) (Yu et al., 2022) is presented for concept drift identification; it can recognise many types of concept drift. The cluster or distribution based drift detection approach is incorporated with federal learning (Manias et al., 2021) system to detect concept drift. And their investigation was able to detect various levels of drift.

In (Baier et al., 2021), the author primarily employs the concept of monte carlo dropout and considers the uncertainty accompanying the deep network structure during the prediction process to be the input rather than the prediction outcome. Several studies have successfully leveraged deep learning techniques to manage environments prone to concept drift (Hamad et al., 2021) (Hu et al., 2021). The body of research associated with thermal detection can be viewed as an application of concept drift detection (Luo et al., 2019) (Gao et al., 2016).

The researchers investigate Unsupervised Open Set Domain Adaptation (UOSDA) (Fang et al., 2020) by presenting the inaugural learning bound and introducing a pioneering algorithm termed Distribution Alignment with Open Difference (DAOD), which exhibits exceptional performance on benchmark datasets. Furthermore, the scholars delve into the Probably Approximately Correct (PAC) learning theory concerning out-of-distribution (OOD) detection, ascertaining essential prerequisites for learnability and delineating learnability in realistic situations (Fang et al., 2022). The researchers also put forward conditional adversarial domain adaptation, a framework that augments domain alignment in multimodal distributions by leveraging discriminative data derived from classifier predictions.

The proposed Conditional Domain Adversarial Networks (CDANs) utilize multilinear and entropy conditioning approaches, surpassing contemporary techniques on five benchmark datasets (Long et al., 2018). These studies (Tengtrairat et al., 2022) (Zhang et al., 2022) (Hamad et al., 2022) (Alkassar et al., 2021) also demonstrate that effective management of concept drift can significantly enhance the performance of the model.

Chapter 3

Concept Drift Detection in Data Stream with Multi-label

Machine learning as a significant component of the Industrial Internet of Things (IIoT), has been widely applied in many fields. Then the continuously generated data from the various sensors are collected and stored, also known as data stream. Therefore, the detection method for concept drift is needed to alert the requirement to maintain or replace some components in advance, so as to avoid or mitigate the risk of malfunction of the IoT system. The majority of existing literature focus on concept drift detection on binary classification. To fill this gap, here we propose an algorithm to detect multi-class. Moreover to improve the performance of detection, we also introduce an algorithm which integrates the existing error rate which is widely used with our newly proposed False Positive rate. The new method is called Drift Detection Method with False Positive rate for multi-label classification (DDM-FP-M). The DDM-FP-M firstly define the false positive rate calculation method in multi-label classification, then integrates it with Drift Detection method with False Positive rate (DDM-FP). The performance of the proposed method is evaluated through Intel Lab data and is found to outperform the Drift Detection method (DDM) over 50% cases.

3.1 Introduction

With the growing advancement of IoT in recent years, gadgets are interconnected with one another to screen and control adequately and proficiently. Enormous machine conditions are collected con-

tinuously and in time, and are uploaded to the sensor administration system, in which production managers can adopt cyber-physical systems to control all operations of each machine ideally in real time (Lin et al., 2019). As the center parts, sensor systems assume a huge job in keen observing and administrations in IoT. In practice, machine components get ageing over time. If they were not replaced in time, enormous defective or low-quality products would be manufactured, and machines would perform abnormally or be damaged. Therefore, concept drift detection method in analysing such data is needed to make alert when an abnormal situation occurs (Bifet et al., 2010).

We propose a Drift Detection method with False Positive rate for multi label classification (DDM-FP-M). Firstly, we introduce the False Positive rate (FPR) for multi label classification, where the FPR is extended to broader application area. Then we integrate FPR into DDM-FP method to detect a drift and generate the warning and drift level in DDM-FP-M. Finally, to ascertain if the warning and drift thresholds have been met, we utilize a trained classifier integrated with the DDM-FP-M method for effective monitoring. Results indicate that our method not only successfully detects concept drift, but also tends to detect it earlier than the traditional DDM.

The main contribution of this chapter are the extend of False positive rate for drift detection, and the new method for concept drift detection for multi label classification. Meanwhile, the DDM-FP-M can usually detect the concept drift ahead of the widely used method, DDM. In this way, we may handle and react to the concept drift in time, and improve the performance of the data stream mining with concept drift.

3.2 Drift Detection Method (DDM)

The DDM is a classical method to detect drifts in data stream mining (Gama et al., 2004). It is based on the statistical theories of Probably Approximately Correct learning and prequential probability (Dawid and Vovk, 1999). When the class distribution of samples remains stationary, the error rate will decrease. Given the confidence level for warning of detecting drift to be 95%, the warning will be issued when the following condition is met:

$$p_i + s_i \geq p_{min} + 2 \times s_{min} \tag{3.1}$$

where i is the i^{th} new sample in the testing data set, p_i is the probability of misclassification, s_i is the standard deviation given by $s_i = \sqrt{p_i(1-p_i)/i}$, p_{min} is the minimum probability found during training, and s_{min} is the minimum standard deviation found during training.

$$p_i + s_i \geq p_{min} + 3 \times s_{min} \tag{3.2}$$

Similarly, Equation 3.1 indicates the warning level, which the confidence level is set to 95%. When the Equation 3.2 is satisfied the drift level is reached and drift is detected.

3.3 Drift Detection Method with False Positive Rate for multi-label Classification (DDM-FP-M)

In this section, we propose DDM-FP-M method for concept drift detection in multi label classification. We address the issue of concept drift detection in multi-label classification. To clarify our work, it should be noted that in our approach, each output has only one label, but this label can belong to a category that includes two or more classes. This is distinct from scenarios where there could be more than one output label per instance.

3.3.1 False Positive Rate Calculation for Multi-label Classification

In binary classification, the confusion matrix is a 2×2 matrix , and True Positive (TP), False Negative (FN), False Positive (FP) and True Negative (TN) can be got easily, so the False Positive Rate (FPR) value is equal to $FPR = FP/(FP + TN)$ (Flach, 2012).

The False Positive Rate (FPR), a crucial parameter traditionally conceived for binary classification problems, essentially quantifies the proportion of false alarms. However, this measure encounters an inherent limitation when we extrapolate its application to multi-label classification paradigms. In these intricate scenarios, a global FPR value is not directly computable, but we are confined to calculating individual class-specific FPR values.

According to Figure 3.1, there are $N(N > 2)$ classes in total, the confusion matrix of multi label classification consists of the actual and predict values of each class.

We take class A_1 as a sample, the FPR value of class A_1 equal to:

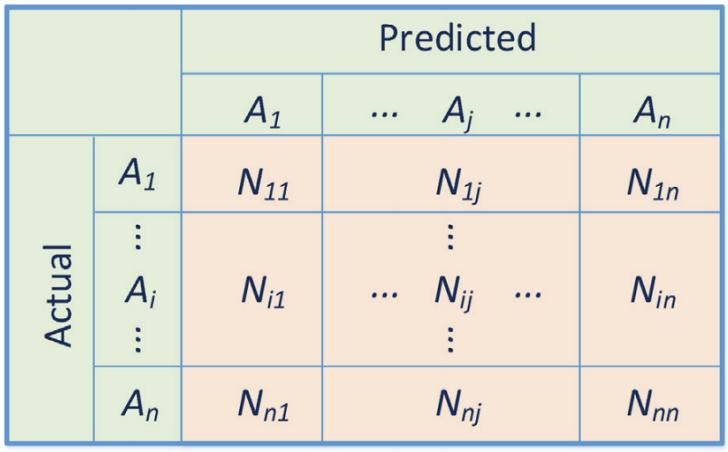


Figure 3.1: The confusion matrix for multi label classification.

$$FPR_{A_1} = \frac{FP_{A_1}}{FP_{A_1} + TN_{A_1}} \tag{3.3}$$

3.3.2 DDM-FP-M

In the pursuit of improving the performance of concept drift detection in multi-label classification, we introduce a novel method called DDM-FP-M. This technique uniquely blends both error rate and false positive rate (FPR) to detect concept drift with higher precision. Below, we elucidate the mathematical formulation and the strategic process underpinning this method.

Step 1: The first crucial step in DDM-FP-M is to calculate the false positive rate for multi-label classification, tailored to incorporate the complexity and specificities of multi-label scenarios.

Based on Equation 3.3, For FP and TN value of class A_1, the calculation is:

$$FP_{A_1} = \sum_{i=1}^{n} N_{i1} - N_{11} \tag{3.4}$$

Equation 3.4 indicates how many instances of FP of class A_1 that do not belong to A_1 but are predicted to be A_1.

$$TN_{A_1} = \sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij} - \sum_{i=1}^{n} N_{i1} - \sum_{j=1}^{n} N_{1j} + N_{11} \tag{3.5}$$

Equation 3.5 indicates how many instances of TN of class A_1 that do not belong to A_1 but are not predicted to be A_1.

Step 2: The FPR for class A_1, as expressed in Equation 3.6, is derived from the FP and TN values:

$$FPR_{A_1} = \frac{\sum_{i=1}^{n} N_{i1} - N_{11}}{\sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij} - \sum_{j=1}^{n} N_{1j}} \tag{3.6}$$

Step 3: For incorporating the FPR into the DDM-FP-M algorithm, we employ the arithmetic mean of all FPR values, denoted as $FPR_{\bar{v}}$, and formulated in Equation 3.7:

$$\begin{aligned} FPR_{\bar{v}} &= \frac{\sum_{k=1}^{n} FP_{A_k}}{n} \\ &= \sum_{k=1}^{n} \frac{\sum_{i=1}^{n} N_{ik} - N_{kk}}{\sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij} - \sum_{j=1}^{n} N_{kj}} \times \frac{1}{n} \end{aligned} \tag{3.7}$$

Step 4: Upon integrating the calculated FPR into the Drift Detection Method, we instantiate the DDM-FP-M technique. The innovative inclusion of $FPR_{\bar{v}}$ as per Equation 3.7 offers a sophisticated refinement to the conventional DDM approach.

When the instance i is coming by sequence:

$$p_i + fpr_{\bar{v}-i} \leq p_{min} + fpr_{\bar{v}-min} \tag{3.8}$$

Equation 3.8 is the update mechanism of p_{min} and $fpr_{\bar{v}-min}$, when equation is satisfied, $p_{min} = p_i$, $fpr_{\bar{v}-min} = fpr_{\bar{v}-i}$.

$$p_i + fpr_{\bar{v}-i} \geq p_{min} + 2 \times fpr_{\bar{v}-min} \tag{3.9}$$

$$p_i + fpr_{\bar{v}-i} \geq p_{min} + 3 \times fpr_{\bar{v}-min} \tag{3.10}$$

Drawing a parallel to the conventional DDM methodology, DDM-FP-M encompasses these dual thresholds. Here, p_i symbolizes the misclassification probability. Diverging from the traditional

DDM, in our advanced model, the standard deviation of p_i is replaced by $FPR_{\bar{v}}$. Thus, Equation 3.9 delineates the warning threshold, whereas Equation 3.10 demarcates the drift threshold.

From Algorithm 1, the DDM-FP-M is similar to DDM method, which has two thresholds, warning and drift respectively. The establishment of these two thresholds is based on the 3 sigma rule. We use two standard deviations to represent the warning threshold, indicating that there is only a 5% probability that the drift reaches this warning level. The drift threshold is set higher; we define the occurrence of drift using three standard deviations, which corresponds to 1% probability. Additionally, for the error rate, we continue the approach used in DDM, representing it through the cumulative error rate.

Algorithm 1 DDM-FP-M

Require: Current error rate, er; Current false positive rate average value, $fpr_{\bar{v}}$; Current minimum of error rate, er_{min}; Current minimum of false positive rate average value ,$fpr_{\bar{v}-min}$;
Ensure: warning threshold *or* drift threshold;

minimum of error rate initialised as error rate
minimum of false positive rate initialised as false positive rate
$Warning\_flag = False$
$Drift\_flag = False$
$concept\_status = None$
if $er + fpr_{\bar{v}} \leq er_m + fpr_{\bar{v}-min}$ **then**
    $er_{min} \leftarrow er$, $fpr_{\bar{v}-min} \leftarrow fpr_{\bar{v}}$
end if
if $er + fpr_{\bar{v}} \geq er_{min} + 2 \times fpr_{\bar{v}-min}$ **then**
    $Warning\_flag = True$
else
    $Warning\_flag = False$
end if
if $er + fpr_{\bar{v}} \geq er_{min} + 3 \times fpr_{\bar{v}-min}$ **then**
    $Drift\_flag = True$
else
    $Drift\_flag = False$
end if
if $Warning\_flag == True$ and $Drift\_flag == False$ **then**
    $concept\_status = warning\ threshold$
end if
if $Warning\_flag == True$ and $Drift\_flag == True$ **then**
    $concept\_status = drift\ threshold$
end if
return $concept\_status$;

3.4 Experiment and Result Analysis

This section primarily focuses on the illustration and evaluation of the newly devised methodology, DDM-FP-M, within the context of real-world data streams pertinent to an IoT monitoring platform. This platform serves as an ideal example to emphasize the applicability and effectiveness of our proposed method, given the inherent dynamism and non-stationarity in IoT data streams.

3.4.1 Datasets and Experiment Setting

The dataset for this research study is sourced from the TinyDB, an in-network query processing system built on the robust TinyOS platform. The arrangement of the sensors utilized in this experiment within the Intel Berkeley Research lab is depicted in Figure 3.2. A total of 54 sensors were deployed, collecting timestamped topological data entailing attributes such as humidity, temperature, light, and voltage, at an interval of 31 seconds. The data collection phase spanned over 37 days.

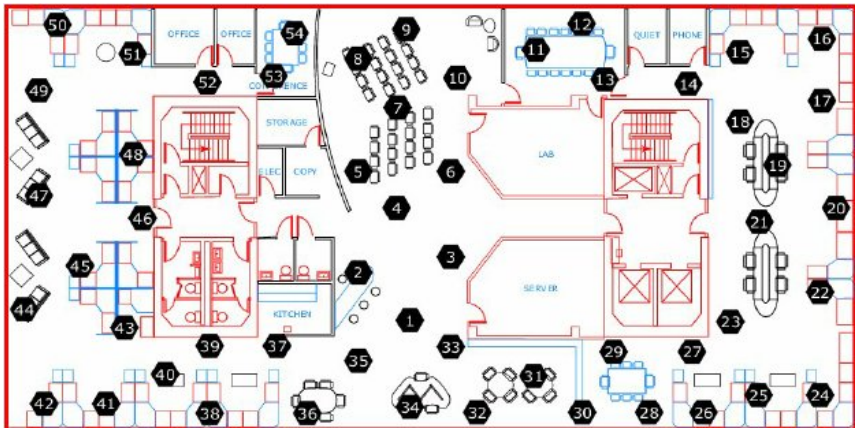


Figure 3.2: The arrangement of sensors in the Intel Berkeley Research lab(Xu et al., 2019)

Each data instance in this dataset comprises four attributes - temperature, humidity, lighting, and velocity. Furthermore, a class attribute segregates these instances into four distinct classes (Liu, Feng, Wu, Hou and Han, 2017). For our experimental framework, we selected the initial 1,000,000 samples from the dataset and subdivided them into 100 groups, each containing 10,000 instances.

To facilitate a comprehensive comparison, we designed two sets of experiments on this split dataset. The DDM and our proposed DDM-FP-M methods were applied to these two dataset

groups, thus enabling us to demonstrate the performance of the newly conceived DDM-FP-M in relation to the traditional DDM. This comparative approach allows us to showcase the enhancements brought forth by DDM-FP-M and its potential applicability in complex, multi-label classification tasks in dynamic environments such as IoT data streams.

3.4.2 Experiment Result Analysis

The summary of the result from two groups is shown in Table 3.1. There are data structure errors in instance 950000 and 980000 when take the second group test, so there are 198 groups of result in total.

Table 3.1: The result from two group of experiments

	Instance	***early than DDM***	***DDM fail***	***fail***	***Both fail***	***Roughly equal***
overall	198	102	26	26	17	35
first group	101	52	19	20	13	18
second group	97	50	7	6	4	17

Table 3.2: The further analysis on experiment result

	earlier than DDM	***DDM fail***	***fail***	***Both fail***	***roughly equal***
overall	51.51%	13.13%	13.13%	8.59%	17.68%
first group	51.49%	18.81%	19.80%	12.87%	17.82%
second group	51.55%	7.72%	6.19%	4.12%	17.52%

Table 3.2 is the further analysis based on Table 3.1, except from the column *Instance*. The column *DDM-FP-M early than DDM* indicates the perception of DDM-FP-M method detect drift early than DDM in overall scope, so as to rest of columns in Table 3.2.

As the Table 3.2 shows, the results in the columns *DDM fail* and *DDM-FP-M fail* indicate a prediction failure rate of 13.13% for both methods. It suggests both methods have the same ability to detect the drift in the IoT data stream. Specifically, the DDM method perform slightly better than DDM-FP-M in first group test. In contrast, the DDM-FP-M method performed better in the second group test.

On the other hand, there is a column *both fail* in Table 3.2, which has two means, one is there is no concept drift. Another is that the concept drift in this period belongs to gradual concept drift and the drift amplitude is so slight that detection method cannot detect it. Meanwhile, the failure rates from both groups show the number of training instance has significant influence on

the performance of prediction.

From the analysis of detection fail rate, we infer that DDM-FP-M and DDM method both can effectively detect the concept drift and have some similar performance.

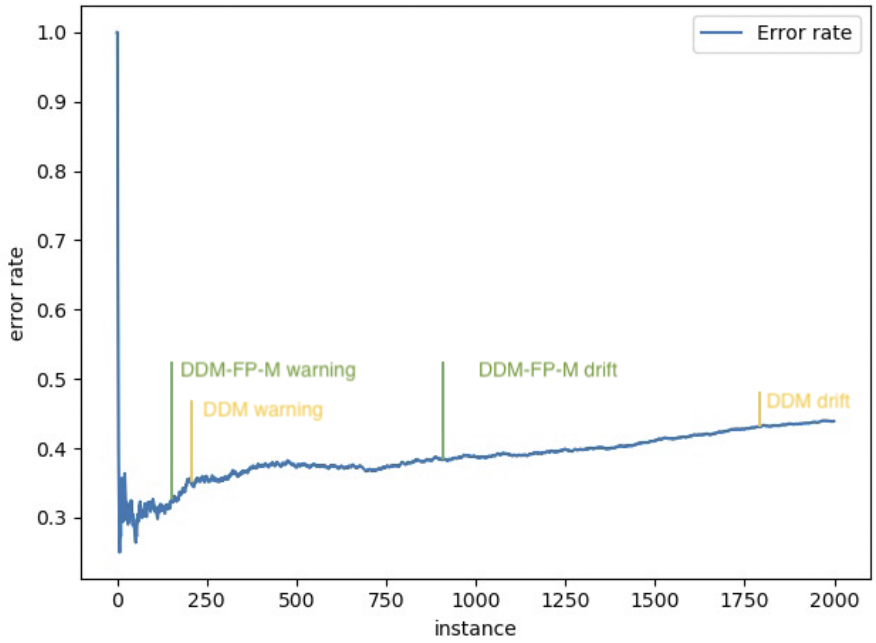


Figure 3.3: The drift detection result from DDM and DDM-FP-M method

For the comparison of detection of DDM and DDM-FP-M, we could find columns *DDM-FP-M earlier than DDM* in Table 3.2, the first column, which means the DDM-FM-M can detect the concept drift earlier than DDM method. The result from overall or any single group are both equal to around 51.5%, which proves the consistent better performance of proposed DDM-FP-M method.

For example, in Figure 3.3. At the instance of 159, the error rate is 33.2% and the FPR is 11.3%, warning level reached because it reaches the *warning threshold* according to Equation 3.9 of DDM-FP-M, however, the DDM found the warning threshold until time of 195. For the finding of the drift threshold, DDM-FP-M first detect the drift at instance of 955 based on Equation 3.10. However, the DDM detect the drift until instance of 1866, which is far later than DDM-FP-M.

Besides, last column in Table 3.2 called *roughly equal*, which means the result from the DDM-FP-M is just slightly later than DDM method (we define the tolerance interval is 20% based on

the result from DDM method). For instance, in the experiment [start: 990000, training: 500, test: 4000], the instance point that DDM detect drift is 925, the DDM-FP-M is at instance point 946, DDM just early than DDM-FP-M for 21 instances, which maintain the tolerance interval. These instance accounts for 17.68% of all instance.

Combining the two findings that DDM-FP-M early than DMM and roughly equal, we can find out that around 70% (*DDM-FP-M early than DDM*: 51.51% + *roughly equal*: 17.68%) probability that the performance of the DDM-FP-M is superior to or similar to the DDM algorithm.

Based on the experiment result on data from Intel Lab, the DDM-FP-M and DDM method can applied to concept drift detection. However, considering the detection performance, the DDM-FP-M can always detect concept drifts faster than the DDM method.

3.5 Summary

With the development of the IoT, deployment of the massive number of sensors in many practical applications could continuously collect environmental data. Hence, the way that effectively and efficiently processing within reality the data streaming methods are dealing with incoming streams, not pre-collected data. Here we executed the methods as if the data were generating while running the algorithms. The incoming streams have the same sequence and value that has been collected for research in literature and available in the public domain. Data stream be considered a significant role in the development of the IoT. However, the concept drift happened in the data stream challenges data processing, control and system management. Especially in healthcare data area, which has quite rich data and usually contains human survival-related information. In this chapter, we propose a novel concept drift detection method based on DDM, Drift Detection Method. The proposed method can effectively detect the concept drift by monitoring the statistic indicator of the learner trained on the data stream, but also can detect the concept drift earlier than the DDM method through the combination of the accuracy and FPR of the trained learner. By calculating and observing the change of the two error rates and FPR, to determine if the warning and drift level is reached. Experimental result in Intel Lab data shows that our method can effectively detect the concept drift and the time that drift detected is earlier than the DDM method. Therefore, our method has better performance than the DDM method on concept drift detection.

Chapter 4

Concept Drift Detection in Noisy Data Stream

Drift detection methods identify changes in data streams. Such changes are called concept drifts. Existing drift detection methods often assume that the input is a noise-free data stream. However, in real world applications, for example, data streams generating from internet of things are normally contaminated with noise. (noise, i.e. class noise and/or attribute noise). In this work, we propose a Noise Tolerant Drift Detection Method (NTDDM), which is based on two-step detection and validation function to detect drifts, and filters out the false drifts caused by the noise. The NTDDM is compared with six well-known drift detection methods and tested on four benchmarks having different levels. Three performance indicators are proposed to determine whether the drift detection is made within a reasonable time, and the length of time to the known drift starting point. The comparative studies demonstrate that NTDDM outperforms the existing methods, over these performance indicators. Our proposed method has achieved a statistically significant improvement on drift detection compared to the methods in experiment. The proposed NTDDM makes it possible to efficiently and effectively detect drift in a noisy data stream.

4.1 Introduction

Concept drift detection methods aim to identify if and when data patterns within the stream alter (Gama et al., 2004), (Baena-Garcıa et al., 2006), (Bifet and Gavalda, 2007), necessitating con-

tinual updating of detection models. A data stream originating from a stationary environment, characterized by a fixed probability distribution, is void of drift (Ditzler et al., 2015).

Concept drift mitigation primarily leverages two strategies, active and passive (Gama et al., 2014). Active strategies monitor the data stream, issuing a warning upon drift detection (Lu et al., 2019), while passive strategies do not explicitly raise warnings. This research focuses on the active strategy.

Consider a data stream consisting of n samples, represented as $H_{1,n} = [h_1, \cdots, h_n]$, where each sample $h_i = (X_i, y_i)$ is a data point comprising a d-dimensional feature vector X_i and a class label y_i. Typically, the data stream follows a joint distribution $D_{1,t}(X, y)$. Concept drift is said to occur at time point $t + 1$ when $D_{1,t}(X, y) \neq D_{t+1,\infty}(X, y)$ (Lu et al., 2016), (Liu et al., 2020).

Concept drift can be categorized into *real* and *virtual* drift based on Bayesian theory (Souza et al., 2020). *Real* drift arises when $P_t(y|X) \neq P_{t+1}(y|X)$ while $P_t(X) = P_{t+1}(X)$, indicating a change in conditional probability. Conversely, *virtual* drift transpires when $P_t(X) \neq P_{t+1}(X)$ while $P_t(y|X) = P_{t+1}(y|X)$, marking a shift in marginal probability. Our research focuses on real concept drift, as it directly impacts the decision boundary (Lu et al., 2019), (Gama et al., 2014), (Wang et al., 2018).

On the other hand, noise can significantly hinder the performance of data mining models (Pang et al., 2021). Noise has an adverse impact on the both interpretation of the data, and reduction in the performance of the machine learning model (Mahan et al., 2021). However, the majority existing drift detection methods assume no noise. This introduces new challenges to the data stream mining model (Lu et al., 2019), which this work addresses.

Clearly, drift detection becomes more complex when noise is present in the data stream. Noise will mask the true signal and degrade the performance of machine learning models. This may mistakenly trigger a drift detection, miss an antual drift, or delay the drift detection time. However, few efforts have been made to detect drifts while noise exists in data stream. We address this issue with a noise-tolerant detection method. Our contributions are:

1. Incorporating two types of noise into the data stream with concept drift, so that data contained concept drift and one of two kinds of noise.

2. Presenting a noise tolerant drift detection method to detect the drift in noisy data streams.
3. Proposing sparse sliding window strategy that could mainly reduce the false drift detection, where the statistical test is used to compare the differences between the data stream before and after the drift happens.
4. Proposing three measures for drift detection, to evaluate the performance of detection accuracy and timeliness.

The rest of section is organized as follows: Section 4.2 discusses the problem statement and the definition of the two kinds of noise. The proposed method is presented in Section 4.3. Experiments and results are discussed in Section 4.4. Finally, in Section 4.5, the summary is provided.

4.2 Problem Statement

4.2.1 Introduction of Noise

Noise is an unavoidable problem in real-world data. Real-world data, which forms part of the training data, suffers from noise. Noise could reduce the performance of the system, e.g. decrease in accuracy, or a delay in decision-making. The majority of the drift detection methods assume the data stream is noise free. Data acquisition and collection inevitably contain noise.

4.2.2 Definition of Noise

Noise is divided into two categories, class noise and attribute noise (Luengo et al., 2021). A data sample could be represented by $h_i = (X_i, y_i)$, where X_i is the set of attributes and y_i is the label at time point *i*. Noisy data could be part of a collection of n data samples $(h_1, \cdots, h_n)$. For the noisy data, the noise could be on attributes or label, even on both meanwhile. In order to introduce two types of noise precisely, we only consider independent noise.

Here we define $X_s = (X_{s_1}, X_{s_2})$ as a set of data samples X, where X_{s_1} and X_{s_2} represent the subset of data samples where the corresponding labels are y_1 and y_2, respectively. We also define the label mapping function $M(\cdot)$ as $y_i = M(X_{s_i})$. From Figure 4.1, there are two sub-figures which have two data samples collections, i.e. X_{s_1} square and X_{s_2} triangle, corresponding to two different classes.

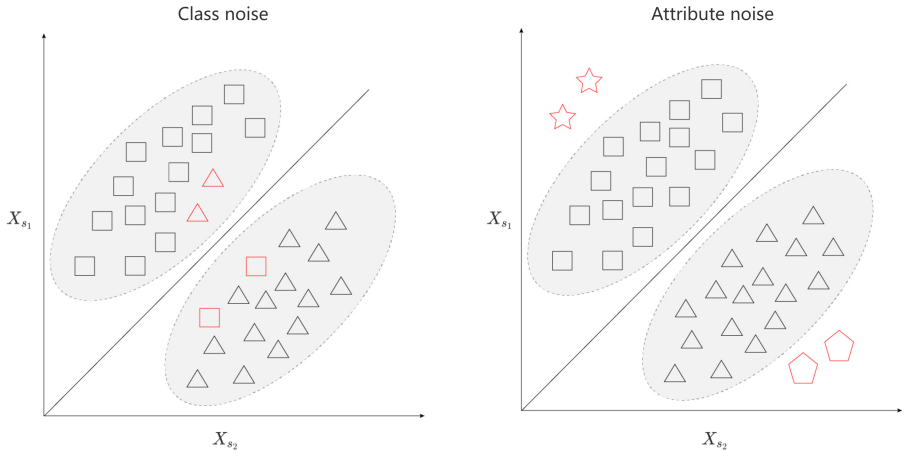


Figure 4.1: Visualization of class noise (left) and attribute noise (right), each shadowed ellipse represents a class X_{s_1} and X_{s_2}.

However, there are also some abnormal data samples in Figure 4.1 (left) where have similar attribute distribution but has been mislabelled. This situation corresponds to class noise. Class noise alters the observed labels assigned to instances, e.g. misclassifications, assigned with false label, shown on Equation4.1:

$$M(X) \neq M(X_{s_i}), X \in X_{s_i} \tag{4.1}$$

On the other hand, attribute noise is the case that data sample with label which is significantly different from corresponding X_s, e.g. data samples colored by red in Figure 4.1 (right). Attribute noise affects the observed values of the attribute, e.g. erroneous attribute values, missing attribute values, or redundant attribute values, etc. As shown in Equation 4.2, data with attribute noise has the same class label as the adjacent class, but it does not belong to its current class since it is significantly different from the current distribution:

$$M(X) = M(X_{s_i}), X \notin X_{s_i} \tag{4.2}$$

4.2.3 Difference between Noise and Concept Drift

The notions of noise and concept drift both play significant roles in the field of machine learning and predictive modeling. While they share similarities, particularly in their potential to reduce the accuracy of predictive models, they fundamentally differ in their underlying causes and effects.

Here, we delve into their distinctions and implications.

Nature of Changes:

Concept Drift: This phenomenon occurs when there is a shift in the underlying distribution of the data over time, leading to a change in the decision boundary. Such changes impact the relationship between the input features and the target variable. Concept drift can be mathematically represented as $P(y_{t+1}|X_{t+1}) \neq P(y_t|X_t)$, given that $P(X_{t+1}) = P(X_t)$. This expression encapsulates the drift in the conditional probability of the target variable y given the input features X.

Noise: Unlike concept drift, noise does not represent a systematic change in the data distribution. Instead, it refers to random variations or errors that can occur in the data collection, transmission, or processing stages. Noise does not lead to alterations in the decision boundary, and it does not follow a specific pattern.

Temporal Aspect:

Concept Drift: It is characterized by a specific drift point, after which the alterations occur. The changes can be gradual, abrupt, or incremental, depending on the underlying dynamics of the environment. The detection and adaptation to concept drift require specialized techniques to maintain the model's predictive performance.

Noise: Noise, on the other hand, is typically persistent and randomly distributed throughout the data. It lacks a specific temporal pattern or point of change and is an inherent challenge in various data-intensive applications.

Impact on Modeling:

Concept Drift: The presence of concept drift necessitates continual monitoring and adaptation of the model. Failure to recognize and respond to drift can lead to a significant degradation in the predictive accuracy and reliability of the model.

Noise: Noise contributes to a decrease in model accuracy but does not necessitate continual adaptation in the same way as concept drift. Rather, techniques for noise reduction and robust modeling can mitigate its impact.

In summary, while both noise and concept drift pose challenges to predictive modeling, they differ in their underlying nature, impact, and the strategies needed to mitigate their effects. Understand-

ing these differences is vital for the design, deployment, and maintenance of effective and robust predictive models in dynamic and noisy environments.

4.3 Proposed Noise Tolerant Drift Detection Method

We firstly give the notation to be used in this section in Table 4.1.

Table 4.1: Notation and abbreviations

Symbol	Description
$H_{1,n}$	data stream $H_{1,n} = [h_1, \cdots, h_n]$ starts at time point 1 and end at n
h_i	single data sample at time point *i* in H, $h_i = (X_i, y_i)$
$D_{i,t}$	distribution of the data stream from time point *i* to *t*
σ	standard deviation
p	probability
w	time window
t_s	the starting time of data stream
t_g	the starting time of drift
t_e	effective detected drift point which falls into EDR
t_f	first effective detected point
t_l	last detected point
S_i	i-th independent experiment
O_i	detection output, $O_i = [o_1, o_2, ..., o_n]$ in one trial
l_t	time from start point t_s to the starting time of drift t_g
γ	threshold of the EDR
EDR	effective detected range, $[L_t, L_t * (1 + \gamma)]$
$FEDP$	time span between t_g and t_f
LDP	distance between t_g and t_l

We propose a NTDDM to address the problem of data with noise for drift detection. NTDDM is

designed to have two steps: detection and validation. The first step is aimed to detect the potential drift since the drift could manifest as the change of the performance of the trained machine learning model. Hence, we use precision as the indicator to monitor this change. Precision is more sensitive and meaningful to locate the drift compared to other comprehensive indicators, e.g., accuracy (used by DDM), F1-score and AUC, etc (Klinkenberg et al., 1998) (Halstead et al., 2021). The second step is designed to filter out the detected potential drift points which are impacted by the noise. We apply the modified KS two-sample test on potential drift points after subsampling. The subsample used here is to further mitigate the impact of noise. Lastly, the output of the proposed method is the filtered detected concept drift and the experiments show that our method has effectively detected the drift and is more robust to noise compared to other methods. Section 4.3.1 provides the overview of NTDDM method. Section 4.3.2 proposes Detection to detect potential drifts, and Section 4.3.3 proposes Validation to validate if the potential drift is true.

4.3.1 Design of NTDDM

The design of the proposed NTDDM is shown in Figure 4.2, and Algorithm 2. Here the Detection (line 2-6) is executed at real time and continuously monitor the performance of classifier for input data. When the change reaches a predefined threshold, a potential drift is reported.

Validation (line 8-15) will be triggered once a potential drift is detected in the detection stage. Validation determines if the detected drift is real or not. After completing the Validation, the NTDDM returns to the detection to continuously monitor and detect the potential drift (line 17), or raise a real drift flag and then restart the detection (line 12). The outputs of NTDDM are the drift starting time T.

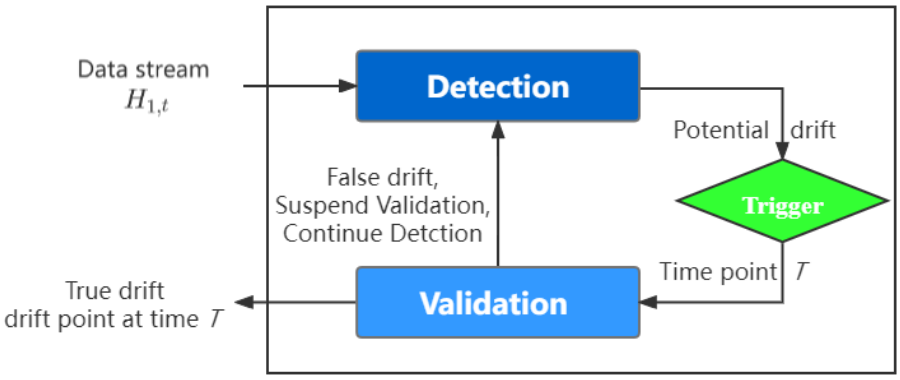


Figure 4.2: Design of NTDDM, two process modules are Detection and Validation, data stream input in Detection firstly, potential drift found in Detection will trigger the Validation. True drift will be reported if Validation confirms it, otherwise suspend Validation and return to Detection since the potential drift is a false drift.

Algorithm 2 Proposed NTDDM

Require: data stream $H_{1,t}$ where $i \to +\infty$, incremental trained classifier f
Ensure: Confirmed drift time point T

for i in $[0, +\infty)$ **do**
    Perform *Detection* (detail in Algorithm 3)
    if drift threshold reached at T **then**
        Potential drift is detected by *Detection*
        Current time point T store in buffer, used as drift point when potential drift is validated.
        Validation is triggered
    end if
    Perform *Validation* (detailed in Section 4.3.3)
    Continue to collect windows-size data
    Calculate dissimilarity between two sub-windows (Figure4.3)
    if Current potential drift is real drift **then**
        Report real drift and corresponding time point
    else
        Current potential drift is false drift
        Validation suspend and waiting to be triggered
    end if
    Continue *Detection*
end for

4.3.2 Detection

Algorithm 3 shows the proposed detection function. Line 1 initializes the boolean $Drift\_flag$ with False, representing no potential drift detected. Line 2-4 initializes the value of p, σ, p_{max} and σ_{max}. Line 6-7 then defines the updating rules of the p_{max} and σ_{max}, values of them will be updated and saved on two registers. Line 10-11 describes the trigger for $Drift\_flag$, the

$Drift\_flag$ will become true when condition in line 10 is satisfied. Line 13-15 explains how the validation is triggered after $Drift\_flag$ becomes true. Meanwhile, the current time point T is recorded for the use of validation.

Algorithm 3 Detection steps

Require: Current precision value p_i and its deviation σ_i, Current maximum of precision value p_{max} and corresponding deviation σ_{max}, time point *T*
Ensure: potential drift time point

1: $Drift\_flag = False$
2: **if** p_{max} and σ_{max} equal to None **then**
3: $p_{max} \leftarrow p_i$
4: $\sigma_{max} \leftarrow \sigma_i$
5: **else**
6: **if** $p_i - \sigma_i > p_{max} - \sigma_{max}$ **then**
7: $p_{max}, \sigma_{max} = p_i, \sigma_i$
8: **end if**
9: **end if**
10: **if** $p_i - \sigma_i < p_{max} - 2 \times \sigma_{max}$ **then**
11: $Drift\_flag = True$
12: **end if**
13: **if** $Drift\_flag = True$ **then**
14: activate the validation
15: save current time point *T* for validation
16: **end if**

In above p_i is the precision which is defined in Equation 4.3, and measures the quality of the classifications (Jain et al., 2020). Here we introduce precision as the indicator of drift detection. Similar to accuracy of classifier, precision value will change accordingly when drift occurs. In this work, the precision values are calculated incrementally, applicable to both binary and multi-class(Virtanen et al., 2020).

$$precision = \frac{TP}{TP + FP} \tag{4.3}$$

In Equation 4.3 True Positive (TP) represents an outcome where the model correctly predicts the positive class, and False Positive (FP) represents an outcome where the model incorrectly predicts the positive class. Precision means the proportion of positive identifications was actually correct predicted. It remains stable until the drift occurs (Yu et al., 2019). Since in a stationary environment, the predicted result will maintain a relatively stable high accuracy. It can be deducted that the TP value will increase and the FP value will remain at a relative small value. As the noise

increases, the precision falls. Moreover, in our experiments, through testing different indicators, we found that precision is the most sensitive measure for detecting concept drift. The value of precision p_i corresponding to the standard deviation as $\sigma_i = sqrt(p_i(1 - p_i))$.

When the performance of the learned classifier is unchanged for the new data, i.e. there is no drift, then the term $p_i - 2 \times \sigma_i$ should be always lower than $p_{max} - \sigma_{max}$. The weighting parameter α depends on the confidence level. The drift detection method manages two registers during the training of the machine learning model, p_{max} and s_{max}. Every time a new example (X_i, y_i) is processed, those values are updated when $p_i - \sigma_i$ is larger than $p_{max} - \sigma_{max}$, corresponding to line 6-7 in Algorithm 2.

4.3.3 Validation

The detection is used to detect the potential drifts in the noisy data stream, which is then followed by the Validation to validate if the potential drifts are real or not. In practice, some potential drifts are caused by noise and not due to a change in the data stream distribution. To filter out the potential drifts, a statistical test is employed in the validation function. Validation takes incrementally generated precision values as input. There are several approaches that can be used to determine whether two sets of data input from the same distribution. Here we choose Kolmogorov–Smirnov two-sample test (KS test) (Jr., 1951). It is a non-parametric test and makes no assumption about the distribution of data. In addition, its relaxed restriction on the sample size is an advantage for fast drift detection. These properties fit the purpose of the proposed validation function.

Kolmogorov–Smirnov (KS) two sample test

Suppose there are two sets of one-dimensional data D_m and D_n, where each set contains the same number of samples, i.e. $m = n$. We aim to compare the difference between two sets of data. There are multiple ways to achieve this goal, for example KS test and Kullback-Leibler (KL) divergence Shaker and Lughofer (2014). We selected the Kolmogorov-Smirnov two-sample test due to its lightweight, non-parametric nature, ideal for one-dimensional data analysis. However, the KL divergence will be preferred in multi-dimensional situation. The KS test is able to reject the null hypothesis that D_m and D_n are generated from the same distribution. The KS test requires the data to be independent and identically distributed. The null hypothesis is verified if the following

condition is satisfied:

$$D_{n,m} > c(\alpha)\sqrt{\frac{|D_n| + |D_m|}{|D_n||D_m\ |}} \tag{4.4}$$

where the value of $c(\alpha)$ is obtained from Statistical Table Wei et al. (2020), which is a pre-calculated table. Here $|D_n|$ is the number of data in D_n and $|D_m|$ is the number of data in D_m. $D_{n,m}$ is the KS statistic value, and is defined as follows:

$$D_{n,m} = \max_x |F_{1,n}(x) - F_{2,m}(x)| \tag{4.5}$$

where max represents the maximum value of the set of distances between $F_{1,n}$ and $F_{2,m}$, and $F_{1,n}$ is the empirical distribution function, defined as below:

$$F_{1,n}(x) = \frac{1}{n}\sum_{i=1}^{n} I_{[-\infty,x]}(X_i) \tag{4.6}$$

where $I_{[-\infty,x]}(X_i)$ is indicator function, defined as follows. $F_{2,m}$ is defined similarly in Equation 4.6.

$$I_{[-\infty,x]}(Xi) = \begin{cases} 1 & X_i \leqslant x \\ 0 & X_i > x \end{cases} \tag{4.7}$$

For the experiments in Section 4.4, we set α to 0.001, and $|D_1|$ and $|D_2|$ are set to k in NTDDM, KS test is able to compare the difference between two distribution based on cumulative distribution function. Therefore, the null hypothesis will be rejected when the $p - value < 0.001$.

Proposed new spare sliding time window strategy

The validation examines the data within a sliding time window. We use the setup of the experiments in Section 4.4 as an example. To perform this function, we have a fixed size ($2w$) sliding window to capture two sets of consecutive data before and after a potential drift, as shown in Figure 4.3. In this example, the length of the time window is $2w$. Here the detection function detects a potential drift at time T.

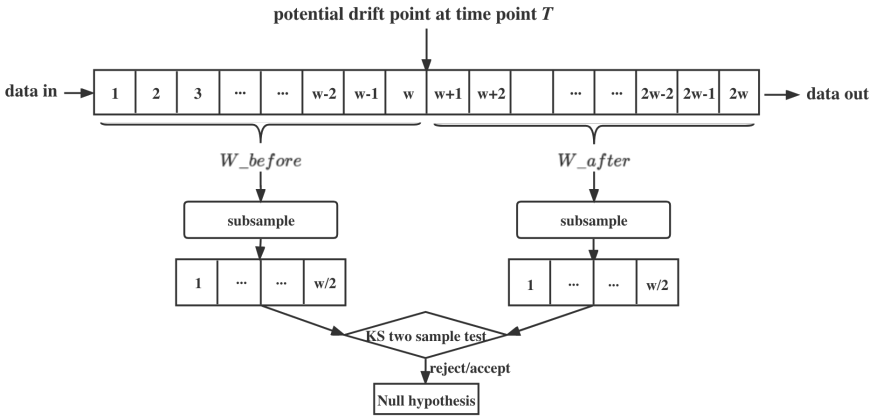


Figure 4.3: Validation with spare sliding time window.

The sliding window will continue to collect additional data, the subsampling is applied on each sub-window. Then the dissimilarity between two sub-windows will be compared. Lastly, the real drift will be confirmed if the difference is statistically significant. The additional $w/2$ data, although it will introduce a delay time to validate a drift, it affords a lower detection false positive.

Specifically, we do not take all the data collected by the sliding window. The reason is that a slight change in a short period of time will cause the validation function to generate false $Drift\_flag$. To avoid this, we use a strategy that randomly selects $w/2$ instances from each of the sub-windows. This not only satisfies the requirement of continuous data, but also reduces the chances of a false $Drift\_flag$. The datasets collected by the sliding window correspond to the data before and after the detection of the potential drifts respectively. If the data stream is stationary, the $p - value$ should be larger than 0.001.

4.4 Experiment and Result Analysis

This section compares the proposed NTDDM with the six existing drift detection methods, over four benchmark datasets and one real-world dataset. These datasets have different noise levels, which could be adjusted via noise setting parameters. Section 4.4.1 describes the four benchmark datasets. Section 4.4.2 introduces the summary of the measures used to evaluate the performance of NTDDM and analyze the experimental results, comparing the seven drift detection methods. The real-world dataset experiments and evaluation are introduced in Section 4.4.3.

4.4.1 Datasets and Experiment Set-up

Four benchmark datasets widely used in drift detection literature are selected for experiment and evaluation, namely SEA, SINE, AGRAWAL, and LED. They are available at the public stream data mining platform *skmultiflow* (Montiel et al., 2018). Descriptions of the datasets are in Table 4.2. Each dataset in Table 4.2 consists of 1000 data points, corresponding to 1000 time points. At the 500th time point, a drift occurs. For each of these datasets, 100 distinct versions are generated . To compare the performance of drift detection methods, we test them on each dataset for 100 trials, which are initialized with different random seeds.

Table 4.2: Description of the datasets.

Dataset	No. of attributes	No. of classes	Level of noise	Type of noise
Sea_0	3	2	0	class noise
Sea_0.05	3	2	0.05	class noise
Sea_0.1	3	2	0.1	class noise
Sea_0.15	3	2	0.15	class noise
Sea_0.2	3	2	0.2	class noise
Agrawal_0	9	2	0	class noise
Agrawal_0.05	9	2	0.05	class noise
Agrawal_0.1	9	2	0.1	class noise
Agrawal_0.15	9	2	0.15	class noise
Agrawal_0.2	9	2	0.2	class noise
Sine_False	2	2	NO	attribute noise
Sine_True	4	2	YES	attribute noise
Led_0.05	24	7	0.05	attribute noise
Led_0.1	24	7	0.1	attribute noise
Led_0.15	24	7	0.15	attribute noise
Led_0.2	24	7	0.2	attribute noise

Six drift detection methods are accessible in *skmultiflow* (Montiel et al., 2018). We use the parameters recommended by *skmultiflow* for each drift detection method. We adjust the parameters of ADWIN for better performance. The detailed values or thresholds are shown in Table 4.3. The α and β of DDM, HDDM, HDDM_A and HDDM_W represent the warning and drift threshold. ADWIN, and PH follow a similar way but only have a drift threshold marked with δ. λ in HDDM_W represents the weight given the data, more recent data has less weight.

Table 4.3: Algorithm parameters or threshold description

Algorithms	**Parameters(Thresholds)**
NTDDM	$\alpha = 0.01, p\_value = 0.001$
DDM	$\alpha = 0.05, \beta = 0.01$
Adwin	$\delta = 0.05$
EDDM	$\alpha = 0.9, \beta = 0.95$
PH	$\delta = 0.005$
HDDM_A	$\alpha = 0.005, \beta = 0.001$
HDDM_W	$\alpha = 0.005, \beta = 0.001, \lambda = 0.05$

4.4.2 Measures for the Experimental Results

Here $S_i, i \in [1, 100]$ represents 100 experimental trials on each dataset. The O_i represents the output of drift detection method in *i*-th trial, which is the list of detected drift points, $[o_1, o_2, \cdots, o_n]$, $n \geq 0$. The detection method detects no drift when $n = 0$.

Detection delay is common for performance comparison (Lu et al., 2019). Based on this, we propose three measures for comparison:

- **Effective Detected Drift Rate (EDDR)**: the proportion of experimental trials where a detected drift time point falls into the Effective detected drift rate (EDR).
- **Distance to First Effective Detected Point (FEDP)**: whether a drift detection method can detect a drift point early enough.
- **Distance to Last Detected Point (LDP)**: this is to evaluate if the drift detection method could stop in time, instead of repeating the true $Drift\_flag$.

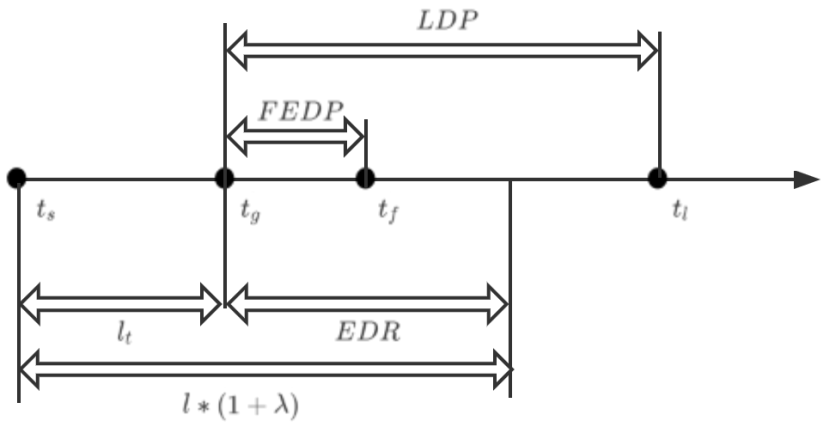


Figure 4.4: Illustration of the measures. x-axis represents time. t_s is start time point, t_g is drift time point, t_f and t_l are the first and last detected drift point. l_t is the time range from start to drift point. Then EDR, is our defined time range($l \times \lambda$) which represents the reasonable time range for detected drift point collection, t_e. FEDP represents the time range from drift point t_g to first drift point in t_e. LDP represents the time range from drift point t_g to the last detected drift point. These two time ranges are to evaluate if the detection method could detect the drift in a reasonable time range.

We use an example to explain the terms in Figure 4.4. There are 5 groups of outputs on Table 4.4, which are the experimental results obtained from five tested methods on the same dataset.

Table 4.4: Sample of detected drift points from experiments

Methods	Output
NTDDM	[581]
DDM	[39, 605]
Adwin	null
EDDM	[332,560,681,863]
PH	[725]

Based on our experiment setting, the start point of data stream t_s is from 0, drift point t_g is 500, and the EDR is between [500, 700]. Taking the output of NTDDM as an example, the detected point 581 is t_e, as it falls within EDR. At the same time, it also corresponds to t_f and t_l, since only one drift point is detected by NTDDM. However for Adwin, it detects no drift point, so the size of O is 0. On the other hand, the EDDM detects drifts at [332, 560, 681, 863], t_f is 560, and t_l is

863, then $FEDP$ is 60 and the LDP is 363.

Effective Detected Drift Rate

EDDR is used to measure the efficiency of drift detection methods. In some cases, a detection method may not be able to detect any drift at all (see ADWIN in Figure 4.5). Alternatively, a large number of false drift will be detected when noise level is high, e.g.(see EDDM in Figure 4.5).

The term Effective Detection Range (EDR), is considered as the effective and reasonable range for drift detection, therefore the detected drift points within the EDR are defined as effective detected point t_e. l_t in Figure 4.4 as the length from the start point t_s to the drift point t_g, and λ is the threshold for the length of EDR. Therefore, the EDR is defined as $[l_t, l_t * (1 + \lambda)]$.

If the detected concept drift is outside the EDR, this shows that the concept drift is detected too late. Ideally, there should be only one detected concept drift within the EDR. However, due to the sensitivity of detector, it will not detect the concept drift only once. Due to noise, they will also cause the detector to falsely report the concept drift. To account for both conditions, we propose that there should only be a limited number of detected drift points within EDR. In this study, we propose to set the number of detection to be 3, which is motivated by the experimental observation from all the datasets investigated in this work.

The EDDR is used as the indicator to evaluate if the method could detect drift within a reasonable time, as shown below:

$$EDDR = \frac{\sum_{i=1}^{N} E_i}{N} \tag{4.8}$$

In Equation 4.8, *N* is the number of trials on one dataset, E_i as an indicator, represents if the drift detection method performs successfully in i_{th} trial. For example, O is the list of detected drift points in one trial, $O = [o_1, o_2, \cdots, o_n]$. E_i is defined in Equation 4.9 as:

$$E_i = \begin{cases} 1 & \exists o_i \in EDR \; and \; 1 \leq |O| \leq 3 \\ 0 & otherwise \end{cases} \tag{4.9}$$

When E_i equals to 0, the following two conditions must be met at the same time, (1) At least one sample o in O is within EDR, (2) the number of elements in O must be between 1 and 3. The first condition shows at least one effective drift has been detected, while the second condition limits the number of false drifts.

The results of EDDR are shown in Table 4.5, the proposed NTDDM shows best detection performance on 13 out of 16 datasets. The corresponding visualization on the SEA dataset is shown in Figure 4.5. It is clear that drift points detected by NTDDM are concentrated next to the drift point t_g (red bar), even in the higher noise level. However, for other methods, the ability of detection declines significantly. It is observed that either the number of detection drifts greatly reduced, e.g. ADWIN (3rd), HDDM_A (6th), or a large number of false drifts reported, e.g. EDDM (4th), HDDM_W.

Table 4.5: Effective Detected Drift Rate(%) on 16 datasets from 7 methods, NTDDM perform best 13 out of 16 datasets.

	DDM	ADWIM	EDDM	PH	HDDM_A	HDDM_W	NTDDM
Sea_0	75	10	85	1	10	67	**85**
Sea_0.05	51	3	60	1	4	62	**74**
Sea_0.1	25	1	32	3	2	54	**74**
Sea_0.15	9	1	31	1	2	41	**69**
Sea_0.2	6	0	28	0	0	30	**49**
Agrawal_0	31	5	50	30	29	53	**55**
Agrawal_0.05	31	2	52	31	35	**55**	39
Agrawal_0.1	33	2	**58**	36	38	39	47
Agrawal_0.15	38	3	46	42	48	47	**50**
Agrawal_0.2	43	5	49	46	52	50	**53**
Sine_0	70	39	64	48	57	80	**83**
Sine_1	75	34	72	45	67	79	**82**
Led_0.05	61	**75**	73	0	0	27	71
Led_0.1	84	78	88	93	95	87	**95**
Led_0.15	84	93	87	94	93	89	**94**
Led_0.2	84	91	90	93	91	89	**94**
Average	50 ± 25.8	27.6 ± 34.6	60.3 ± 20.1	35.2 ± 33.1	38.9 ± 33.6	59.3 ± 19.9	69.6 ± 18.0

We have applied Friedman test and Nemenyi post hoc test on groups of results. The $p-value$ from Friedman test is much smaller than 0.05, therefore the null hypothesis that there is no significant difference between those methods is rejected.

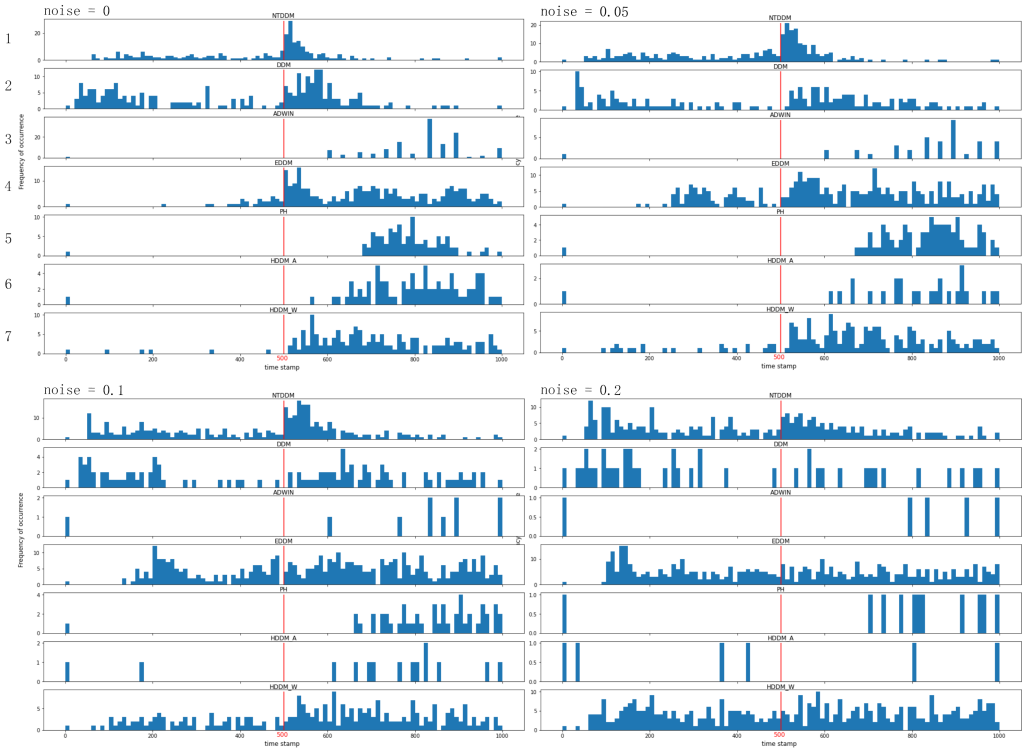


Figure 4.5: The histograms of the detected drift time points by the seven detection methods on dataset SEA. The noise levels are 0, 0.05, 0.15 and 0.2 respectively. X-axis represents the time point, the red bars denote the drift point t_g, whereas the blue bars are the histogram of detected drift points summarized over 100 independent experiments. The ideal case is that all detected drift points concentrate on EDR in Figure 4.4, the $EDDR$ indicates the degree of concentration in EDR of the detected points.
1:NTDDM, 2: DDM, 3: ADWIN, 4:EDDM, 5:PH, 6:HDDM_A, 7:HDDM_W

We then use Nemenyi to do post hoc test on pair-wise generated p-value. The result is visualized with heatmap Figure 4.6. It is clear that our method NTDDM is significantly better than DDM, ADWIN, PH and HDDM_A. Then for EDDM and HDDM_W, we compare the EDDR with them separately, and combined with average value in Table 4.5. It is apparent that our method shows higher EDDR than these two in most datasets.

Distance to First Effective Detected Point

Ideally, concept drifts should be detected quickly. Some detection methods detect multiple drifts when there is only actually one occurrence. We measure the first effective detected point, t_f that falls into EDR. We then use the distance FEDP between the t_g and t_f to evaluate NTDDM how promptly it can detect drift. Smaller FEDP represents the better drift detection performance.

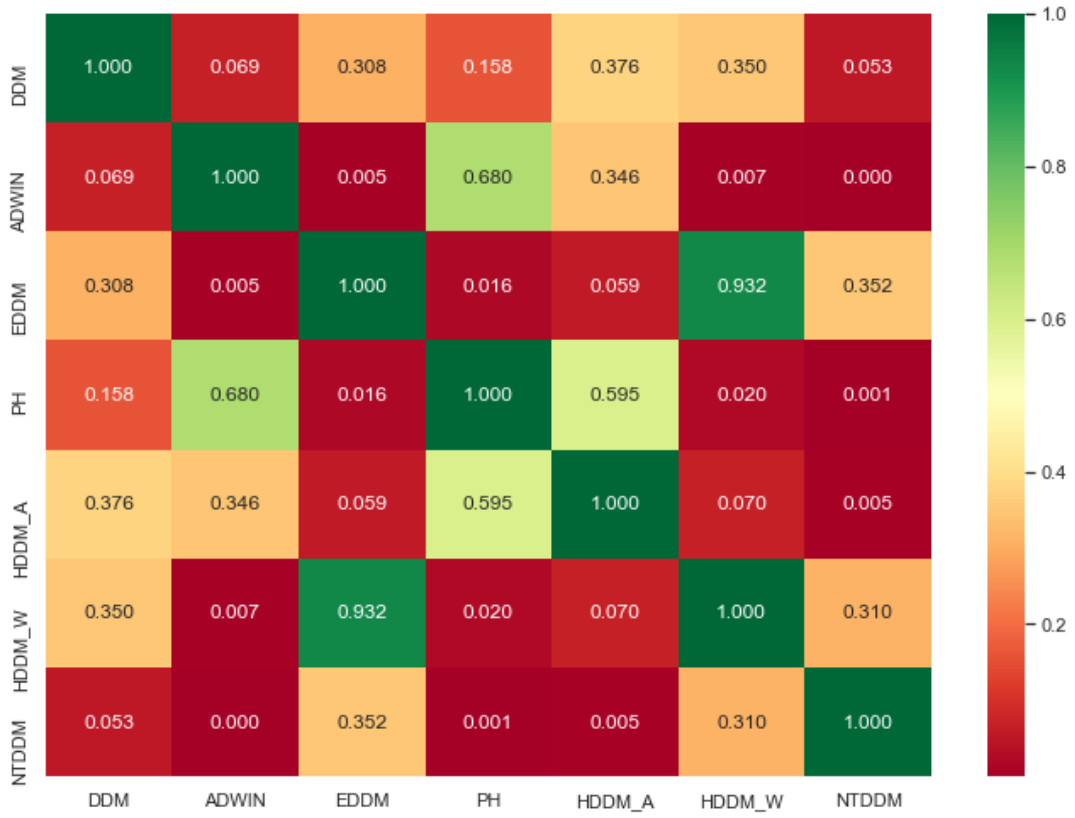


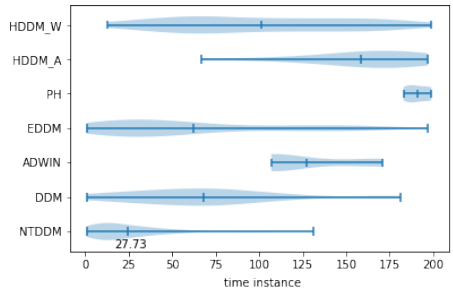
Figure 4.6: The heatmap for the Friedman test and Nemenyi post hoc test on result of EDDR, each data represents the p-value of statistical difference and is calculated in pairs. The red coloured squares corresponding the statistically significant differences, the squares in the other colors represents less significant differences.

We take the experiments on SEA dataset as an example. When the noise level are 0% and 20%, the results of FEDP are shown in the violin figures of Figure 4.7 (a) and Figure 4.7 (b). From the plots, the mean of FEDP from NTDDM is the smallest compared to other methods. This suggests that the proposed NTDDM can detect drift most quickly.

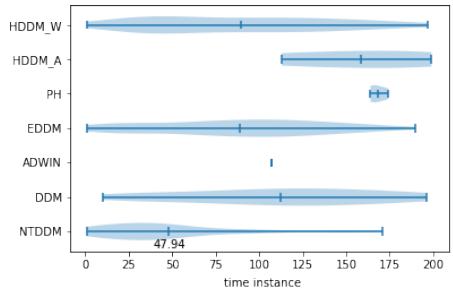
Distance to Last Detected Point

We take the same set of the experiments result as in the Section 4.4.2. Their LDP values are shown in Figure 4.7 (c) and Figure 4.7 (d). It is clear, on average, that NTDDM has obtained the smallest distance to Last Detected Point. Again, this suggests that the proposed NTDDM perform best in detecting drifts quickly.

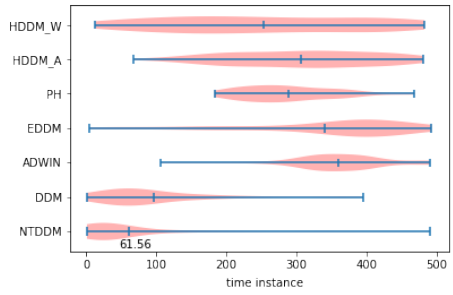
LDP values on four artificial datasets are shown in Table 4.6, it is clear that our proposed NTDDM has the best performance among the rest of the methods. It is worth mentioning that EDDM and



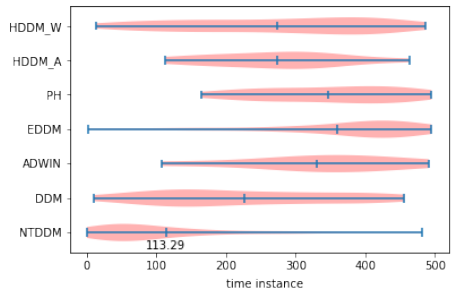
(a)



(b)



(c)



(d)

Figure 4.7: FEDP and LDP on SEA dataset. (a) FEDP with noise level = 0 (b) FEDP with noise level = 0.2 (c) LDP with noise level = 0 (d) LDP with noise level = 0.2.

HDDM_W which perform great in FEDP have quite large latency. The reason is that drift points detected by these two methods are spread across the whole range.

Table 4.6: LDP on four artificial datasets

	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W	NTDDM
SEA	113.53	320.33	416.34	325.18	298.34	255.21	89.91
AGRAWAL	224.21	430.12	430.64	183.65	181.25	455.58	164.35
SINE	81.12	301.68	389.52	188.03	78.15	425.11	76.15
LED	102.51	331.52	374.41	133.55	108.12	454.95	93.53
Average	107.84	345.91	402.72	207.60	166.46	397.71	87.98

4.4.3 Experiments on Real-world Dataset

To further evaluate the performance of NTDDM, we use a real world dataset Posture (Kaluža et al., 2010). The posture dataset collected data at 0.1s intervals, u, from sensors on left/right ankle right, belt and chest (Kaluža et al., 2010), each data instance is combined with 3 features and 1 class label, the features represent the spatial location and the class label represents the corresponding posture type. The posture type (class label) has been noted by a human observer. Figure 4.8 shows the posture data. The patterns of posture data are clearly different, before and after 500 u. Given the posture data, an ideal detection method should be able to detect such a change soon after 500 u. In many applications, the causes of such a change are unknown. However, in our controlled environment we know the cause of the abrupt change. A detected drift can be used to advise the need to investigate the causes. We select the data stream whose posture changes from sitting to walking, which we want to detect the change.

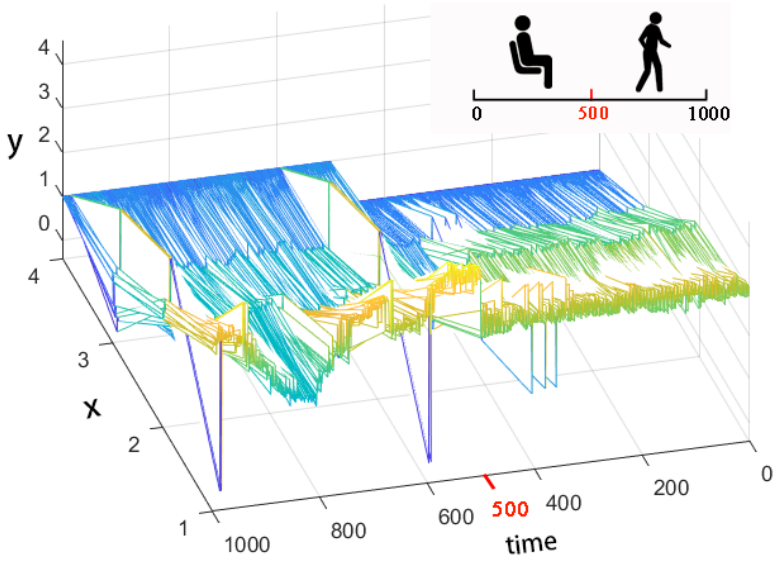


Figure 4.8: Data visualization of real-world posture data, x and y axis represents the result of data, axis of time represents the time range of the data stream.

This dataset contains data stream with noise and without noise. The noise level in the data stream with noise is unknown. The drift starting time t_g at 500 u. We ran 100 trials for each set of experiments. The result analysis and evaluation are based on three measures defined in Section 4.4.2.

Effective Detected Drift Rate

Table 4.7 shows that the NTDDM performs among the best in its ability to detect drifts when either no noise or noise level = 0.1, where the noise level is the noise variance as a fraction of signal variance.

Table 4.7: EDDR on real-world dataset, NTDDM has the highest value.

	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W	NTDDM
No Noise	93	90	73	71	73	33	**97**
Noise(10%)	84	70	81	52	74	35	**97**

Distance to First Effective Detected Point

Table 4.8 lists the average result of FEDP on two sets of experiments, from 100 trials. Figure 4.9 shows the boxplots of the FEDP. From the result it is seen that FEDP of NTDDM is just slightly

larger than EDDM and HDDM_W, which also demonstrated in sub-figure (a) and (b) in Figure 4.9. This suggest that our proposed NTDDM is the best in FEDP.

Table 4.8: The average values for 100 trials: FEDP and LDP on Posture dataset with and without noise

	No Noise		**Noise (10%)**	
	FEDP	LDP	FEDP	LDP
NTDDM	**41.02**	**65.84**	**42.08**	**82.98**
DDM	69.94	94.13	62.30	107.81
ADWIN	85.86	306.04	76.52	328.61
EDDM	43.77	245.87	42.35	289.34
PH	126.02	165.32	102.17	201.94
HDDM_A	48.07	118.93	57.25	156.39
HDDM_W	46.54	329.25	44.38	357.77

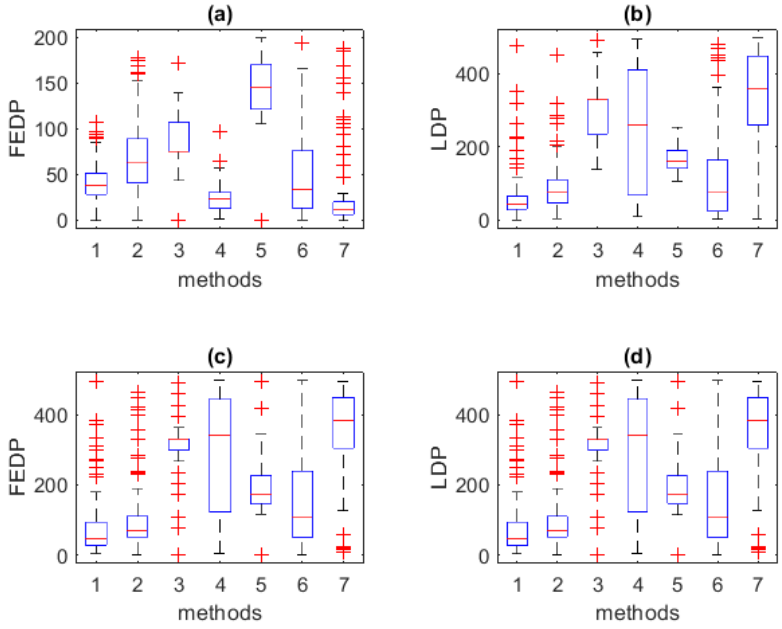


Figure 4.9: Boxplot visualization of the value of FEDP and LDP on Posture dataset, (a) FEDP on Posture dataset without noise (b) LDP on Posture dataset without noise (c) FEDP on Posture dataset with noise (d) LDP on Posture dataset with noise. The methods from left to right are 1:NTDDM, 2: DDM, 3: ADWIN, 4:EDDM, 5:PH, 6:HDDM_A, 7:HDDM_W.

Distance to Last Detected Point

The result of LDP is shown in Table 4.8, and in Figure 4.9. It is visually apparent that NTDDM obtains the best performance, detected the drift points within EDR. In summary, the experiments on the real world dataset show that the proposed NTDDM performs the best overall.

4.5 Summary

This chapter presents a noise tolerant drift detection method for data stream mining. It addresses the challenge of detecting drifts when noise is present in the data. The proposed NTDDM method introduces a detection function to detect the potential drift, and is then validated to determine whether it is caused by noise or not. The comparative studies have shown that the NTDDM method performs the best compared to the six established detection methods. Our work fills the research gap with a drift detection method designed specifically for data streams with noise. For many real-world applications where noise in data stream is present, our proposed Noise Tolerant Drift Detection Method will provide efficient, effective drift detection for better monitoring and decision support.

Chapter 5

Concept Drift Detection by Tracking Weighted Prediction Confidence of Incremental Learning

Data stream mining is great significant in many real-world scenarios, especially in the big data area. However, conventional machine learning algorithms are incapable to process because of its two characteristics (1) potential unlimited number of data is generated in real-time way, it is impossible to store all the data (2) evolving over time, namely, concept drift, will influence the performance of predictor trained on previous data. Concept drift detection method could detect and locate the concept drift in data stream. However, existing methods only utilize the prediction result as indicator. In this article, we propose a weighted concept drift indicator based on incremental ensemble learning to detect the concept. The indicator not only considers the prediction result, but the change of prediction stability of predictor with occurs of concept drift. Also, an incremental ensemble learning based on vote mechanism is especially used to get constantly updated value of indicator. Based on the experiment result on both benchmark and real-world dataset, our method could effectively detect concept drift and outperform other existing methods.

5.1 Introduction

The data stream is now very widely distributed in many real-world scenarios, especially in big data era. Such areas include IoT sensors network, social network and traffic information stream (Zyblewski et al., 2021). There are two common characteristics in these areas. The first is unlimited amount of data in the stream and limited memory cannot store all of them. Therefore, data stream mining must be operated online, process the newly input and discard the old data. The second is dynamic evolving environment of data stream, it will bring the concept drift problem. Concept drift means machine learning model trained on past data will be inaccurate, since the decision boundaries of the model has been changed (Lu et al., 2019). For example, a naïve bayes model applied on electricity price to do future price prediction. When new renewable energies (concept drift) added to the grid, the accuracy of the original prediction model will obviously reduce. Conventional machine learning algorithm always suffer from concept drift. There are two main strategies to deal with concept drift, active and passive (Gama et al., 2014). The active strategy will monitor the performance of model trained on data stream, and report a concept drift while pre-defined threshold is reached. On the other hand, passive strategy only self-update to adapt the concept drift rather than report it actively.

In this work, we focus on active concept drift detection strategy, aiming to detect the drift more accurately and with less delay. Similar to the concept drift process, there are two kinds of concept drift in a data stream, real and virtual (Lu et al., 2016). Given a data stream of length n, denoted as $D_{(1,n)} = [d_1, \ldots, d_n]$, where $d_n = (X_n, y_n)$ represents a single data sample in the data stream $D_{(1,n)}$. X_n and y_n are the feature and class label, respectively. If there is no concept drift in the data stream, then its joint distribution $JD_{(1,n)}(X, y)$ will remain stable. However, when concept drift happens at time $t + 1$, $JD_{(1,t)}(X, y) \neq JD_{(t,\infty)}(X, y)$, denoted as $\exists t, P_t(X, y) \neq P_{t+1}(X, y)$, where P_t represents the joint distribution of (X, y) at time t (Liu et al., 2020).

Based on Bayes' theory, $P(X, y) = P(X) \times P(y|X)$. Real concept drift refers to $P_t(y|X) \neq P_{t+1}(y|X)$ when $P_t(X) = P_{t+1}(X)$, indicating a change in conditional probability (Wang et al., 2018). Virtual concept drift is the opposite, $P_t(X) \neq P_{t+1}(X)$ when $P_t(y|X) = P_{t+1}(y|X)$, which only indicates a change in marginal probability. We focus on real concept drift.

In this section, we propose an incremental weighted performance drift detection method (IW-PDDM) by monitoring proposed weighted and combined indicator. The principal idea is first to construct the indicator, which chooses not only the accuracy as one of the indicators, but also considers the prediction stability. This is because the latter contains attributes related to precision rather than accuracy. Statistics is used to define two thresholds for concept drift detection and validation. Furthermore, an incremental vote-based ensemble predictor is developed on data stream to make prediction and it is capable to self-update with each data input. Finally, the update indicator will be triggered when concept drift occurs and it resets the predictor. The main contributions of this study are summarised as follows:

1. To the best of our knowledge, this work is the first to introduce prediction stability to concept drift;
2. Defining the weighted prediction stability and its calculation rule under vote-based ensemble learning;
3. Developing statistics to configure the threshold of concept drift detection and validation;
4. Developing an incremental vote-based ensemble learning strategy on data stream mining to deal with concept drift detection.

5.1.1 Prediction Stability

The efficacy of machine learning models is commonly gauged through outcome-oriented predictive monitoring techniques such as precision, recall, and Area Under the Receiver Operating Characteristic Curve (AUC) (Teinemaa et al., 2019). However, these metrics alone may not be sufficient to adequately measure predictive performance as argued by (Teinemaa et al., 2018). For instance, machine learning models are often prone to performance degradation when faced with previously unseen data, leading to undesirable consequences in domains that demand high precision, such as healthcare and public transportation. Therefore, it is imperative to consider the stability of a classifier, especially when it is employed for sequential predictions. In this study, we introduce a new perspective on gauging prediction stability by examining the distribution of class votes in a classifier. This distribution reflects the degree of stability in the predictions made by the classifier, thereby providing a more comprehensive evaluation of the model's performance. A

balanced distribution would indicate a stable predictive model, whereas an imbalanced one may signal potential instability. Hence, our approach provides an additional facet of performance evaluation, augmenting traditional metrics and offering a more holistic understanding of the model's robustness in the face of novel data.

5.2 Weighted Prediction Confidence

We propose an Incremental Weighted Performance Drift Detection Method (IWPDDM) to address the concept drift problem in data streams. The proposed IWPDDM is meticulously designed to cater to the dynamic nature of streaming data and is comprised of three major steps: incremental vote-based ensemble learning, detection, and validation. These steps synergize to form a robust and adaptive mechanism for drift detection.

5.2.1 Design of IWPDDM

The proposed IWPDDM has three steps, Incremental learning, Detection and Validation, shown as Figure 5.1. The first incremental learning step is executed in real time, incrementally to make prediction and update itself with new data sample input. With the prediction result and confidence from first step, weighted incremental indicator is constructed and updated. Then the detection step and validation step will be successively triggered when weighted incremental indicator trigger corresponding thresholds. The IWPDDM will report the drift and reset the incremental classifier.

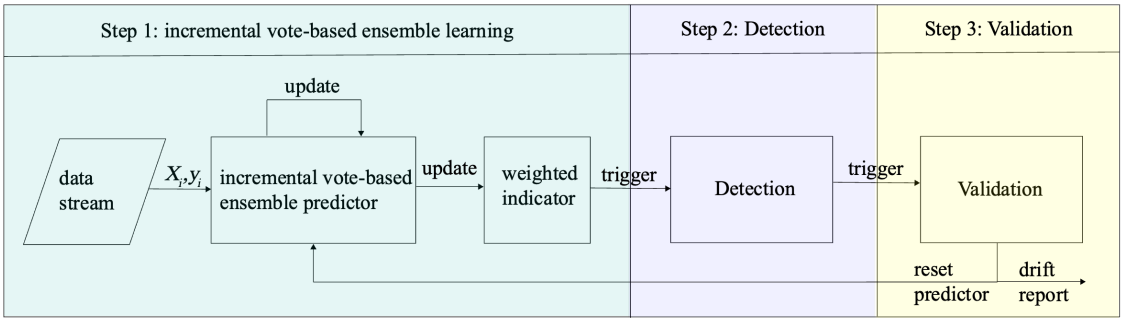


Figure 5.1: Design of IWPDDM, include three steps, incremental vote-based ensemble learning, detection and validation respectively. In step 1, weighted indicator will be updated and monitored. Once corresponding statistical thresholds are reached, step 2 and 3 will be triggered successively. When validation step validate a concept drift, signal for predictor reset will be sent, and drift will be reported synchronous.

5.3 Incremental Vote-based Ensemble Learning Step

The aim of incremental vote-based ensemble learning step is to dynamically train the model to make predictions with upcoming data samples. The performance of an incremental ensemble learning model will change when a concept drift occurs. The prediction accuracy of the model is considered as the first indicator, because it provides a general representation of the model's prediction performance. On the other hand, accuracy is inherently limited as it can only generate binary decisions, true or false.

Motivated by (Teinemaa et al., 2018), we define a term called prediction stability (PS) for incremental learning algorithms in data stream mining. Normally, a learning algorithm is considered unstable if small changes in the training set can cause significant changes in the predictor. In this section, we use an incremental vote-based ensemble learning algorithm as the predictor. Therefore, changes in PS can reflect the impact of concept drift on the incremental learning model.

We consider the weighted sum $S_{1,t}$ of the accuracy value and prediction stability as the indicator of performance over the time period from 1 to t, which is shown in Equation 5.1:

$$S_{1,t} = W_1 \cdot AC_{1,t} + W_2 \cdot PS_{1,t} \tag{5.1}$$

where $AC_{1,t}$ is the average value of accuracy of the incremental learning model from the starting time point 1 to t, $PS_{1,t}$ is the average value of prediction stability , W_1 and W_2 are the weight values for $AC_{1,t}$ and $PS_{1,t}$ respectively. To maintain focus on our contribution, the weight value allocation strategy always uses the average weight value. $AC_{1,t}$ is the ratio of the number of correct predictions to the time period from the start to time point t. $PS_{1,t}$ is the average value of PS from the start to t, as shown in the Equation 5.2:

$$PS_{1,t} = \frac{\sum_{i=1}^{t} PS_i}{t} \tag{5.2}$$

where PS_i represents the degree of prediction stability of the predictor at time point i, which is calculated by the Equation 5.3:

$$PS = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} |NC_i \cdot H_i - NC_j \cdot H_j|}{\sum_{i=1}^{N-1} H_i + \sum_{j=i+1}^{N} H_j} \cdot N \tag{5.3}$$

where NC_i represents the stability of the predictor for class i, and H_i is the corresponding weight value. As mentioned before, the weight strategy always uses the average weight, so $H_i = H_j = \frac{1}{N}$. N is the number of classes in the data stream (e.g., N equals 2 in a binary classification task). NC_i is calculated using the Equation 5.4:

$$NC_i = \frac{\text{Number of classifiers vote for class } k}{\text{Number of all classifiers}} \tag{5.4}$$

Since the predictor used in this section is a vote-based ensemble learning model, the distribution of vote numbers for each class reflects the prediction result. For example, in a binary classification task, $A_1 = A_2 = 0.5$ represents the worst PS, as it is almost the same as random classification with a PS value equal to 0 based on Equation 5.3. Then the highest PS value 1 will be reached when A_1 equals 1 or 0. Therefore, the value interval of PS is $[0, 1]$, representing the degree of prediction stability from highest to random.

Based on Equation (5.1)-(5.4), our proposed indicator for drift detection is the weighted sum $S_{1,t}$ of accuracy ($AC_{1,t}$) and prediction stability ($PS_{1,t}$). The generation of $S_{1,t}$ is shown in Algorithm 4.

Algorithm 4 Incremental Vote-Based Ensemble Predictor Learning and Generation of $S_{1,t}$

```
Input: Data stream (X_i, y_i), initial incremental vote-based ensemble predictor M
Output: S_{1,t}, AC_{1,t} = 0, PS_{1,t} = 0
while data stream continues do
    Perform partial training with incoming data using M
    Update and make predictions using M
    for prediction result on each class do
        Calculate overall AC_t and PS_t (based on Equation 5.3 and 5.4) at time t
    end for
    Calculate AC_{1,t} and PS_{1,t} based on Equation 5.2
    Calculate S_{1,t} based on Equation 5.1
end while
```

In Algorithm 4, the incremental vote-based ensemble learner is initialized and dynamically trained with incoming data while simultaneously making predictions on the input data. The values of AC_t and PS_t are then calculated at time point t. Finally, the indicator $S_{1,t}$ is calculated based on the

availability of $AC_{1,t}$ and $PS_{1,t}$.

5.4 Detection and Validation Step

Change of the indicator $S_{1,t}$ reflects the performance change of the predictor. If no concept drift occurs, it will gradually increase until a certain amount of data is reached and then stabilize. However, in the presence of concept drift, both accuracy and prediction stability, which are two components of $S_{1,t}$, will decrease.

To define the threshold for concept drift detection, the Three-sigma rule is used. The detection threshold is set at 2 (95%) standard deviations, and the validation threshold is set at 3 (99.7%) standard deviations. Therefore, the standard deviation of the indicator $S_{1,t}$ is calculated as $\text{sd}(S_{1,t}) = \sqrt{S_{1,t} \cdot (1 - S_{1,t})}$.

Algorithm 5 Detection and Validation Step

Input: Indicator value $S_{1,t}$
Output: Drift point
Initialize: *drift_flag* = False, $S_{\max}$ = 0, $\text{sd}(S_{\max})$ = 0
while $S_{1,t}$ continues **do**
    Calculate its standard deviation $\text{sd}(S_{1,t})$
    if $S_{1,t} - \text{sd}(S_{1,t}) > S_{\max} - \text{sd}(S_{\max})$ **then**
        $S_{\max} = S_{1,t}$
        $\text{sd}(S_{\max}) = \text{sd}(S_{1,t})$
    end if
    if $S_{1,t} - \text{sd}(S_{1,t}) < S_{\max} - 2 \times \text{sd}(S_{\max})$ **then**
        drift_flag = "Detection"
    end if
    if $S_{1,t} - \text{sd}(S_{1,t}) < S_{\max} - 3 \times \text{sd}(S_{\max})$ **then**
        drift_flag = "Validation"
    end if
end while
Return *drift_flag*

In Algorithm 5, the max value of the $S1{,}t$ and its standard deviation will update incrementally. At the same time, the drift_flag will be updated when corresponding threshold is reached. When the drift_flag is set to Validation, it means concept drift is validated and predictor should be reset for retraining from scratch.

5.5 Experiment and Evaluation

This section compares the proposed IWPDDM with the six existing drift detection methods, over four benchmark datasets and one real-world dataset. Section 5.6 describes the datasets and drift detection methods used in experiment. Section 5.7 analyzes the experimental results and compares with other 6 detection methods.

5.6 Datasets and Drift Detection Methods

Four benchmark data streams widely used in drift detection literature are selected for experiment and evaluation, namely SEA, SINE, AGRAWAL, and LED. They are available at public stream data mining platform (Montiel et al., 2018). Descriptions of the datasets are in Table 5.1. Each data stream has 1000 time points, the concept drift occurs at time point (CDT) 500. To compare the performance of drift detection method, we test the methods on each dataset for 100 trials which are generated with different random seeds.

Table 5.1: Description of the datasets

Dataset	No. of attributes	No. of classes	Drift width	Drift type
SEA	3	2	1	abrupt
SINE	2	2	1	abrupt
AGRAWAL	9	2	1	abrupt
STAGGER	3	2	1	abrupt

The real-world dataset used in this section is Posture dataset, which is collected at 0.1s intervals(u), from sensors on ankle, belt and chest. Its class label is made by a human observer. The patterns of each data stream of posture dataset are clearly different, before and after 500 u. So, the concept drift point in Posture is time point 500.

Six drift detection methods are accessible in *skmultiflow* [18]. We use the parameters recommended by *skmultiflow* for each drift detection method. The detailed parameters are shown in Table 5.2.

5.7 Experiment Result and Evaluation

We use E_i to represent an independent experiment, which will run 100 times on both benchmark datasets and real-world dataset. Concept drift point will be reported if drift detection successfully

Table 5.2: Methods parameters

Algorithms	Parameters
IWPDDM	$\alpha = 2, \beta = 3$
DDM	$\alpha = 0.05, \beta = 0.01$
Adwin	$\delta = 0.05$
EDDM	$\alpha = 0.9, \beta = 0.95$
PH	threshold=30, $\delta = 0.005$
HDDM_A	$\alpha = 0.005, \beta = 0.001$
HDDM_W	$\alpha = 0.05, \beta = 0.01, \lambda = 0.05$

detects the drift. To compare the performance of IWPDDM with other 6 drift detection methods, we use drift detection accuracy and delay. The drift detection accuracy measures how the drift detection method works with concept drift. To compare it with other 6 drift detection methods, in terms of the ratio of the number of drifts detected data stream to the total 100 trails. The higher the accuracy, the better the performance.

Table 5.3: Drift detection accuracy (%)

Methods	IWPDDM	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W
SEA	89	75	13	96	5	18	71
SINE	99	93	51	100	55	71	96
AGRAWAL	95	50	14	96	37	47	94
STAGGER	94	86	84	80	76	86	91
POSTURE	96	94	94	94	82	95	91
Average	94.6	79.6	51.2	93.2	51	63.4	88.6

From the result of drift detection accuracy (Table 5.3), our proposed IWPDDM has the highest value among all the methods. On the other hand, we also compare the performance of drift detection methods based on drift detection delay. Drift detection delay refers to the distance between detected drift point and CDT. The smaller value of delay indicates the better drift detection performance. Then the Figure 5.2 describe the result of the drift detection delay. From the boxplot, it is clear that our proposed IWPDDM has the lowest value of drift detection delay among all the methods.

Based on the obtained result of drift detection accuracy and delay, it is evidenced that our proposed IWPDDM has led to the best performance of drift detection among other methods by highest average drift detection accuracy, 1.4% improvement compare to second method EDDM, and IWPDDM also have lowest drift detection delay, which is much smaller than others.

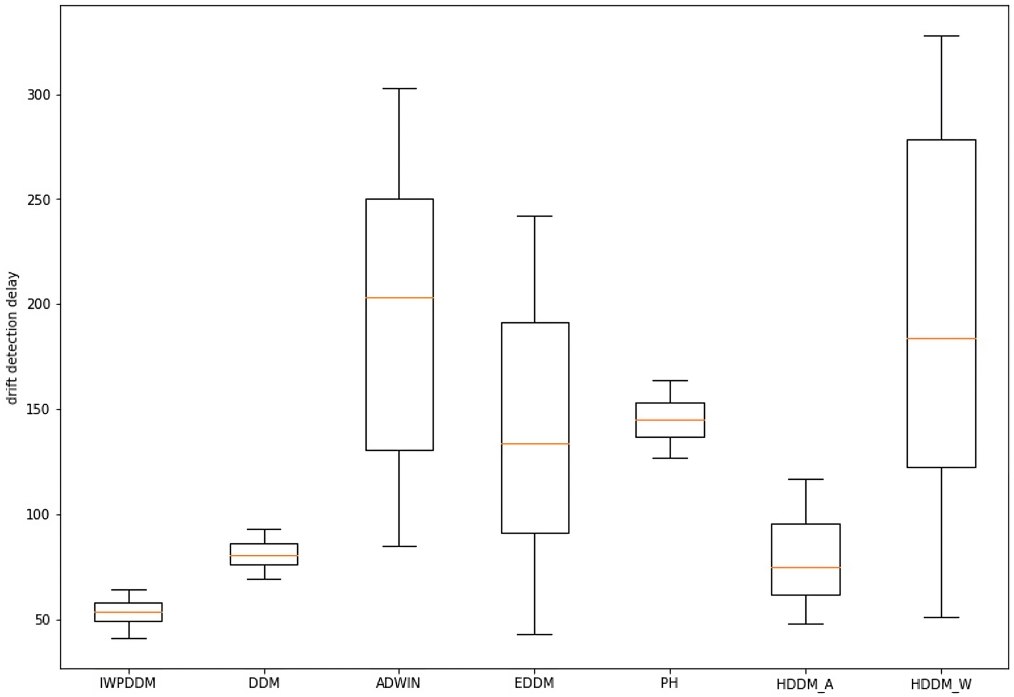


Figure 5.2: Drift detection delay result, x-axis represents the methods used for comparison and y-axis is the drift detection delay. From the boxplot, our proposed IWPDDM method has the lowest delay among all the drift detection methods

5.8 Summary

This chapter presents an incremental weighted performance drift detection method (IWPDDM) for data stream mining. It addresses the concept drift issue by monitoring the performance of predictor incrementally trained on data stream. The prediction accuracy and stability are both used to construct the weighted indicator for the concept drift. The weighted indicator will then also update with incremental learning model. Once the indicator reaches a pre-defined statistical threshold, the concept drift will be reported immediately. The comparative studies have shown that the proposed IWPDDM performs best among the other six established detection methods, with better detection accuracy and delay.

Chapter 6

Concept Drift Detection by Model-centric Transfer Learning Framework

Concept drift refers to the inevitable phenomenon that influences the statistical features of the data stream. Detecting concept drift in data streams quickly and precisely remains challenging, and failure to detect it will render model trained on historical data ineffective. Current drift detection methods suffer from the following problems: detection delay, many false detected drifts, and rarely utilize the deep neural network directly in the field of concept drift detection, which is considerably competent at addressing the classification problems of data stream. Furthermore, the output of the model is usually taken as a metric to detect drift, while changes in the model parameters are often ignored which contain highly useful information. In this section, we propose a model-centric framework for concept drift detection (MCDD) that uses deep neural network to detect concept drift by focusing on the change in the model itself, rather than the model output. In addition, transfer learning is developed to accelerate the drift detection process and reduce the computational complexity by freezing parts of the network. To further reduce false detected drifts, we propose long and short time windows method to determine the real drift from the potential detected drift. Experiments with real-world and artificial datasets have been undertaken to demonstrate the effectiveness of the proposed framework.

6.1 Introduction

Today, big data is constantly created from such sources as social media, online stores, the cryptocurrency market, and other places. This real-time generated data is called data stream. The results analyzed from the data stream can assist in making smarter decisions. However, the nature of the real-time data stream is dynamic, and it poses problems for models trained with previous data. The question is, how does the model keep stable performance on unstable data streams? There are many factors that cause models to lose effectiveness, and concept drift is one of the unavoidable problems. It is caused by changes in the underlying pattern, which subsequently cause a large downturn in the accuracy of the trained model after the drift occurs.

For convenience, the notations in this section are listed in Table 6.1.

Table 6.1: Notation and abbreviations

Notation	**Description**
$H_{1,n}$	data stream $H_{1,n} = [h_1, \cdots, h_n]$ beginning from time point 1 and ending at n
h_i	single data sample at time point *i* in H, $h_i = (X_i, y_i)$
$D_{i,t}$	the distribution of the data stream between time point *i* and *t*
L	layer of the deep neural network, e.g. L_I represents input layer
δ	neuron of the deep neural network
W	weights stream
w	weight of the neuron
t_s	the starting time of drift
t_f	time point of first detected drift
t_l	time point of last detected drift
S_i	i-th independent experiment
O_i	output, $O_i = [o_1, o_2, ..., o_n]$ in one trial
EDR	effective detected range, $[L_t, L_t * (1 + \gamma)]$
$FEDP$	time span between t_s and t_f
LDP	distance between t_s and t_l

There are two types of concept drift may exist in data stream. Given a data stream containing of a number of samples, indicate by $H_{1,n} = [h_1, \cdots, h_n]$, where $h_i = (X_i, y_i)$ represents each sample in data stream. X_i is the feature vector of d dimensions , and y_i is the class label. The data stream typically follows a joint distribution. $D_{1,t}(X, y)$ at time point $t + 1$, a drift occurs when $D_t(X, y) \neq D_{t+1}(X, y)$, denotes as $\exists t, P_t(X, y) \neq P_{t+1}(X, y)$ (Lu et al., 2016) .

The decomposition of joint probability $P_t(X, y)$ follows the formula $P_t(X, y) = P_t(X) \times P_t(y|X)$. According to Bayesian theory, there are two types of concept drift (Souza et al., 2020): real concept drift, indicating a change in posterior probability, denoted as $P_t(y|X) \neq P_{t+1}(y|X)$ while $P_t(X) = P_{t+1}(X)$, and virtual concept drift shown by a change in marginal probability, denoted as $D_t(X) \neq D_{t+1}(X)$ while $P_t(y|X) = P_{t+1}(y|X)$. The majority of existing articles regard concept drift as a change in the posterior probability because it indicates a change in the decision boundary (Lu et al., 2019) (Gama et al., 2014) (Wang et al., 2018). Our work focuses on detecting real concept drift in the data stream.

In order to solve the problems caused by real concept drift, a number of concept drift detection methods have been previously proposed. The purpose is to detect drift after it occurs, and then adjust the model according to the detection results to adapt to the new data stream. However, the existing works usually only monitor the output performance changes of the model to judge whether drift occurs. These models are usually conventional machine learning models, but today's data streams have more complex structures. Compared with traditional machine learning, the deep learning model with stronger fitting ability and robustness has not been applied in the field of concept drift detection as far as we know. However, the problem is that deep learning usually requires more computing resources and time. Under the same computing resources, the processing time is slower than that of conventional shallow models with simple structures. Excessive delay is not conducive to drift detection.

We proposed a MCDD in streaming data. Firstly, the initialised deep neural network will be trained and evaluated on base data without concept drift, while a second network with an identical structure will be partially frozen based on the transfer learning principle. The partially frozen network will merge with the pre-trained network's weights and only execute training mode on data containing concept drift. Transfer learning integration could considerably accelerate the concept drift detection process and reduce its computing cost. Furthermore, the strategy of long and short

time windows is designed to collect the weights of the transferred network's final hidden layer. In the end, the real drift will be detected, while the false drift will be filtered away.

Our contributions are:

1. Proposing a model-centric deep neural network framework to detect the concept drift in data streams.
2. Incorporating transfer learning to enable the drift detection more efficient and precise.
3. Proposing a long and short windows strategy to accurately detect and validate real drifts while filtering out false drifts.
4. Three types of concept drift, i.e., abrupt drift, early abrupt drift and incremental drift, will be constructed on both artificial and real-world datasets. In addition, the performance of various drift detection algorithms will be evaluated.

The rest of chapter is organized as follows: The proposed method is presented in Section 6.2. Experiments and results are discussed in Section 6.3. Finally, the summary is provided in Section 6.4.

6.2 Methodology

The purpose of this work is to accurately detect real concept drift and reduce false drift. The real concept drift will degrade the performance of the model trained with historical data. We design a deep neural network with transfer learning to detect real concept drift. When new data becomes available, the trained model will adapt accordingly. Therefore, if a deep neural network is selected as the training model, the weights of the network change to indicate the model change. A statistically significant change in the weights of the neural network represents the occurrence of concept drift.

Model initialization and pre-training, transfer learning, and model-centric concept drift detection are the three main parts of the framework we propose. In accordance with the three sections depicted in Figure 6.1, respectively.

In the first section, we will generate two neural networks with the same structure: one that will be trained and evaluated on base data, and another in which the majority of neurons are frozen,

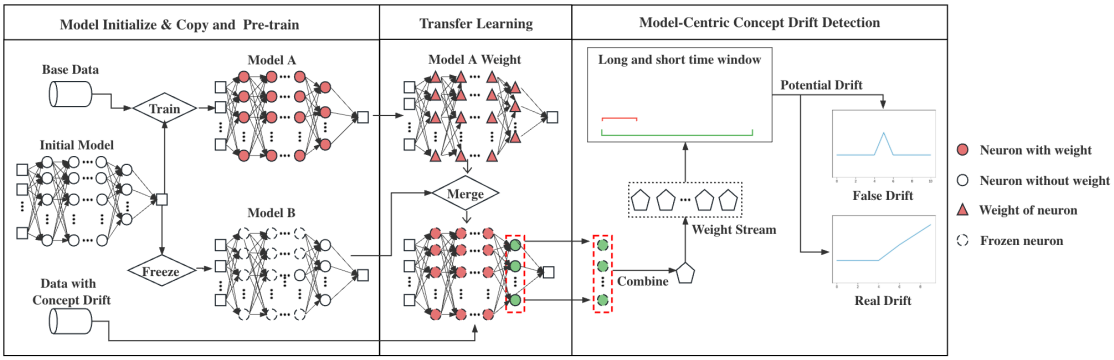


Figure 6.1: Flowchart of the Model-Centric Transfer Learning Framework on Concept Drift Detection.

leaving only neurons on the last layer to update their own weights during train mode. Then, in the second section, primarily merge the partially frozen model with the weights from the pre-trained model, and then apply the newly fitted model only in train mode to the data with concept drift; hence, only neurons on the last layer can update their weights. The last phase will monitor the weight stream that represents the change of entire model, and then the aggregated weights will be fed into the long and short time windows in order to detect the real drift and filter out the false drift.

6.2.1 Model Initialization and Pre-training

In Figure 6.1 the first part mainly does two tasks, the first is to initialise a deep neural network model, M_i, and then use base data to train on it and get a trained model, M_A; the second is to reproduce and reconstruct a model, M_B based on M_i, where part of the neurons are frozen.

We use L to represent the hidden layers in a deep neural network as the left side of the Equation 6.1. The input layer, the output layer, and the hidden layer are represented by the variables L_I, L_O, and L_n, where $n \geq 1$, respectively.

And on the right side of the Equation 6.1, $\delta_{n,m}$ represents the m_{th} neuron in the n_{th} layer

L_n.

$$\begin{bmatrix} L_I & L_1 & \cdots & L_{n-1} & L_n & L_O \end{bmatrix} = \begin{bmatrix} \delta_{I,1} & \delta_{1,1} & \cdots & \delta_{n-1,1} & \delta_{n,1} & \delta_{O,1} \\ \delta_{I,2} & \delta_{1,2} & \cdots & \delta_{n-1,2} & \delta_{n,2} & \delta_{O,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \delta_{I,m} & \delta_{1,m} & \cdots & \delta_{n-1,m-1} & \delta_{n,m} & \delta_{O,m} \end{bmatrix} \tag{6.1}$$

The first task aims to pre-train a deep neural network with base data. Training will be similar to conventional machine learning training procedures: train and evaluation phases. After the training of the model is finished, the neuron and its weights will be saved and used in the second stage.

The second task is to rebuild a model based on M_i. The neural network structure of M_B is the same as that of M_i, but a portion of the neuron is frozen. Furthermore, the neuron in a deep neural network has two modes: active and frozen (Csiszár et al., 2023); the active mode means the neuron can be updated during the model training. The neuron, on the other hand, cannot change when it is in frozen mode, which is represented by $\bar{\delta}$.

And to perform the idea of model-centric, no matter what kind of neural network structure is applied to the base data, the last layer of the network is always the linear layer, and we freeze all the hidden layers except the last layer, the partly frozen network shown by the following matrix:

$$\begin{bmatrix} \delta_{I,1} & \bar{\delta}_{1,1} & \cdots & \bar{\delta}_{n-1,1} & \delta_{n,1} & \delta_{O,1} \\ \delta_{I,2} & \bar{\delta}_{1,2} & \cdots & \bar{\delta}_{n-1,2} & \delta_{n,2} & \delta_{O,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \delta_{I,m} & \bar{\delta}_{1,m} & \cdots & \bar{\delta}_{n-1,m-1} & \delta_{n,m} & \delta_{O,m} \end{bmatrix} \tag{6.2}$$

As the matrix in Equation 6.2 shows, only the neurons from L_1 to L_{n-1} of M_B are frozen. The neurons $[\delta_{n,1}, \cdots, \delta_{n,m}]$ in the last layer L_n is able to be updated.

6.2.2 Transfer Learning

In this section, transfer learning is used to train and perform drift detection in a model-centric way. The principle behind it is that the network model after transfer learning (Mahdavi et al., 2022) is

still highly sensitive to data with drift, and it can also speed up the detection efficiency of drift while reducing computing consumption. The workflow is shown in Figure 6.2.

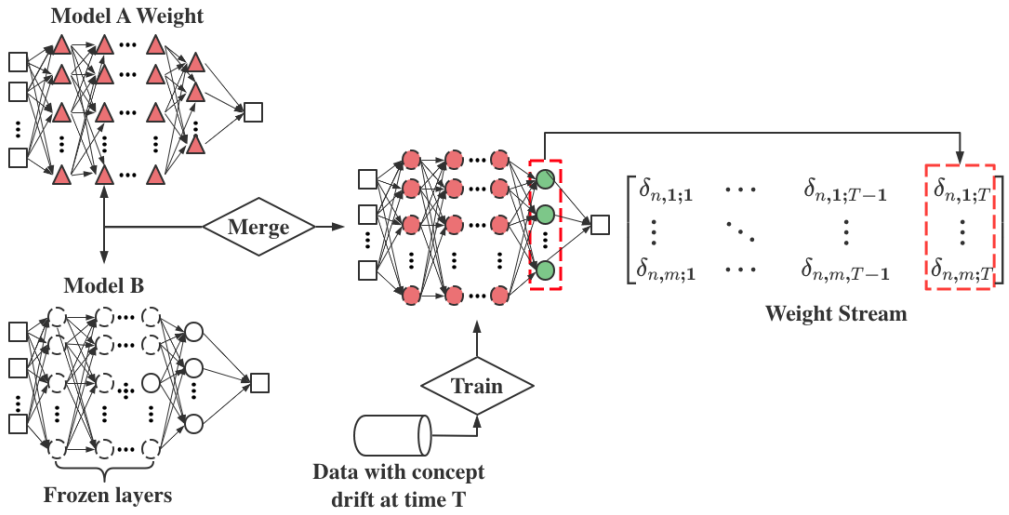


Figure 6.2: Flowchat of Transfer Learning, there are three steps. First, merging model B with weights from pre-trained model A. Then, Model B performs training with data with concept drift. Lastly, the weights (green neurons in red dashed box) from last layer form the Weight Stream.

Firstly, the M_A weights will be combined with the partially frozen neural network M_B, leaving only the last hidden layer L_n is activated. The fitted M_B will then be trained on concept drift data, and the weights of the final hidden layer L_n will be collected for drift detection.

The idea of being model-centric in this section is that we conduct drift detection by observing changes in the model itself, instead of recording and analysing the output of the model to detect drift. Specifically, instead of collecting the weights of the entire model, we innovate the concept of transfer learning to only activate the last layer of the model, and only collect the weight changes of the last layer to ensure the efficiency requirements of drift detection. The specific proposal steps are shown in Algorithm 6:

From the algorithm 6, the weights from M_A will be assigned to M_B, then each data instance will feed into M_B and deep neural network will start to train with $Data_{drift}$. The highlighted train indicates that we only use train mode in this phase, as opposed to the train and evaluation in pre-training Section 6.2.1. The reason for this is that the deep neural network learning process has two modes: training and evaluation (Chen et al., 2022). In the train mode, each neuron in the network could be updated; on the other hand, the weight of the neuron is fixed in evaluation mode.

The principle is that once a deep neural network is trained on a historical data, it will regard

Algorithm 6 Transfer learning for MCDD

Require: weights δ of pre-trained M_A, partly frozen M_B, data with concept drift $Data_{drift}$, Weight stream W

assign δ to M_B
while $Data_{drift} \neq Empty$ **do**
    $T = 1$
    for each data instance $Data_{each}$ in $Data_{drift}$ **do**
        $T \leftarrow T + 1$
        $Data_{each}$ **train** on M_B (only perform train, no evaluation)
        Collect and save weights $[\delta_{n,1} \cdots \delta_{n,n}]$ in W with label w_T
        Discard $Data_{each}$
    end for
end while

the data with concept drift as a new task, degrading the performance of the neural network on new tasks. In short, the emergence of the concept drift, will lead to major changes in the originally stable deep neural network model. Our method is able to detect concept drift by observing changes in the model itself, rather than observing its predictive performance on new data. Furthermore, we found that it is not necessary to monitor the entire change of the model, and only one layer of the deep neural network model is enough to reflect the change of whole model. In our framework setting, we only leave the last hidden layer L_n for the system to train and update itself. The single weight vector is $[\delta_{n,1;T}, \delta_{n,2;T}, \cdots, \delta_{n,m;T}]^{tr}$, which corresponds to the last column (red dashed box) of Weight Stream at time T. Each column of weight stream represents the state of the deep neural network at each time point. The weight stream will be then used as input to detect drift in long and short time windows.

6.2.3 Model-Centric Concept Drift Detection (MCDD)

Our MCDD focuses on the change in the model itself instead of the output. To do this, we use pre-training and transfer learning to take the model's weights and use them as features for drift detection.

In previous section, the weights has been incrementally updated during the transfer learning stage, and compose the weight stream matrix $W = [w_1, w_2, ..., w_T]$, where w_T is the weight of last hidden layer L_n collected at time T, equivalent to $[\delta_{n,1;T}, \delta_{n,2;T}, \cdots, \delta_{n,m;T}]^{tr}$. If there is no drift, refer to $P_t(y|X) = P_{t+1}(y|X)$ while $P_t(X) = P_{t+1}(X)$, indicates the properties of data stream remain unchanged. Then, model trained on historical data do not experience knowledge

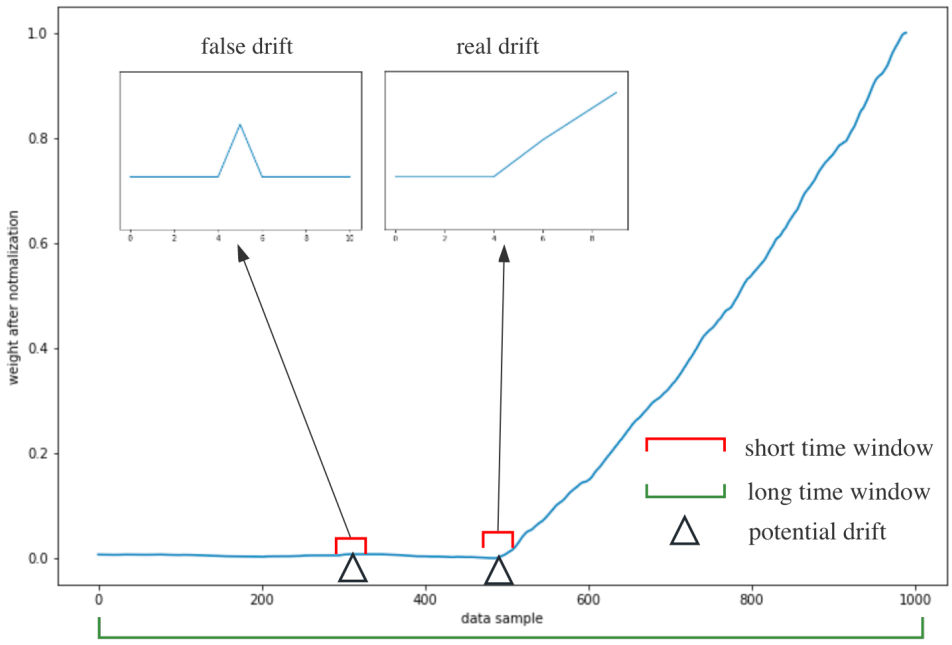


Figure 6.3: Long and short time windows, its goal is to detect real drift while filtering out false drift. In this case, the real drift point starts at 500 with a duration of 1. But there is more than one potential drift detected by the long time window. Then short time window will be initialized and validate whether the potential drift is true drift or false drift.

updates; therefore, the weights will remain relatively stable.

On the other hand, the model will update its weights when the posterior probability of data changed in the stream, refer to $P_t(y|X) \neq P_{t+1}(y|X)$ while $P_t(X) = P_{t+1}(X)$. It will trigger the model to update itself to adapt to the new concept. The change in the model is reflected in the weight change of the neurons. In order to improve the detection efficiency, MCDD only perform drift detection by observing the change of weight on the last hidden layer L_n.

There are two steps for drift detection: weights aggregation and the long short time windows method. The first step, aggregation will convert the weight vector w_T to a single value to accelerate the drift detection efficiency. The second step aim to use long and short time windows strategy to detect the real concept drift and filter out the false drift.

The aggregation as first place is because there are m number of neurons in L_n, where $|m| \geq 1$. The amount of data that needs to be processed in the time period T can drop from $|m \times T|$ to $|T|$ when we use data aggregates instead of whole weight stream matrix. To make our work focus on the drift detection framework, we use the arithmetic mean for aggregation, therefore the aggregated

weight $\dot{w_T}$ in time T is shown as:

$$\dot{w_T} = \sum_{i=1}^{m} \delta_{n,i;T}/m \tag{6.3}$$

where $\delta_{n,i;T}$ is the weight of m_{th} neuron in L_n. The aggregated weight stream denotes as $\dot{W}$, where $\dot{W} = [\dot{w_1}, \dot{w_2}, \cdots, \dot{w_T}]$, then will be fed into the long and short time windows for drift detection.

In the second step, a long and short time windows method is employed to validate real concept drift. These two sub-windows have a progressive relationship: the long time window starts accumulating data as the detection phase begins, while the short time window is activated upon capturing potential drift. To mitigate false drift caused by noise or abnormal cases, we adopt a strategy from the article (Wang et al., 2022) that involves randomly subsampling instances from the short time window, the sampling ratio used in this section is 50%. This approach not only maintains data continuity but also minimizes the likelihood of false drift. The workflow is illustrated in Figure 6.3.

The Algorithm 7 illustrates the rule for determining real drift from potential drift and filtering out the false drift.

In lines 4–7, the parameters max_weight, max_std, min_weight, and min_std are initialised with the values $weight\_pre\_train$ and std_pre_train pairwise. These two parameters, $weight\_pre\_train$ and std_pre_train, which are derived from the pre-train stage. It takes advantage of the stable weights from the pre-train phase, and lets the initialization of the max_weight, max_std, min_weight, min_std to be more smooth. Instead of initialing the value to ∞ or $-\infty$, which will cause high fluctuations at the beginning of the detection phase.

The calculation of $weight\_pre\_train$ and std_pre_train, utilizes the weights of the model from the pre-training phase in Section 6.2.1. The reason is that the model and train data will tend to fit to the maximum extent in the final stage of training. Therefore, the model at this time will be relatively stable. Then, we will collect the last 500 sets of weights of last hidden layer of model, denoted as $w_{pt}T$, where T is the time point. Then it will be used in Equation 6.4.

Algorithm 7 Model-Centric Concept Drift Detection

Input: aggregated weight stream $\dot{W}$, $drift\_flag$, $potential\_drift\_flag$, max_weight, max_std, min_weight, min_std, $weight\_pre\_train$, std_pre_train, Time point T
Output: $drift\_flag$
if $weight\_pre\_train \neq 0 \quad and \quad std\_pre\_train \neq 0$ **then**
    $max\_weight \leftarrow weight\_pre\_train, \quad max\_std \leftarrow std\_pre\_train$
    $min\_weight \leftarrow weight\_pre\_train, \quad min\_std \leftarrow std\_pre\_train$
end if
while $\dot{W} \neq Emtpy$ **do**
    current aggregated weight $\dot{w}_i$ at time point i
    if $max\_weight - max\_std \leq \dot{w}_i - std_i$ **then**
        $max\_weight \leftarrow \dot{w}_i, \quad max\_std \leftarrow std_i$
    end if
    if $max\_weight - max\_std \geq \dot{w}_i - std_i$ **then**
        $potential\_drift\_flag \leftarrow True$
    end if
    if $min\_weight + min\_std \geq \dot{w}_i + std_i$ **then**
        $min\_weight \leftarrow \dot{w}_i, \quad min\_std \leftarrow std_i$
    end if
    if $min\_weight + min\_std \leq \dot{w}_i + std_i$ **then**
        $potential\_drift\_flag \leftarrow True$
    end if
    if $potential\_drift\_flag == True$ **then**
        Short window start to collect nine more $potential\_drift\_flag$
        if All ten continuously $potential\_drift\_flag$ are all $True$ **then**
            $drift\_flag \leftarrow True$
        else
            $drift\_flag \leftarrow False$
            Stop and empty short window collect
        end if
    else
        $drift\_flag \leftarrow False$
        Stop and empty short window
    end if
end while

$$weight\_pre\_train = \sum_{T=0}^{N} W_{pt} T/N \tag{6.4}$$

Where N is number of the collected weights.

The calculation of std_pre_train is based on the $weight\_pre\_train$, which follows the Equation 6.5 below.

$$std\_pre\_train = \sqrt{(weight\_pre\_train) \times (1 - weight\_pre\_train)/N} \tag{6.5}$$

In lines 8–21, the long and time windows begin to collect upcoming weights from weight stream. Then the max_weight, max_std, min_weight, and min_std will be updated. Since the weight changes in two directions, increase or decrease, therefore the only group of update will be executed. Then the $potential\_drift\_flag$ will become $True$ if the pre-defined threshold is reached.

The principle of potential drift detection is based on the 3σ rule (Pukelsheim, 1994). The idea is 99.7% of the values lie within one standard deviation of the mean, therefore outside of this range is considered potential drift.

If the $potential\_drift\_flag$ equal to $True$, the short time window from lines 22 to 29 is activated. The indication of the true drift is validated if ten consecutive true $potential\_drift\_flag$ are collected. Otherwise, it will be considered false drift, shown in the Figure 6.3; meanwhile, the short time window will be cleared and wait to be reactivated.

In summary, the long time window will be activated with the entry of data, followed by the activation of the short time window when potential drift is detected. And short will only validate that this is a real drift if it receives 10 successive possible drift flags; otherwise, it will identify the potential drift as a false drift, clear itself, and wait to be triggered again.

6.3 Experiments

6.3.1 Datasets and Concept Drift Description

In this section, four artificial datasets and two real-world datasets are described. Then, proposed approach for constructing three types of concept drift with real-world data is presented.

Artificial Datasets

For the experiment and evaluation, four artificial datasets often used in drift detection literature are chosen: SEA, SINE, AGRAWAL(AG), and LED. They could be accessed on the public stream data mining platform (Montiel et al., 2018). Table 6.2 provides descriptions of the datasets.

Table 6.2: Description of Artificial and Real-world Datasets. 0-Abrupt concept drift, 1-Early abrupt concept drift, and 2-Incremental concept drift

Dataset	No. attributes	No. class	Drift types	Dataset type
SEA	3	2	0/1/2	Artificial
Sine	2	2	0/1/2	Artificial
AGRAWAL	9	2	0/1/2	Artificial
Mixed	3	2	0/1/2	Artificial
Posture	3	2	0/1/2	Real-world
Powersupply	2	2	0/1/2	Real-world

Real-world Datasets

Two real-world datasets are used for drift detection evaluation. Powersupply (Wu et al., 2014) and Posture (Kaluža et al., 2010), are two data streams collected in the real world with multiple sensors. The details shown below:

Powersuppply: It is the hourly powersupply recording of an Italian electrical provider that records power from two sources: the main grid and other networks that have been transformed. This stream contains records for the years 1995 to 1998. Each data point contains two attributes and one label, where each attribute corresponding to a source of powersupply and the label indicating the moment of collection as a 0 to 23 integer.

Posture: The posture dataset collects data from sensors on the left/right ankle, belt, and chest at 0.1s intervals. Each data instance is made up of three features and one class label, with the features indicating the spatial position and the class label denoting the kind of posture.

Concept drift can be generated with a specified start point and duration in artificial datasets. However, no such definition exists in real-world data. Consequently, we reorganise and rebuild the real-world data to have three types of concept drift: abrupt concept drift, early concept drift, and incremental concept drift.

To generate various types of concept drifts, we first extract each concept from the dataset, and then we construct concept drift based on decision boundary movements corresponding to posterior probability changes in the data stream. For example, in the powersupply dataset, 24 hours of recording correspond to 24 concepts; therefore, we filter the several most distinct concepts and use them to generate different types of concept drift.

Figure 6.4 shows the difference between abrupt drift and incremental drift measured by the dura-

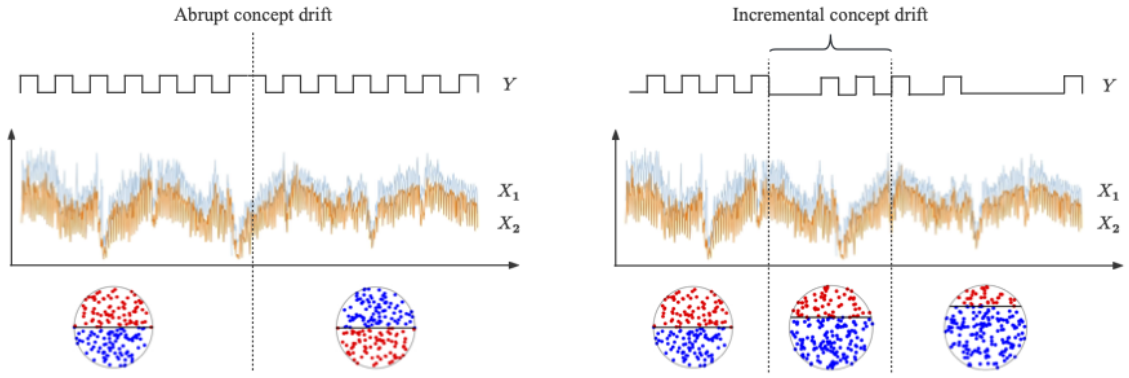


Figure 6.4: Different concepts in abrupt drift and incremental drift.

tion of the drift. The X_1 and X_2 represent the two features in the data stream, and it is clear that the data itself will remain unchanged before and after the drift. However, the class Y corresponding to the posterior probability of the data stream will have abrupt or incremental change after the drift occurs. The movement of the decision boundary also confirm this observation. The difference between these two drifts is the duration of the drift. We chose 180 degrees as an extreme example, although a lesser rotation drift is also reasonable depending on the circumstance. The movement of the decision boundary also reflects the pattern of drift change.

Three kinds of concept drift

For a more comprehensive evaluation on our methods, we employ three strategies for evaluating the performance of drift detection. Each dataset contains one of three different types of concept drift: abrupt concept drift, early abrupt concept drift, and incremental concept drift. Each data stream has one thousand time points. We set the concept drift with different start point p_s and duration d to generate distinct concept drift. Below are the drift start point and its duration:

Drift type	**Drift start point** (p_s)	**Duration** (d)
Abrupt drift	500	1
Early abrupt drift	100	1
Incremental drift	400	200

Table 6.3: Three different types of concept drift with drift point and duration (time point)

The abrupt and early abrupt concept drifts have the same duration, one time point, indicating that the concept will abruptly change to a different concept. The only difference is the drift start point. Comparatively, the later the drift occurs, the easier to be detected, as the detection method has more time to react. Therefore, the early abrupt concept drift is used to measure the responsiveness

to the concept drift. The incremental drift has a longer duration than the abrupt drift and early abrupt drift, thereby making it harder to identify due to its modest changes. These three drifts and the corresponding data stream changes are shown in Figure 6.5.

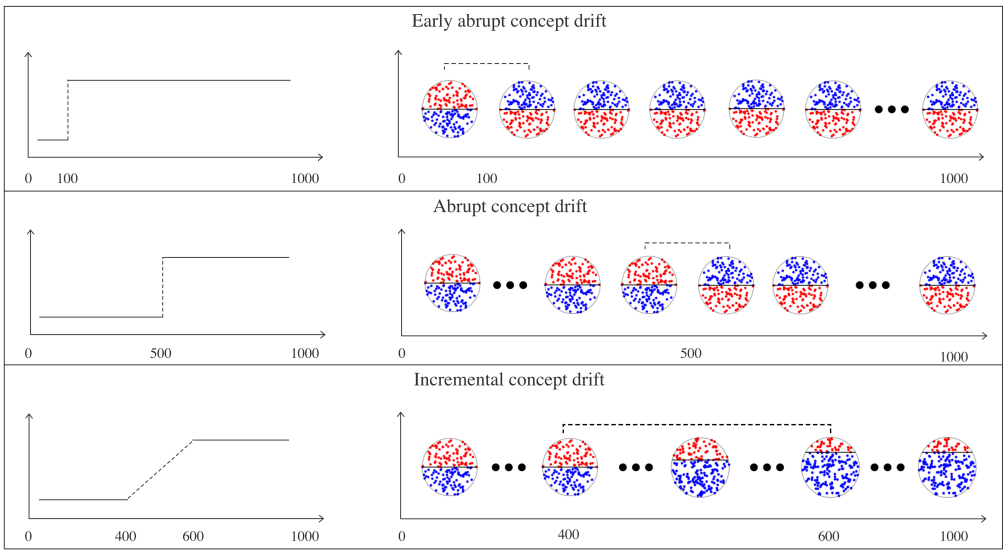


Figure 6.5: Three types of concept drift: abrupt concept drift, early abrupt concept drift and incremental concept drift. The left represents the concept change, and the right represents the decision boundary change caused by concept drift.

It is clear that both abrupt drift and early abrupt drift have dramatic changes in a short period of time, only differing in the drift start point. However, the change of the incremental drift takes longer, resulting in a gradual shift to a different concept.

6.3.2 Baseline Methods

In this section, we employ nine methods, of which six drift detection techniques are sourced from the *skmultiflow* library (Montiel et al., 2018). These methods include DDM, ADWIN, EDDM, PH, HDDM_A, and HDDM_W. Additionally, we utilize two methods from the *river* library (Montiel et al., 2021), namely FHDDM and MDDM. Lastly, OASW (Yang and Shami, 2021), a method similar to DDM, is publicly available on GitHub.

For each method, we employ the default parameters, with the exception of ADWIN, for which we modify the settings to enhance performance. Table 6.4 presents the specific values or thresholds. The warning and drift thresholds for DDM, HDDM, HDDM_A, and HDDM_W are denoted by α and β, respectively. In the case of ADWIN and PH, there is only a drift threshold, represented by

δ.

In HDDM_W, λ signifies the weight assigned to the data, with more recent data receiving less weight. For FHDDM, MDDM, and OASW, we employ a grid search strategy to determine the most suitable parameters, thereby achieving optimal performance in this section.

Table 6.4: Algorithm parameters or threshold description

Algorithms	**Parameters(Thresholds)**
DDM	$\alpha = 0.05, \beta = 0.01$
ADWIN	$\delta = 0.05$
EDDM	$\alpha = 0.9, \beta = 0.95$
PH	$\delta = 0.005$
HDDM_A	$\alpha = 0.005, \beta = 0.001$
HDDM_W	$\alpha = 0.005, \beta = 0.001, \lambda = 0.05$
FHDDM	$\delta = 0.01$
MDDM	$difference = 0.1, \delta = 0.01, \lambda = 0.05$
OASW	$\alpha = 0.85, \beta = 0.95$

6.3.3 Evaluation Metrics

The detection delay is often used to compare performance. (Liu et al., 2022). This work uses three metrics from (Wang et al., 2022) for evaluation; details are provided below.

- **Effective Detected Drift Rate (EDDR)**: the proportion of experimental trials where a detected drift point falls inside the Effective Detected Range (EDR).
- **Distance to First Effective Detected Point (FEDP)**: to measure how quickly the drift point could be detected.
- **Distance to Last Detected Point (LDP)**: determine if the drift detection method could be halted in time after real drift has been detected.

The EDR is defined by a time interval that indicates the acceptable time interval for detected drift point collection. Only drift points detected within EDR are considered effective. This section defines three groups of EDR for each type of concept drift in response to varying circumstances.

EDR: abrupt drift [400, 600] early abrupt drift [50, 150] and incremental drift [300, 500].

6.3.4 Experiment Setup

Our method was implemented using $Python$ 3.7.10. We established the neural network structure using the $PyTorch$ 1.10.0 package. The functions we used to build the networks depicted in the Table 6.5:

Table 6.5: Function used to construction of the neural network

Layer	**PyTorch layer**
Linear layers	torch.nn.Linear
Recurrent layers	torch.nn.GRU
Non-linear layers	torch.nn.Tanh
Activate layer	torch.nn.Sigmoid
optimizer	torch.optim.SGD

Throughout network construction and implementation, simple network structures and functions will be utilised in order to keep our contribution more focused on the framework. The MCDD-FCN network structure is composed of linear layers, nonlinear layers (optional), and an active layer. Utilizing recurrent and activate layers, the MCDD-RNN structure network is constructed. Stochastic gradient descent (SGD), which is accessible in $PyTorch$, used as the optimizer approach for both structures, other solutions are also acceptable. The loss function used is Binary Cross Entropy Loss from PyTorch Library.

The experiment hyper-parameters consist of first two stages of the proposed framework, which correspond to the two phases of model-centric drift detection. The details of the hyper-parameters are listed in the Table 6.6.

Table 6.6: Hyper-parameters used for proposed framework

	Pre-training	**Detection**
Learning Rate	0.01(MCDD-FCN) 0.1 (MCDD-RNN)	1e-4
Step size	200	10
Epoch	3	1
Decay rate	0.9	1.0

The only difference between the hyper-parameters is the learning rate; experiments indicate that this setting delivers more consistent results. By optimizing the hyper-parameter settings for pre-training, the efficiency of the pre-training process can be enhanced. And for the detection phase,

the slower learning rate will keep the weight update of the neural network stable until the concept drift occurs. The detection phase's epoch is set to 1 to meet with the data stream learning criterion one-pass (Gu et al., 2021), that data can only be used once before being discarded.

6.3.5 Comparison Results on Benchmark Datasets

The experiments aims to validate the performance of the framework in comparison with the baseline methods on three types of concept drift as in section 6.3.1. Section 6.3.5 is to verify whether our proposed framework is compatible with different neural network models. For each kind of concept drift, we run 100 trails, the metrics to evaluate the result shown in section 6.3.3. The experiments and discussion with metrics EDDR shown in section 6.3.5. Then the remaining two metrics FEDP and LDP will be evaluated in section 6.3.5 Abrupt Drift Detection, section 6.3.5 Early Abrupt Drift Detection and Incremental Concept Drift, section 6.3.5. We also discussed the Parameter Adjustability of the framework for different neural network structures.

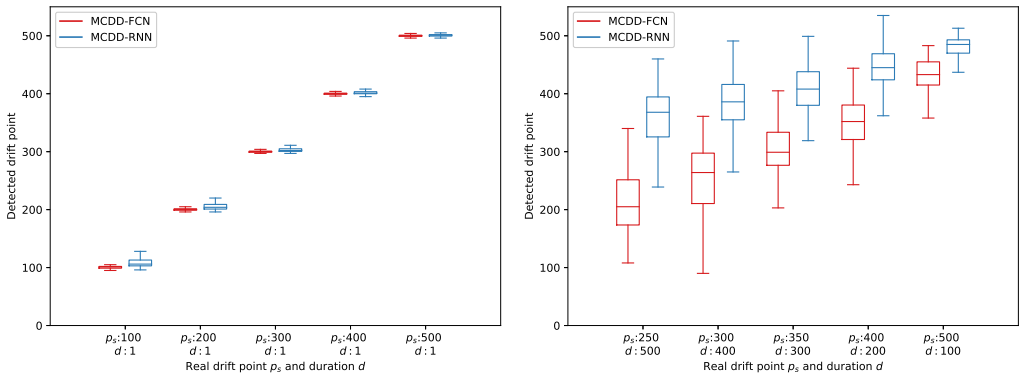
Feasibility verification of MCDD

Our proposed framework could detect concept drift by monitoring the weight change of the deep neural network. Neural network structures exist in various forms and are effective for different types of machine learning problems. As a consequence, our proposed framework is compatible with various neural network structures. In this section, we adopt the widely-used neural network structures MCDD-FCN (fully connected network) and MCDD-RNN (recurrent neural network). Both network structures include three hidden layers, with the first two layers being linear (MCDD-FCN) and recurrent (MCDD-RNN), and the third layer being linear for both structures. Table 7.2 displays the hyper-parameters implemented in two structures.

The MCDD-FCN and MCDD-RNN are evaluated on datasets containing abrupt and incremental concept drift, with different drift start point and duration. The result shown in Figure 6.6a and Figure 6.6b. From the result, it is clear that both are capable of detecting concept drift. However, these two methods respond differently to different concept drifts.

Our two methods produce similar results, as shown in Figure 6.6a. However, MCDD-FCN maintains a higher concentration of detected drift points than MCDD-RNN. Inter-quartile range (IQR) for MCDD-RNN is relatively larger at (p_s=100, d=1) and (p_s=200, d=1) on the boxplot, where the

higher the IQR, the more discrete the detected drift points.



(a) Result on various abrupt drifts

(b) Result on various Incremental drifts

Figure 6.6: Result of feasibility verification experiments of the framework on abrupt and Incremental drift (p_s: drift start point, d: duration of drift)

As shown in Figure 6.6b, incremental concept drift leads in a greater difference between MCDD-FCN and MCDD-RNN than abrupt drift. Overall, both methods could detect drift and generate drift points surrounding the real drift point. However, the MCDD-RNN detects drift later than the MCDD-FCN on five different sets of concept drift. On the other hand, the shorter the duration of the drift, the smaller the difference between the drift detection results on our two methods, indicating that the duration of the drift will affect the drift detection capability.

Overall, the findings of the MCDD-FCN and MCDD-RNN suggest that the proposed framework can successfully detect various types of concept drift; however, the detection performance may be affected by the neural network structure employed.

Evaluation on EDDR

Firstly, we compare the EDDR to investigate how the drift detection systems respond to various types of concept drifts within the EDR, the results are presented in Table 6.7. Overall, our proposed method has higher EDDR scores than all other methods, which are 93.61% and 95.83%. However, the EDDR on each type of drift is quite different. It is clear that the majority of methods have the best performance on abrupt drift. This is because abrupt drift generates a dramatic change in the properties of the data over a short amount of time, making it simpler to detect. The majority of methods perform poorly when dealing with incremental drift because its duration is longer than that of abrupt drift, making it more difficult to detect.

Table 6.7: EDDR result on 6 datasets from 11 methods, the unit is time point.The higher the EDDR value, the better the performance of detection method. Our two methods have the highest average EDDR value among other methods, and also the highest on each drift.

	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W	FHDDM	MDDM	OASW	MCDD-RNN	MCDD-FCN
AG_0	99	22	35	92	100	43	92	93	97	100	100
AG_1	97	0	43	0	70	13	94	91	94	100	100
AG_2	44	0	14	0	6	16	0	0	47	96	89
Mixed_0	100	65	54	100	100	100	93	93	100	100	98
Mixed_1	99	0	97	41	98	100	98	98	0	100	100
Mixed_2	78	0	1	9	53	67	19	22	92	97	86
Sea_0	67	0	93	0	5	49	39	41	53	100	92
Sea_1	28	0	1	0	1	42	31	29	0	56	85
Sea_2	6	0	41	0	0	2	1	1	0	57	89
Sine_0	98	95	57	100	100	100	100	100	100	100	100
Sine_1	100	0	99	94	99	100	100	100	0	100	100
Sine_2	86	0	1	18	66	83	39	43	96	99	95
posture_0	92	3	8	53	100	37	0	0	62	96	100
posture_1	99	0	84	0	99	22	0	0	0	100	100
Posture_2	69	1	100	0	74	99	1	1	80	96	94
powersupply_0	98	40	63	100	100	100	100	100	100	100	100
powersupply_1	100	0	99	0	99	100	100	100	0	95	97
Powersupply_2	96	2	100	0	97	100	76	76	100	93	100
Average_0	92.33	37.5	51.67	74.17	84.17	71.50	70.66	71.17	85.33	99.33	98.33
Average_1	87.17	0	70.50	22.5	77.67	62.83	70.50	69.67	31.50	91.83	97.00
Average_2	63.16	0.50	42.83	4.50	49.33	61.16	22.67	23.83	69.16	89.66	92.16
Average	80.89	12.67	55.00	33.72	70.39	65.17	54.61	54.89	62.00	**93.61**	**95.83**

Similarly, this phenomena is present in the single dataset. In the AG, Mixed, and Sea datasets, for instance, all drift detection methods have a higher EDDR for abrupt and early abrupt drift, indicating that incremental drift is more complex and challenging to detect. Our proposed method exhibits consistent detection performance across all three drift types. It achieves nearly identical results for abrupt and early drift. For incremental drift, the detection performance is slightly lower but still robust, with 89.66% and 92.16% for incremental and abrupt/early drifts, respectively.

Wilcoxon-holm signed-rank method with Holm's alpha (5%) is used to compare the pairwise significance of the EDDR score. To visualise this type of comparison, we drawn a Wilcoxon critical difference diagram (Ismail Fawaz et al., 2019). The result depicted in Figure 6.7. Our two methods are ranked first and second, which demonstrates that the performance of MCDD is superior to that of others. Moreover, the scores of our two methods, 3.25 and 3.44 respectively, are extremely similar. Furthermore, this proves that the efficiency of the framework is not primarily dependent on the deployed network structure, since it is capable of successfully detecting concept

drift.

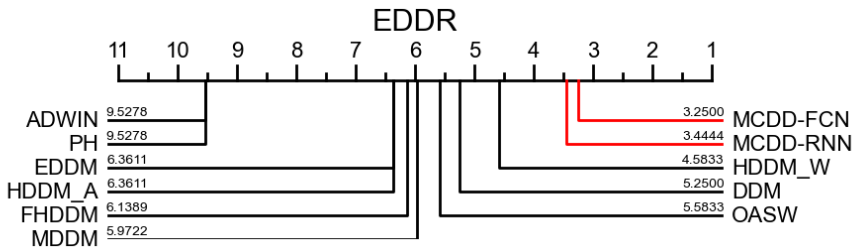


Figure 6.7: Comparison of average rank (lower is better) of methods w.r.t. performance across datasets. MCDD-RNN and MCDD-FCN outperforms all methods.

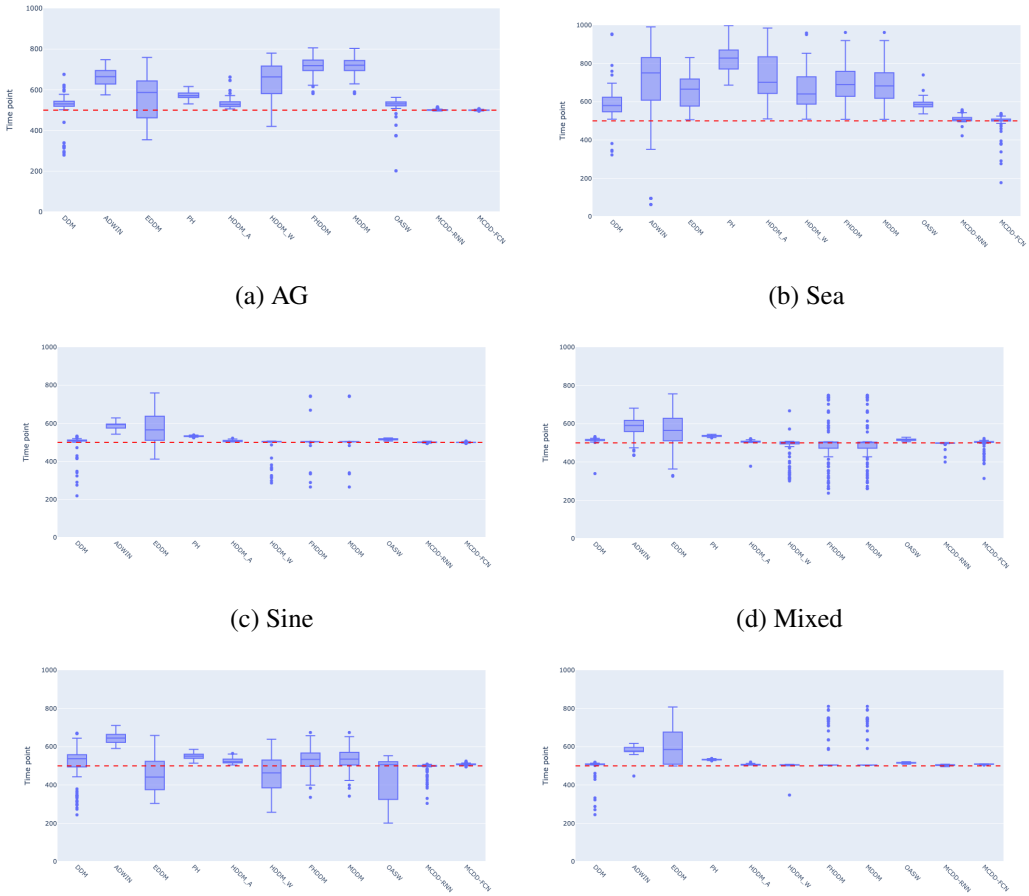
Abrupt Drift

From the evaluation of EDDR, it is clear that the results on abrupt drift are best for majority methods. It is evident from the examination of EDDR that the findings for abrupt drift are superior to those for the other two types of drift. Indeed, abrupt drift changes more rapidly and visibly than other types of drift, and it occurs in the middle of the data stream, giving detection methods more time to react and preventing false reports because the small amount of data.

Figure 6.8 shows the results of the drift points found by nine methods across six datasets. The y axis indicates the distribution of detected drift points, while the concept drift occurs at real drift point 500. According to the boxplots, the majority of detected drift points for each detection method concentrate around point 500. Particularly in datasets AG, Sine, and Powersupply. However, our framework still detect false reports in other datasets that deviate significantly from the ground-truth drift point, e.g. results from dataset Mixed and Posture in Figure 6.8.

For a more in-depth comparison of the results, the evaluation metrics FEDP and LDP defined in Section 6.3.3 are used. From Table 6.8 and 6.9, the average value of the deviation between the detected drift point and real drift point is 6.39 and 5.88, respectively, which is smaller than other methods. Our method is clearly able to detect the drift early.

In addition, we may observe that the FEDP and LDP for the same dataset are typically not equal. It is because a method sometimes detect more than one concept drift point. For example, a method could detect three drift points [300,500,700] with the condition that the real drift point is at 500. However, our method have equal FEDP and LDP values for abrupt concept drift. For example FEDP and LDP of MCDD-RNN on AG data are both equal to 1.43, as is the case with MCDD-



(a) AG (b) Sea

(c) Sine (d) Mixed

(e) Posture (f) Powersupply

Figure 6.8: Abrupt drift detection result on 6 datasets, it is apparent that our proposed method (last two in each subfigure) has the most concentrated detected concept drifts near the real drift point at 500 (red dashed line), whereas other methods perform well but are rather discrete.

FCN. It denotes that a concept drift is precisely detected in a data stream with a single abrupt drift. This indicates that our method is capable of terminating the detection process once real drift is detected.

In general, our method works best with abrupt drift among these drift detection methods. On the one hand, the deviation between the detected point and the real point demonstrates that our method has the quickest detection capability. Furthermore, our method is able to precisely detect an exact drift and stop the detection process in time, effectively reducing the number of false drift reports.

Table 6.8: FEDP on abrupt drift

	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W	FHDDM	MDDM	OASW	MCDD-RNN	MCDD-FCN
AG	1.18	74.36	190.81	73.11	30.35	84.67	46.65	43.66	23.32	1.43	0.23
Mixed	11.52	37.00	93.02	36.35	4.78	9.32	91.24	83.07	16.80	3.12	4.89
Sea	70.68	89.00	32.57	nan	232.42	105.05	117.03	117.16	88.10	10.69	9.44
Sine	19.94	38.84	36.65	32.41	7.14	28.59	12.77	8.47	16.04	7.00	3.00
Posture	39.57	102.20	353.73	97.16	25.40	301.26	404.96	398.95	73.16	12.38	8.75
Powersupply	15.67	43.00	6.14	57.65	6.42	1.86	3.93	3.93	16.49	3.70	8.96
Average	26.43	64.07	118.82	59.33	51.09	88.46	112.76	109.20	38.99	**6.39**	**5.88**

Table 6.9: LDP on abrupt drift

	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W	FHDDM	MDDM	OASW	MCDD-RNN	MCDD-FCN
AG	40.89	268.28	248.06	73.11	37.42	375.31	455.25	457.39	23.32	1.43	0.23
Mixed	14.93	183.16	190.84	36.35	7.34	9.32	93.05	79.25	16.80	3.12	4.89
Sea	108.16	89.00	276.98	Nan	232.43	221.41	274.53	266.48	88.10	10.69	9.44
Sine	9.92	149.56	185.64	32.41	7.14	4.20	16.24	12.92	16.04	7.00	3.00
Posture	49.84	323.64	185.97	97.16	26.55	229.38	440.56	438.24	73.16	12.38	8.75
Powersupply	8.44	140.60	185.06	57.65	6.42	5.01	81.52	84.55	16.49	3.70	8.96
Average	38.70	192.37	212.09	59.34	52.88	140.77	226.86	223.13	38.98	**6.39**	**5.88**

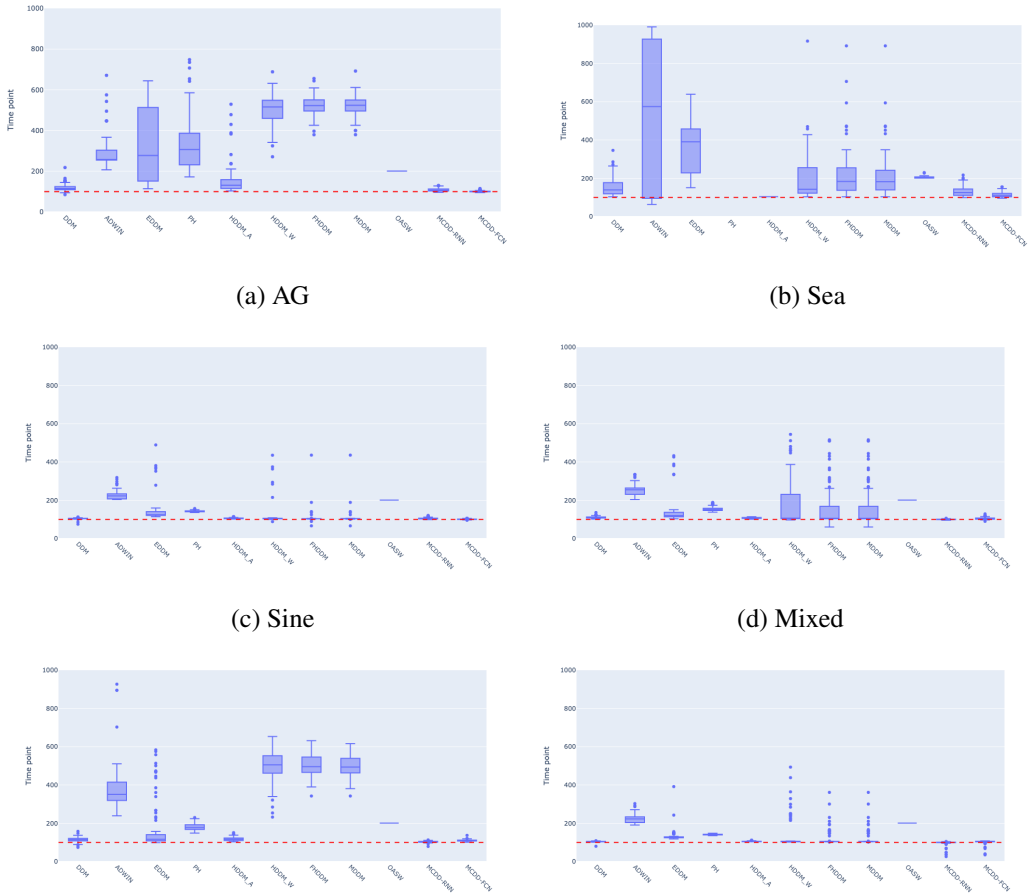
Early Abrupt Concept Drift

Early abrupt concept drift has a similar drift duration to that of an abrupt drift, but it starts much earlier. This requires the drift detection method to use less reaction time for drift detection. Figure 6.9 reflects the difficulty of detection with higher discrete distribution of detected drift points. However, our proposed method still perform the best among all the drift detection methods. We also notice that the degree of concentration of detected concept drifts is less than abrupt drift.

In addition, FEDP and LDP scores from the Tables 6.10 and 6.11 are 8.66 and 5.31, respectively. The lowest values suggest that our method has the best performance in detecting early abrupt concept drift. Moreover, the fact that the FEDP and LDP had the same value shows that our method could precisely detect an exact drift point.

Incremental Concept Drift

According to the EDDR on Table 6.7, incremental concept drift is the most complicated and difficult to be precisely detect across all detection methods. This increased difficulty is attributed to the typically longer duration of incremental concept drift, compared to abrupt and early abrupt drifts. Figure 6.10 demonstrates that the detected drift points for incremental drift lack a distinct



(a) AG (b) Sea

(c) Sine (d) Mixed

(e) Posture (f) Powersupply

Figure 6.9: Early abrupt drift detection result on 6 datasets, it is apparent that our proposed method (last two in each subfigure) has the most concentrated detected concept drifts near the real drift point at 100 (red dashed line). However, the drift points detected by other methods are relatively discrete.

Table 6.10: FEDP on early abrupt drift

	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W	FHDDM	MDDM	OASW	MCDD-RNN	MCDD-FCN
AG	17.21	170.18	37.66	234.38	43.47	9.63	9.33	9.80	101.00	8.20	1.07
Mixed	10.65	59.64	14.97	54.75	7.25	13.92	2.94	1.50	101.00	0.44	4.47
Sea	46.91	619.00	137.25	Nan	4.00	69.61	100.68	93.25	105.27	31.24	14.37
Sine	4.15	39.16	22.07	42.31	5.63	3.61	2.51	2.82	101.00	5.42	1.02
Posture	13.04	335.45	10.92	Nan	17.73	2.28	19.08	18.71	101.00	2.81	9.62
Powersupply	3.62	83.32	24.83	80.48	5.14	4.47	4.16	4.16	101.00	3.82	1.32
Average	15.93	217.79	41.28	102.98	13.87	17.25	23.11	21.71	101.71	**8.66**	**5.31**

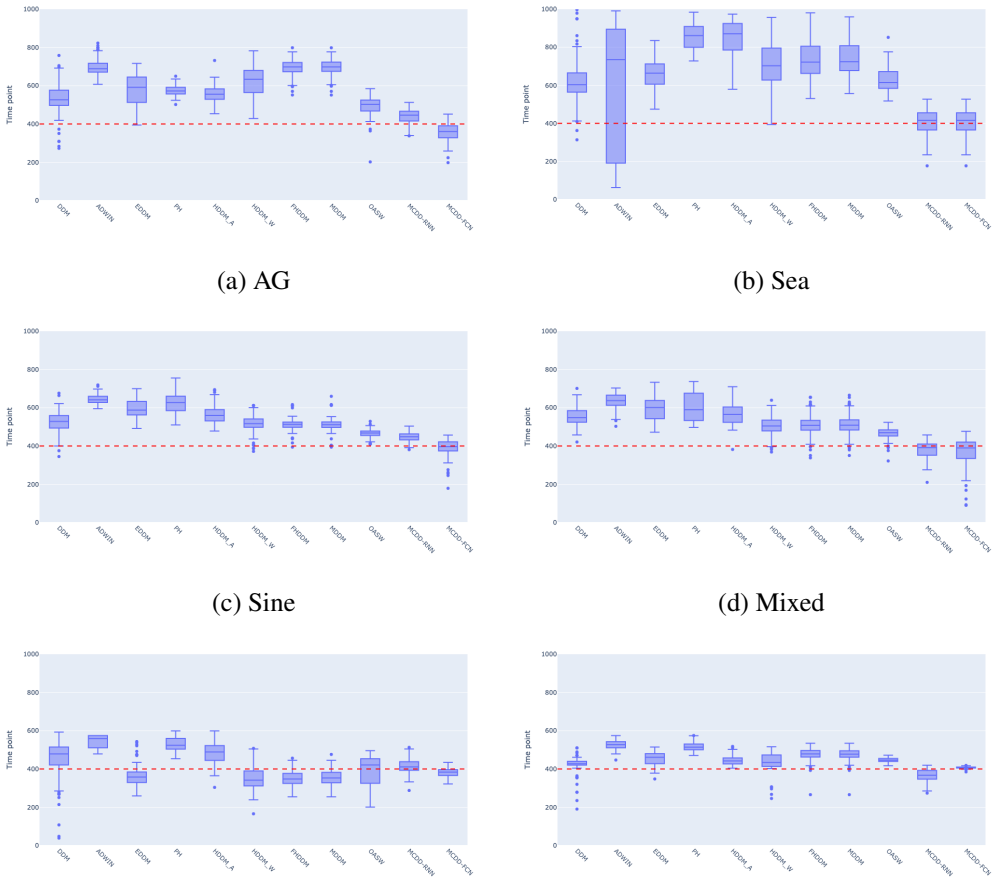
and consistent trend. And the degree of concentration of drift points detected by all detection methods is significantly lower than the abrupt and early abrupt drift.

Our method can still provide higher detection performance compared to other methods. Based on

Table 6.11: LDP on early abrupt drift

	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W	FHDDM	MDDM	OASW	MCDD-RNN	MCDD-FCN
AG	21.30	213.67	416.02	243.12	66.60	765.54	840.94	841.94	101.00	8.20	1.07
Mixed	10.65	276.60	72.30	54.75	7.25	13.92	116.03	116.53	101.00	0.44	4.47
Sea	62.63	619.00	380.34	Nan	4.00	133.56	155.96	153.31	105.27	31.24	14.37
Sine	4.85	235.00	73.40	42.31	5.63	31.12	13.68	13.68	101.00	5.42	1.02
Posture	16.87	492.83	145.61	Nan	17.73	757.46	834.50	833.45	101.00	2.81	9.62
Powersupply	4.47	279.80	44.72	80.48	5.14	5.01	31.44	31.40	101.00	3.82	1.32
Average	20.13	352.82	188.73	105.17	17.73	284.44	332.09	331.72	101.71	**8.66**	**5.31**

the detected points in Figure 6.10 , we can see that the points our method detected are closest to the real drift point, and the trend is relatively concentrated. But it is also evident that our method also produces many detected drift points that deviate from the real drift point.



(a) AG (b) Sea

(c) Sine (d) Mixed

(e) Posture (f) powersupply

Figure 6.10: Incremental drift detection result on 6 datasets. Our proposed method (last two in each subfigure) has the most concentrated detected concept drifts near the real drift point at 100 (red dashed line).

The values of the FEDP and LDP from Table 6.12 and 6.13 also reflect the complexity of the

Incremental drift, our method has higher delay than abrupt and early abrupt drift, with 40.56 and 19.66, respectively, for the FEDP and LDP. The similar tendency applies to the other methods, but the magnitude of the change is significantly greater.

Table 6.12: FEDP on incremental concept drift

	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W	FHDDM	MDDM	OASW	MCDD-RNN	MCDD-FCN
AG	71.78	193.24	104.37	173.01	154.64	17.49	19.81	24.00	94.92	41.59	45.46
Mixed	71.01	63.96	44.42	123.39	91.68	89.33	35.31	25.37	65.48	19.38	31.44
Sea	198.99	97.00	113.77	Nan	444.73	265.94	267.64	276.16	233.43	86.35	6.09
Sine	21.06	119.96	5.54	112.00	89.07	15.41	28.80	38.85	64.76	46.19	9.49
Posture	3.29	162.20	222.44	191.75	77.71	186.97	294.03	291.29	11.00	14.67	17.92
Powersupply	11.81	146.01	3.75	187.93	46.54	6.28	4.57	5.32	47.13	35.20	7.57
Average	62.99	130.40	82.38	157.62	150.73	96.90	108.36	110.16	86.12	**40.56**	**19.66**

Table 6.13: LDP on incremental concept drift

	DDM	ADWIN	EDDM	PH	HDDM_A	HDDM_W	FHDDM	MDDM	OASW	MCDD-RNN	MCDD-FCN
AG	191.80	430.68	404.89	175.63	165.19	482.54	555.97	558.13	94.92	41.59	45.46
Mixed	241.74	427.16	402.21	278.02	247.24	189.36	242.76	237.31	65.48	19.38	31.43
Sea	240.14	97.00	400.52	Nan	444.73	353.64	398.36	397.73	233.43	86.35	6.09
Sine	242.09	438.68	385.63	328.52	238.6	216.01	197.86	191.26	64.76	46.19	9.49
Posture	81.24	162.20	114.54	191.75	92.80	79.68	156.00	158.57	11.00	14.67	17.92
Powersupply	33.90	158.45	108.42	187.93	46.54	70.42	144.14	143.77	47.13	35.20	7.57
Average	171.82	285.70	302.70	232.37	205.85	231.94	282.51	281.12	86.12	**40.56**	**19.66**

Parameter Adjustability

From the above experiments, it is clear that MCDD has the best drift detection performance among all methods. However, we have observed that our two methods sometimes perform inconsistently on the same dataset . In Table 6.13, the dataset Sea with Incremental drift, for instance, MCDD-RNN has a delay of 86.35 time points, but MCDD-FCN has a delay of just 6.09 time points. This demonstrates that our proposed framework need to be adjusted to have better performance; thus, we can benefit from modifying the hyper-parameters.

To reduce detection delay, we may raise the decay rate, which is set to 1.0 by default, so that we can detect drift sooner. From the Figure 6.11, it is clear that the decay rate of 1.02 gives the best results, with just 11.4 time point delays, a significant improvement over previous result of 86.35 (coloured by orange in Table 6.13). Furthermore, neither large nor small decay rates are effective.

After the parameter adjustment, we re-compare the detection performance using Wilcoxon-Holm

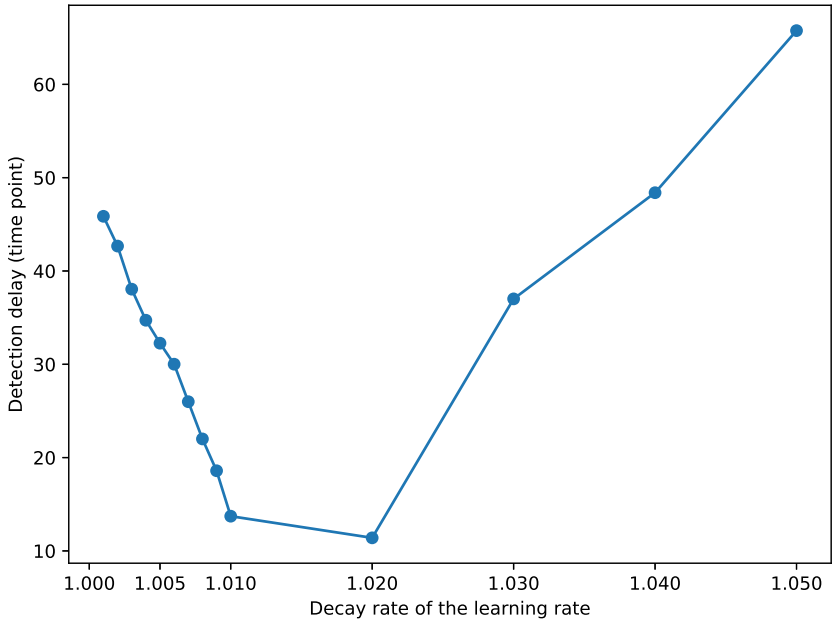


Figure 6.11: MCDD-FCN and MCDD-RNN drift detection result on datasets with incremental concept drift after adjusting parameter

signed rank with the updated EDDR table. MCDD continues to rank in the top two, with scores 3.33 and 3.41 in Figure 6.12. It demonstrates that MCDD is very adaptable and that various datasets favour different network structures.

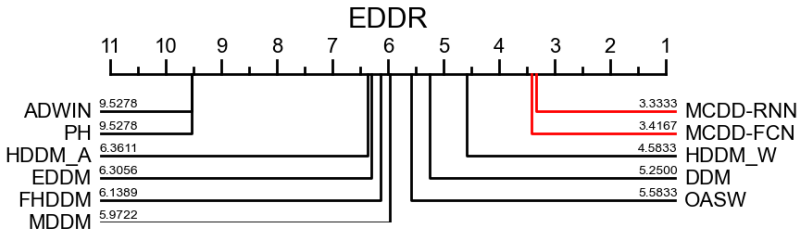


Figure 6.12: Comparison of average rank (lower is better) of methods w.r.t. performance across datasets. MCDD-RNN and MCDD-FCN based method have very close performance and outperform other methods.

MCDD's Framework Robustness

This section presents an evaluation of the robustness of our MCDD framework in relation to data bias and quality issues.

Data bias, in the context of this section, is primarily characterized by data class imbalance. Artificial datasets used in this section are generated with varying class distributions. The most severe cases exhibiting a 30% to 70% split in Powersupply dataset with incremental concept drift. The

details of data class imbalance as illustrated in Table 6.14. The values tabulated represent the average proportion of instances that belong to Class 0 across these data streams. Due to the binary nature of the classification employed in this study, the proportion of instances in Class 1 can be inferred as (1 - Class 0). This table illustrates the extent of data class imbalance present in the datasets.

Table 6.14: Details of Data Class Imbalance (Expressed as the Percentage of Instances in Class 0): This study comprises multiple datasets, each with a distinct type of concept drift and containing 100 data streams.

	AG	Sea	Sine	Mixed	Posture	Powersupply
Abrupt	44.46 ± 1.13	32.54 ± 1.82	50.00 ± 0.05	51.76 ± 5.40	50.03 ± 1.20	50.13 ± 1.45
Early Abrupt	39.58 ± 1.51	43.42 ± 1.57	53.02 ± 1.54	49.50 ± 4.97	47.82 ± 1.58	50.06 ± 1.50
Incremental	44.48 ± 1.44	32.55 ± 1.80	49.97 ± 0.15	49.57 ± 4.92	29.90 ± 1.96	30.01 ± 1.12

To further assess the MCDD method's robustness towards imbalanced data classes, we designate datasets without concept drift as class 0, and those with concept drift as class 1, each group comprising 100 data streams. This arrangement facilitates concept drift detection using MCDD, yielding Receiver Operating Characteristic (ROC) curves that demonstrate the efficacy of our approach in addressing data bias, as shown in Figure 6.13. Each curve corresponds to a distinctive type of concept drift—abrupt, early abrupt, and incremental—with respective AUC scores of 0.87, 0.90, and 0.98. These scores underscore the efficacy of MCDD in identifying concept drift even amidst class imbalances. The AUC values nearing unity further attest to the robustness of the MCDD framework when faced with such data complexities.

As demonstrated in the associated figure, the results from all three types of concept drift yield impressive ROC curves, accompanied by high Area Under the Curve (AUC) scores. These results underscore the ability of our framework to maintain high accuracy, even in the presence of data class imbalance issues. Notably, abrupt drift exhibits the highest AUC score, followed by early abrupt drift, and finally incremental drift. This pattern suggests that the difficulty of drift detection is influenced by the type of concept drift. Crucially, however, the AUC scores appear to remain unaffected by data bias, implying a robust resilience of the MCDD framework to data bias.

We also take into consideration the implications of data quality. For our artificial datasets, we resort to the *skmultiflow* library's default parameters—$noise\_percentage$, $perturbation$, and has_noise—to manipulate the noise types, either attribute noise or class noise (Wang et al., 2022). On the other hand, real-world datasets, which are gathered through sensors in various environ-

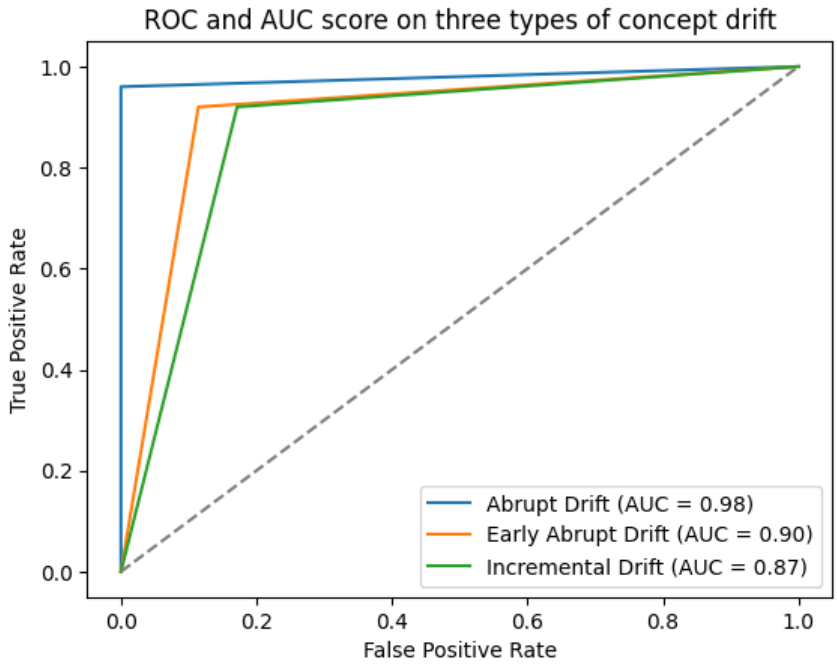


Figure 6.13: ROC Curves for MCDD Performance under Different Concept Drift Scenarios with Class Imbalance.

ments, naturally incorporate unavoidable noise during the data collection process. A summary of the noise situations across all datasets can be found in Table 6.15. This table presents the data quality details of six datasets used in the study. Each row corresponds to a distinct dataset: AG, Sea and Sine represent artificial datasets with controlled noise, while Posture, and Powersupply are real-world datasets with inherent noise complexities. For each dataset, the presence or absence of noise is indicated along with the type of noise (attribute, class, or hybrid) and the specific parameter controlling the noise level, if applicable.

Table 6.15: Data Quality and Noise Characteristics of Artificial and Real-World Datasets.

	Noise	Noise Type	Parameter
AG	Yes	Attribute Noise	Perturbation
Sea	Yes	Class Noise	Noise_Percentage
Sine	Yes	Attribute Noise	Has_Noise
Mixed	No	N/A	N/A
Posture	Yes	Hybrid	Unknown
Powersupply	Yes	Hybrid	Unknown

After examining the EDDR results from Table 6.7 and the ROC curves in Figure 6.13, it becomes evident that our MCDD method exhibits robustness against both data bias and data quality issues. Specifically, our framework maintains consistent performance across varying degrees of data label

imbalance and noise. In conclusion, our method exhibits reliable performance in the face of diverse data bias and quality challenges.

6.4 Summary

To accurately and precisely detect the concept drift in data stream, we proposed a Model-Centric transfer learning framework for Concept drift Detection (MCDD). Our framework is based on the model-centric principle, in which the model adapts to a new concept before its output changes. Model adaptation is reflected in model structure changes, such as changes in the decision boundary of linear regression or weight changes in deep neural networks. Firstly, the initialized deep neural network will be trained and evaluated on base data without concept drift, and meanwhile another structurally equal network will be partially frozen. Secondly, the partially frozen network will combine with the weights from pre-trained network, and just perform training mode in the data with concept drift. It significantly accelerates the concept drift detection process and reduces the computational cost with transfer learning. Lastly, the long and short time windows collects the weights from last hidden layer of the transferred network. In the end, the real drift is detected while the false drift is filtered out. The experiment results on artificial and real-world dataset demonstrate superior performance of our proposed framework. In addition, the effectiveness of MCDD is independent of the type of used deep neural network. The experiment of parameter adjustability also proves that MCDD is highly adaptive to improve its performance. We have also demonstrated the framework's robustness against data bias, specifically class imbalance, and various degrees of data quality issues. The consistent and statistically significant performance across diverse scenarios underscores MCDD's potential as a reliable tool for addressing real-world data complexities. In summary, MCDD can detect concept drift and reduce delay more effectively than other detection methods.

Chapter 7

Concept Drift Understanding by Quadruple-based Approach: QuadCDD

Concept drift is a prevalent phenomenon in data streams that necessitates detection and in-depth understanding, as it signifies that the statistical properties of a target variable, which the model aims to predict, change over time in unforeseen ways. Existing detection methods predominantly aim to identify the drift start time, which lack comprehensive understanding of data streams, leading to a loss of drift information. In this work, we present a novel Quadruple-based Approach for Understanding Concept Drift (QuadCDD) framework in Data Streams that not only detects and predicts the concept drift start point but also offers a more detailed analysis of concept drift through the use of quadruples, encompassing drift start, drift end, drift severity, and drift type. Our framework employs quadruples to enable informed decision-making and adopt appropriate actions to handle various concept drifts, effectively maintaining high and stable performance in data streams with concept drift. Experimental results validate the effectiveness of our QuadCDD framework in accurately detecting and understanding concept drifts, as well as in preserving the stability and performance of models in the presence of these drifts.

7.1 Introduction

In the era of big data, data streams have become increasingly prevalent, with a vast amount of real-time data being generated by sources such as social media, online stores, sensor networks, and financial markets. These data streams provide valuable insights for making informed decisions and driving business intelligence. However, the dynamic nature of data streams poses challenges for predictive modeling, as the statistical properties of the target variables may change over time, resulting in concept drift. Concept drift signifies that the distribution of the data, which the model aims to predict, changes in unforeseen ways, thereby affecting the model's accuracy and stability.

In data streams, concept drift can manifest in two distinct forms. A data stream is typically composed of a sequence of samples, denoted as $H_{1,n} = [h_1, \cdots, h_n]$, where each sample $h_i = (X_i, y_i)$ consists of a d-dimensional feature vector X_i and a class label y_i. The joint distribution of the data stream is represented as $D_{1,t}(X, y)$. Concept drift occurs at time point $t + 1$ if $D_t(X, y) \neq D_{t+1}(X, y)$, which can be expressed as $\exists t, P_t(X, y) \neq P_{t+1}(X, y)$ (Lu et al., 2016) .

The joint probability $P_t(X, y)$ can be decomposed into $P_t(X) \times P_t(y|X)$. Based on Bayesian theory, two types of concept drift can be identified (Souza et al., 2020): real concept drift, characterized by a change in the posterior probability ($P_t(y|X) \neq P_{t+1}(y|X)$ while $P_t(X) = P_{t+1}(X)$), and virtual concept drift, characterized by a change in the marginal probability ($D_t(X) \neq D_{t+1}(X)$ while $P_t(y|X) = P_{t+1}(y|X)$). Most existing research focuses on concept drift as a change in the posterior probability, as it implies a shift in the decision boundary (Lu et al., 2019) (Gama et al., 2014) (Wang et al., 2018). In this work, our primary objective is to detect real concept drift in data streams. And four types are classified according to the rate of change in the data distribution. That is incremental drift, abrupt drift, recurring drift, and gradual drift, which are shown in Figure 7.1.

Existing drift detection methods predominantly focus on identifying the drift start time, often leading to a loss of drift information and an inadequate understanding of the data streams. This lack of comprehensive understanding hinders the ability to effectively handle concept drift and maintain high and stable performance in data streams with concept drift. Furthermore, most existing methods do not provide information about the drift end, drift severity, or drift type, which are crucial for

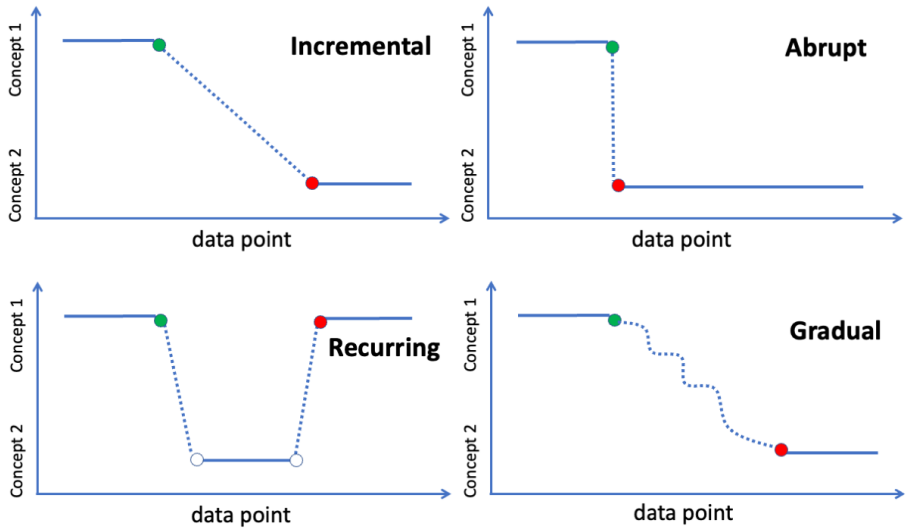


Figure 7.1: Illustration of Quadruple with Distinct Types of Concept Drift in Multiple Data Streams. The green and red dots represent the start and end of concept drift, respectively. The x-axis represents data points in a time series format, while the y-axis denotes two different concepts. The dashed line represents the concept change, with the type varying based on the duration and severity of the concept drift.

understanding the entire drift process and adopting appropriate actions to address the drift.

To address these limitations, we propose a novel Quadruple-based Approach for Understanding Concept Drift in Data Streams (QuadCDD) framework that not only detects and predicts the concept drift start point but also offers a more detailed analysis of concept drift by employing quadruples. These quadruples encompass drift start, drift end, drift severity, and drift type, providing a comprehensive understanding of the phenomenon. Our proposed framework leverages quadruple information to support informed decision-making and enable the adoption of appropriate actions to effectively handle various concept drifts, ultimately maintaining high and stable performance in data streams with concept drift.

The proposed QuadCDD framework is designed to be flexible and adaptable, capable of working with a wide range of data stream scenarios and predictive models. By providing a detailed analysis of concept drift and facilitating targeted action implementation, our approach aims to address the challenges posed by the dynamic nature of data streams, enabling organizations to derive valuable insights and make informed decisions based on their real-time data.

Our contributions are:

1. Developing a novel Quadruple-based Approach for Understanding Concept Drift in Data

Streams (QuadCDD) that comprehensively captures various aspects of concept drift, including drift start, drift end, drift severity, and drift type, enhancing the understanding of the phenomenon.

2. Providing a framework of quantitative analysis for QuadCDD: The QuadCDD framework utilizes quadruples to provide a detailed examination of concept drift, offering valuable insights into the underlying drift process.

3. Developing informed decision-making and targeted action implementation: The QuadCDD framework leverages quadruple information to support informed decision-making and enables the adoption of appropriate actions to effectively address different types of concept drifts.

4. Facilitating the QuadCDD with the maintenance of high performance and stability in data streams with concept drift: Experimental results validate the effectiveness of the QuadCDD framework in accurately detecting and understanding concept drifts while preserving the stability and performance of models in their presence.

The structure of the chapter is organized as follows: Section 7.2 presents the proposed QuadCDD framework. Section 7.3 discusses the experiments and their results. Lastly, Section 7.4 provides the conclusion along with a discussion on limitations.

7.2 Quadruple and QuadCDD Framework

In this section, we propose the Quadruple and the QuadCDD framework, which builds upon the Quadruple concept. The inspiration for the Quadruple comes from the multi-output capabilities of deep learning models, which can generate more than one output simultaneously. This approach enables a more comprehensive understanding of concepts. Subsequently, we introduce the QuadCDD framework in the following sections.

The Quadruple describe the comprehensive picture of the concept drift, iclude the start, end, severity and type of the concept, enable use to investigate more details on concept drift and make better decision to react to the concept drift.

The QuadCDD framework is designed based on Quadruple, which consists of four stages. First,

features from the pre-training data are extracted to capture the essential characteristics of the dataset with different concept drift. Next, an initial model is trained using the pre-training data, establishing a baseline for performance. Following this, the model undergoes fine-tuning with data from different domains, enhancing its adaptability and generalization. During the detection and decision-making phases, the Quadruple is generated and serves as input for decision-making. This process ensures that the machine learning model's performance remains consistent in the presence of concept drifts. By leveraging the deep neural network's multi-output capabilities, the QuadCDD framework can effectively manage concept drifts, maintaining both stability and adaptability in dynamic environments.

The Quadruple will be discussed in Section 7.2.1, and the QuadCDD framework is at Section 7.2.2.

7.2.1 Quadruple Representation of Concept Drift

In this section, we introduce the Quadruple representation, a comprehensive and adaptable method for characterizing concept drift in data streams. The Quadruple is defined as (D_s, D_e, D_v, D_t), which consists of four essential parameters: Drift **S**tart, Drift **E**nd, Drift Se**v**erity, and Drift **T**ype. This representation allows for effective real-time monitoring, analysis, and management of concept drift, ensuring the sustained accuracy and relevance of machine learning models in dynamic environments.

The four parameters are defined as follows:

- D_s: Drift start, indicating the point in the data stream where the concept drift begins.
- D_e: Drift end, signifying the point in the data stream where the concept drift concludes and stabilization occurs.
- D_v: Drift severity, quantifying the impact of the concept drift on the model's performance. This parameter can be adjusted according to the specific needs of various applications and contexts.
- D_t: Drift type, describing the nature of the concept drift, such as abrupt and incremental drift. This parameter can also be customized based on the particular requirements of diverse scenarios.

The quadruple is illustrated in Figure 7.2, where t_{start} and t_{end} are represented by green and red dots, respectively. The D_s and D_e represent the proportion of t_{start} and t_{end} in the total length L. The magnitude of the difference in model performance between these two points defines D_v. Furthermore, the ratio of the distance between D_s and D_e to the L of the data stream determines the value of D_t.

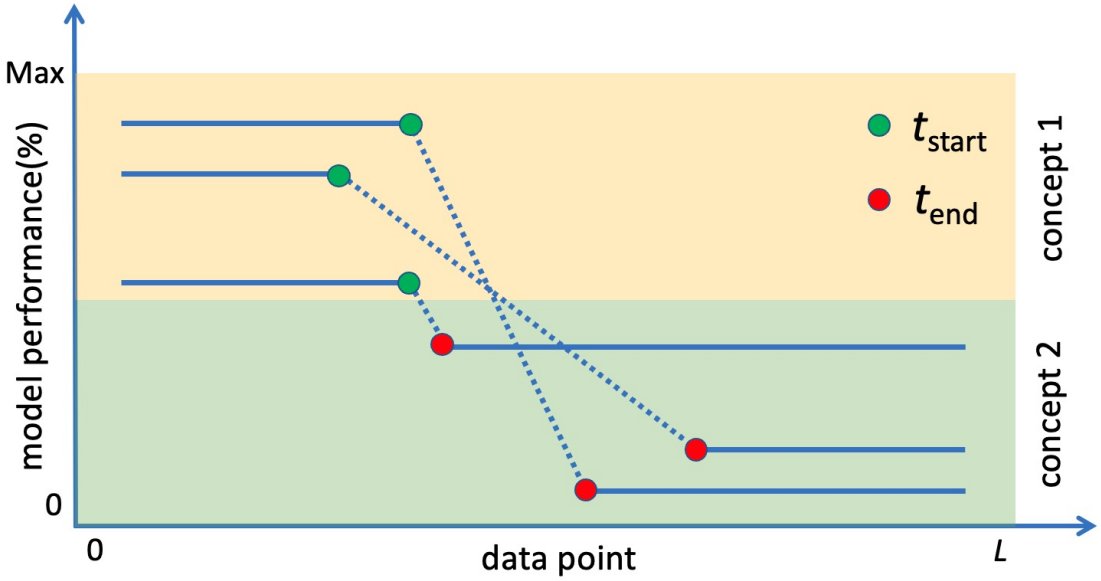


Figure 7.2: Illustration of the quadruple in various data streams with distinct types of concept drift.

Both D_s and D_e are normalized values ranging from 0 to 1, making them applicable to all situations. In contrast, D_v and D_t can be adjusted based on different contexts and requirements. This flexibility enables the Quadruple representation to adapt to a wide range of applications, providing robust monitoring and management of concept drift across various domains.

The following sections delve deeper into the Quadruple representation: Section 7.2.1 discusses D_s and D_e in detail, while Section 7.2.1 introduces D_v and Section 7.2.1 presents D_t.

Normalized Drift Start D_s and Drift End D_e

The components D_s and D_e identify the time points at which the drift begins and ends, respectively. Recognizing these time points is crucial for determining the occurrence and duration of drift. Notably, D_e signifies the point in the data stream where concept drift concludes and stabilization takes place. Despite its importance, only a limited number of studies have explored D_e in the literature.

To make the measurement of D_s and D_e applicable to various data stream situations with concept drift, we employ normalized values within the range of [0, 1] instead of absolute values. This normalization enables seamless integration of these parameters within the QuadCDD deep learning framework.

Given the definitions of D_s and D_e, we can represent them with the following equations:

$$D_s = \frac{t_{\text{start}}}{L}, \tag{7.1}$$

$$D_e = \frac{t_{\text{end}}}{L}, \tag{7.2}$$

where t_{start} and t_{end} denote the start and end position of concept drift in the data stream, respectively, and L represents the total length of the data stream. It is essential to ensure that $D_s < D_e$ for a valid time range of drift occurrence.

Drift Severity D_v

As previously mentioned, the D_v component quantifies the impact of drift on the model's performance, providing valuable insights for guiding the adaptation process. This parameter can be tailored to suit the specific needs of various applications and contexts.

The assessment of severity in concept drift scenarios holds significant value, but its implications may vary depending on the specific context. In this study, severity is defined as the absolute difference in model performance, particularly focusing on the metric of accuracy. Accuracy, being a fundamental measure of a model's predictive capability and its ability to accurately classify instances, provides a reliable basis for evaluating severity. By quantifying the disparity between the initial concept C_i and the concept that arises after the occurrence of drift C_d, we can gain valuable insights into the extent to which concept drift impacts the model's accuracy. We formally define the value of severity as follows:

$$D_v = |ACC_{C_i} - ACC_{C_d}|, \tag{7.3}$$

where $0 < D_v < 1$. It is crucial to note that C_d is distinct from D_e, as the stabilization of C_d always occurs after D_e. For instance, in a given scenario, D_s starts at a certain point, and D_e ends at another point, but the accuracy value will be significantly different after D_e when the severity varies. Therefore, in this study, we set a threshold equal to $C_d = 0.1 * L + D_e$ as the indicator for the next concept.

The value of D_v can result in different levels of accuracy decline when drift occurs. We define $threshold\_severity$ as a threshold to determine if a severity is impactful enough to affect learning in data streams with concept drift. The $threshold\_severity$ needs to be fine-tuned during the adaptation phase, as each scenario presents unique characteristics and definitions of drift. By quantifying drift severity and considering its effects on model performance, we enable more effective adaptation strategies and enhance the overall robustness of learning algorithms in the presence of concept drift.

Drift Type D_t

In this work, we analyze three primary types of concept drift identified in existing literature: abrupt, gradual, and incremental (Wang et al., 2023). We will not be considering recurrent drift in our analysis. However, it is worth noting that these three types of drift can be effectively condensed into two base drifts based on the Quadruple representation. As a result, our focus lies on abrupt and incremental drift, which are represented in the Quadruple as D_t, a binary value that is either 0 or 1. This simplification allows for a more streamlined analysis, as illustrated in Figure 7.3. The invariant Drift Type D_t remains unchanged despite the presence of diverse internal concept drift patterns. The green and red dots represent the starting and ending points of the concept drift within the data stream, while the dashed line denotes the distinct shape of the concept change. It is evident that D_t is determined by D_s and D_e, and how they change with respect to each other has no effect on D_t.

The classification of concept drift in our study is determined by the length of the drift. However, the length is not fixed across all scenarios, as data streams often exhibit varying lengths. To account for the variability in concept drift patterns, we introduce a $Type\_threshold$ parameter that helps determine the type of drift. In our approach, the $Type\_threshold$ is defined as a proportion of the periodic length of the data stream. For instance, consider a data stream with a periodic length

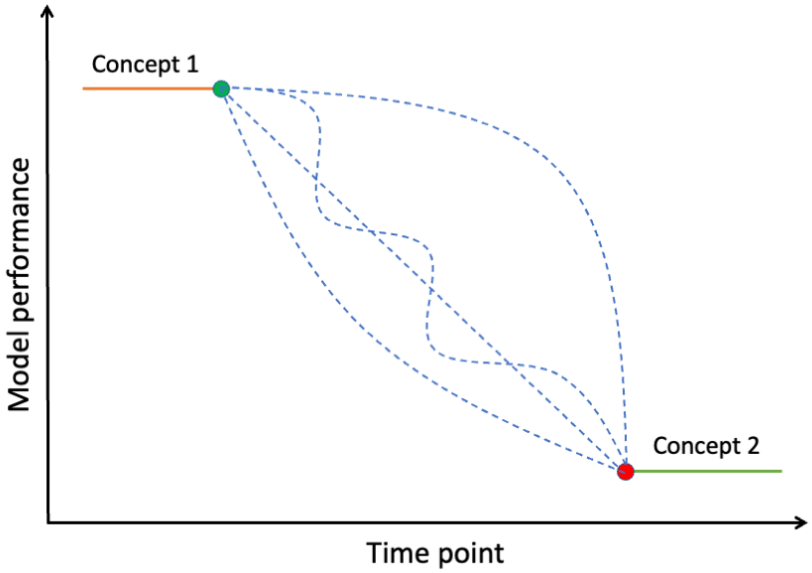


Figure 7.3: The invariant Drift Type D_t remains unchanged despite the presence of diverse internal concept drift patterns. The green and red dots represent the starting and ending points of the concept drift within the data stream, while the dashed line denotes the distinct shape of the concept change. It is important to note that the value of D_t is solely determined by the start and end points, resulting in all these shapes having the same D_t value.

of 1000 and another length of 10000. If we set the $Type\_threshold$ to 0.2, the corresponding lengths used to classify the type of concept drift would be 200 (1000 * 0.2) and 2000 (10000 * 0.2), respectively.

By introducing the $Type\_threshold$ parameter, we can adaptively adjust the length thresholds based on the characteristics of the data stream. This approach allows us to effectively classify different types of concept drift by comparing the actual lengths of the drift periods to the thresholds derived from the $Type\_threshold$ parameter. Similar to the severity of concept drift, the type is only determined by the absolute distance between D_s and D_e.

$$D_t = \begin{cases} 1, & \text{if } |D_S - D_e| > Type\_threshold \\ 0, & \text{otherwise} \end{cases} \tag{7.4}$$

Understanding the type of concept drift is critical for decision-making processes, as it informs the selection of appropriate adaptation strategies. When the drift is abrupt and the severity is high, there is insufficient data between the two concepts to be considered as training data. Conversely, the data obtained during the drift transition can be utilized as training data to refine the model's

performance following concept drift.

7.2.2 QuadCDD framework

The QuadCDD framework is specifically designed to enhance the understanding of concept drift by identifying and quantifying concept drift in data streams, and then offering a reaction strategy to address the performance degradation caused by the concept drift. By utilizing the generated Quadruple, the framework improves our understanding of concept drift and decision-making in response to concept drift.

The entire process is divided into four stages: pre-training, adaptive fine-tuning, detection, and decision-making, as depicted in Figure 7.4.

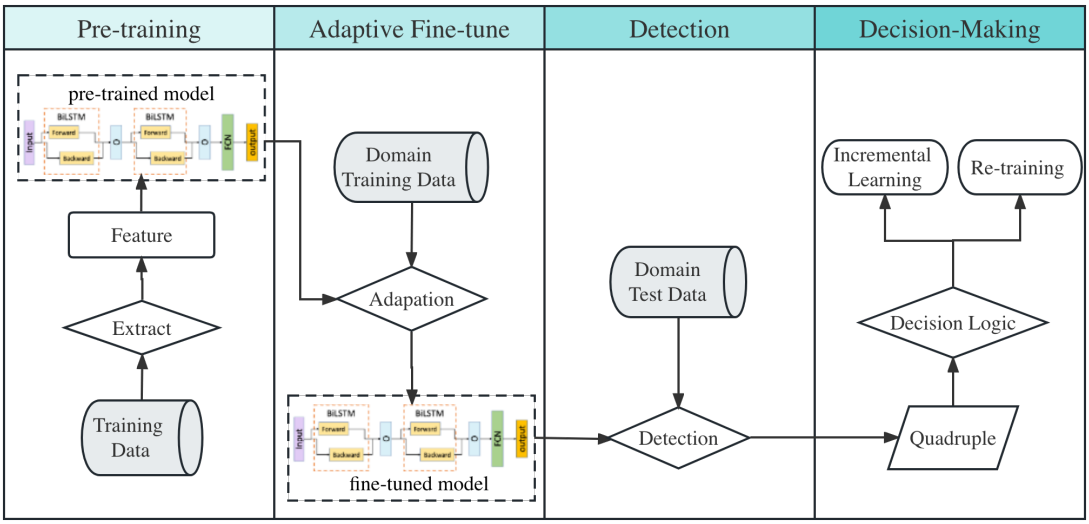


Figure 7.4: The workflow of our proposed QuadCDD framework, consist of four phase, Pre-training, adapative fine-tune, detection and Decision-making

The framework begins with the pre-training phase, where training data is first extracted and fed into the designed deep neural network to establish an initial model. Subsequently, as new data is introduced, the data stream length and severity can be calculated, allowing the trained model to fine-tune adaptively to accommodate the data stream with different concept drifts.

In the detection phase, the fine-tuned model generates a Quadruple, serving as a crucial input for the decision-making process in pinpointing and addressing concept drift. This Quadruple is represented as (D_s, D_e, D_v, D_t). Decisions are based on the Quadruple, with the goal of maintaining the highest possible overall accuracy. Consequently, incremental training on data following the

detection of drift or re-training a new model using post-drift data may be considered as potential options.

Pre-training

In the pre-training phase, there are two sub-phases: feature extraction and initial model training. Firstly, the training data is generated with a pre-defined Quadruple. The D_s and D_e are pre-defined when the data stream is constructed, while the D_t is determined by Equation 7.4. Then, the constructed training data will be processed by a machine learning model; in this work, we use the Naive Bayes as the base model. The feature we used in this work is the accuracy rate. Subsequently, the D_v is calculated by the Equation 7.3. This calculation provides insights into the severity of concept drift by considering the difference in accuracy rates at the drift start and end points. The inclusion of the 10% portion after the D_e accounts for the fact that the change in accuracy becomes significantly different only after a small portion of the data stream has elapsed past the drift end point. This ensures that the severity measured accurately reflects the impact of the concept drift on the model's performance. Finally, the training data with its label is built, which consists of an accuracy list with a length equal to the data stream and the generated Quadruple.

The next sub-phase is the initial model training. As discussed in the previous section, in the Quadruple, it is evident that D_s, D_e, and D_v are floating-point values ranging from 0.0 to 1.0, while D_t is a binary value of 0 or 1. Consequently, equal weights cannot be simply allocated to all four components.

The network structure is designed to be independent of the framework, meaning it can be replaced or modified to suit various scenarios. To maintain focus on our contribution, we employ a simple architecture consisting of two Bidirectional LSTM layers and a fully connected linear layer. While additional layers could lead to increased accuracy, time and computational costs must be considered as trade-offs.

Given that rapid and accurate drift detection is the foremost priority, we assign higher and equal weights to D_s and D_e. Conversely, D_v receives a slightly lower weight, and the smallest weight is allocated to D_t. With this in mind, we define the loss function as follows:

$$loss = \sqrt{\alpha_1(D_s - \dot{D}_s)^2 + \alpha_2(D_e - \dot{D}_e)^2 + \alpha_3(D_v - \dot{D}_v)^2 + \alpha_4(D_t - \dot{D}_t)^2} \tag{7.5}$$

where α_i, with i = [1,2,3,4], represents the weight assigned to each component in the Quadruple. The D_x and $\dot{D}_x$ denote the actual value and predicted value in the Quadruple, respectively.

In this work, we use the weight vector $\alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]$ defined as $\alpha = [0.3, 0.3, 0.25, 0.15]$ to represent the importance of each item in the Quadruple: D_s, D_e, D_v, and D_t, respectively. This allocation of weights acknowledges the significance of different aspects of concept drift while maintaining a balanced approach.

The higher weights assigned to D_s and D_e ensure that the system pays more attention to the starting and ending points of the drift. The start point (D_s) is crucial because it provides an indication of when the drift begins, which directly affects the performance of the predictive model. Likewise, the end point (D_e) is important because it reveals when the drift concludes and stabilization occurs. By assigning different weights to these components, the framework effectively prioritizes their joint contribution to the overall understanding of concept drift.

Adaptive Fine-tuning for Concept Drift Detection

In data streams generated by various sources, accurate concept drift detection necessitates adapting the pre-trained model to new situations. Although data streams may share similarities, their underlying characteristics can differ. To ensure the robustness of the proposed QuadCDD framework, we implement an adaptive fine-tuning strategy for the pre-trained model from Section 7.2.2. This strategy effectively detects concept drifts in diverse contexts.

The adaptation process is especially critical for components D_v and D_t. Severity (D_v) represents the impact level caused by the concept drift, and the maximum severity level may vary depending on the specific scenario. For example, in a binary classification problem, the most severe concept drift could involve a complete label flip where data points initially labeled as 0 become 1 after the concept drift occurs. However, the D_v generated in this case may not apply directly to other situations. Consequently, the framework must be fine-tuned on new data to ensure proper adaptation.

To achieve this, new domain data will be collected and used for adaptive fine-tuning. During the

fine-tuning process, the learning rate is set higher than in the pre-training phase, and the number of epochs is reduced, enabling rapid adaptation. The fine-tuned model can then effectively perform concept drift detection in new data streams, ensuring the detection process remains accurate and robust across various scenarios.

Detection

Traditional concept drift detection methods have predominantly focused on identifying the drift's starting point. This approach, however, may not offer a comprehensive understanding of the concept drift. Quadruple D_s, D_e, D_v, D_t generated encompassing the drift's start point, end point, severity and type in more detailed. This output presents a more holistic perspective on the concept drift, facilitating a deeper comprehension of the underlying causes and empowering the development of more effective strategies to react to concept drift.

Employing a quadruple output in the detection phase of concept drift signifies a crucial advancement in the field of data analysis. This method offers a more intricate and sophisticated insight into the dynamic data environment, paving the way for more effective strategies to adapt to the evolving data landscape. The quadruple output serves as a valuable tool for researchers and practitioners, enabling them to make well-informed decisions about detecting and responding to concept drift in their datasets.

Decision-Making and deployment for Concept Drift handling

The objective of this phase is to ensure the model's performance remains consistent in the presence of concept drift. Decisions are made based on the Quadruple output of the Detection phase.

Two prevalent strategies exist for handling concept drift. The first, incremental learning, is applied when the concept drift is not severe and only marginally impacts the trained model. In this case, incremental learning suffices to maintain the model's performance. The second strategy, retraining, is employed when the drift severity is high, necessitating the discard of the old trained model and training a new model on the data after the drift is detected. Incremental learning on data affected by severe drift can degrade the model's performance.

Therefore, making appropriate decisions to manage concept drift is vital for maintaining the consistent performance of a model operating in a data stream.

To achieve this, we utilize the quadruple generated by the Detection phase. The details shown in algorithm 8. The parameter D_v is crucial in determining whether to retrain the model or use incremental learning. Based on our experiments, $Threshold\_detection$ has been determined as 0.3 ± 0.1, which effectively identifies when the severity of the concept drift is low enough to warrant incremental learning. Nevertheless, this threshold can be adapted to various scenarios based on the specific requirements of the application.

Another essential parameter, D_t, plays a significant role in decision-making as the type of concept drift can offer insights into the most suitable response. Furthermore, the parameters D_s and D_e furnish vital information about when and where to deploy the decision in the data stream. Specifically, when a concept drift is detected at D_s, data from another concept should be collected, and the comprehension of the concept drift should be accomplished by D_e.

Algorithm 8 Understanding Concept Drift in Data Stream

Input: Data stream S with continuous data points

Output: Quadruple and Decision

Detection

for each piece of data chunk in Whole_stream **do**
    Understand the data stream with concept drift using a fine-tuned model
    Record the quadruple (D_s, D_e, D_v, D_t)
end for

Decision

Collect $data\_after\_D_e$ after D_e to deploy a decision to react to concept drift
if D_t **then**
    Type of Concept drift is abrupt
else
    Type of Concept drift is Incremental
end if
Determine the decision to react to the concept drift based on D_v
if $D_v \leq$ threshold **then**
    Decision making on incremental learning
else
    Decision making on re-training model
end if

Deployment

if Decision is incremental learning **then**
    Perform incremental learning on $data\_after\_D_e$ using the original model
else
    Drop the original model
    Initialize a new model and train it on $data\_after\_D_e$
end if

Incremental learning and model re-training are vital techniques for effectively addressing concept drift in data streams, with the choice between them being determined by the severity of the drift.

Incremental learning involves updating the existing model iteratively, allowing it to adapt to evolving concepts while preserving past knowledge. In the presented algorithm, the decision to employ incremental learning hinges on the severity of concept drift, represented by the D_v value, falling below a predefined threshold. When the severity is considered low, the algorithm selects incremental learning as the appropriate strategy. The original model is then leveraged to adapt to the data points after the drift's end point, D_e. This incremental learning process facilitates the model's efficient adjustment to evolving patterns and concepts within the data stream. Conversely, when the severity surpasses the threshold, re-training the model becomes necessary. This entails replacing the original model with a newly initialized one, which is then trained using the data instances after D_e. The re-training process ensures that the updated model captures the latest patterns and concepts, enhancing its performance in the face of significant concept drift. By incorporating severity as the determining factor, the algorithm intelligently selects between incremental learning and model re-training, enabling the model to maintain accuracy and effectiveness while effectively adapting to evolving dynamics in the data stream.

7.3 Experiment and Evaluation

7.3.1 Dataset and Experiment Settings

We implemented our method using Python 3.7.10. The neural network structure was established using the PyTorch 1.10.0 package. The functions employed to construct the networks are depicted in Table 7.1:

Table 7.1: Function used to construction of the neural network

Layer	**PyTorch layer**
Linear layers	torch.nn.Linear
Bidirectional LSTM layer	torch.nn.LSTM(bidirectional=True)
optimizer	torch.optim.SGD

In this study, our main objective is to validate the efficacy of our proposed framework, with less emphasis on exploring diverse neural network architectures. Our framework is designed to be adaptable to various neural network structures. For demonstration purposes, we have chosen Bidirectional Long Short-Term Memory (BLSTM) to construct the LSTM layers, accompanied by a fully connected layer as the output layer. The network employs two layers of BLSTM to ef-

fectively capture the time-series relationships in the data stream, which are influenced by concept drift. The features extracted from this process are then inputted into a fully connected layer, resulting in the generation of the quadruple. It's important to highlight that the network's architecture can be customized to suit different types of data streams.

The experiment hyperparameters consist of the first two stages of the proposed framework, Pre-training and Adaptive Fine-tuning. The details of these hyperparameters are listed in Table 7.2.

Table 7.2: Hyper-parameters used for proposed framework

	Pre-training	**Adaptive Fine-tune**
Learning Rate	1e-3	1e-2
Step size	100	100
Epoch	40	20
Decay rate	0.9	0.9

In the pre-training phase, we aim to optimize the performance of the trained network by utilizing a smaller learning rate and a larger number of epochs, resulting in higher accuracy. Consequently, the adaptive fine-tuning phase can rapidly adapt based on the pre-trained network. In this phase, we employ a learning rate that is ten times larger than in the pre-training phase and half the number of epochs.

Artificial dataset

We generate four artificial datasets using the Python library, *skmultiflow*. This library allows us to pre-define the D_s and D_e points of concept drift. Furthermore, the D_v and D_t of the concept drift can be adjusted by altering the parameters in the data stream settings.

Circle: The circle dataset contains of 1000 data samples, the binary classes are determined by the straight line cross the center of circle. The drift is introduced by changing the slope the straight line.

Sine: The sine data contains 1000 data samples, within x-axis $[0, \pi]$ and y-axis $[-1, 1]$, the decision boundary is based on the line of sine. And the drift is introduced by shifting the line of sine between 0 and π.

Hyperplane: A hyperplane in d-dimensional space is the set of points $\boldsymbol{x}$ that satisfy $\sum_{i=1}^{d} w_i x_i = w_0 = \sum_{i=1}^{d} w_i$, where x_i is the ith coordinate of $\boldsymbol{x}$. Examples for which $\sum_{i=1}^{d} w_i x_i > w_0$, are

labeled positive, and examples for which $\sum_{i=1}^{d} w_i x_i \leq w_0$, are labeled negative. Hyperplanes are useful for simulating time-changing concepts, because we can change the orientation and position of the hyperplane in a smooth manner by changing the relative size of the weights. We introduce change to this dataset by adding drift to each weight feature $w_i = w_i + d\sigma$, where σ is the probability that the direction of change is reversed and d is the change applied to every example. In this work, to generate concept drift, we choose to fix the σ and change the d to have different level of concept drift.

RandomRBF:Random Radial Basis Function stream generator. Produces a radial basis function stream. A number of centroids, having a random central position, a standard deviation, a class label and weight, are generated. A new sample is created by choosing one of the centroids at random, taking into account their weights, and offsetting the attributes at a random direction from the centroid's center. The offset length is drawn from a Gaussian distribution. The drift is created by adding a speed to certain centroids. As the samples are generated each of the moving centroids' centers is changed by an amount determined by its speed.

Real-world dataset

Two real-world datasets are used for drift detection evaluation. *Powersupply* (Wu et al., 2014) and *Posture* **posture**, are two data streams collected in the real world with multiple sensors. The details shown below:

Powersuppply: It is the hourly powersupply recording of an Italian electrical provider that records power from two sources: the main grid and other networks that have been transformed. This stream contains records for the years 1995 to 1998. Each data point contains two attributes and one label, where each attribute corresponding to a source of powersupply and the label indicating the moment of collection as a 0 to 23 integer.

Posture: The posture dataset collects data from sensors on the left/right ankle, belt, and chest at 0.1s intervals. Each data instance is made up of three features and one class label, with the features indicating the spatial position and the class label denoting the kind of posture.

Concept drift can be generated with a specified start point and duration in artificial datasets. To do the same thing, we construct the concept drift in real-world dataset by selecting different concept

and combine them into data stream.

Benchmark Method

In this work, we employ nine methods, of which six drift detection methods are sourced from the *skmultiflow* library (Montiel et al., 2018). These methods include DDM (Gama et al., 2004), ADWIN (Bifet and Gavalda, 2007), EDDM (Baena-Garcıa et al., 2006), PH (Page, 1954), HDDM_A (Frías-Blanco et al., 2015), and HDDM_W (Frías-Blanco et al., 2015). Additionally, we utilize two methods from the *river* library (Montiel et al., 2021), namely FHDDM (Pesaranghader and Viktor, 2016) and MDDM (Pesaranghader et al., 2017). Lastly, OASW (Yang and Shami, 2021), a method similar to DDM, is publicly available.

For majority of the methods, we employ the default parameters, with the exception of ADWIN, for which we modify the settings to enhance performance. Table 7.3 presents the specific values or thresholds. The warning and drift thresholds for DDM, HDDM, HDDM_A, and HDDM_W are denoted by α and β, respectively. In the case of ADWIN and PH, there is only a drift threshold, represented by δ.

In HDDM W, λ signifies the weight assigned to the data, with more recent data receiving less weight. For FHDDM, MDDM, and OASW, we employ a grid search strategy to determine the most suitable parameters, thereby achieving optimal performance in this work.

Table 7.3: Algorithm parameters or threshold description

Algorithms	**Parameters(Thresholds)**
DDM (Gama et al., 2004)	$\alpha = 0.05, \beta = 0.01$
ADWIN (Bifet and Gavalda, 2007)	$\delta = 0.05$
EDDM (Baena-Garcıa et al., 2006)	$\alpha = 0.9, \beta = 0.95$
PH (Page, 1954)	$\delta = 0.005$
HDDM_A (Frías-Blanco et al., 2015)	$\alpha = 0.005, \beta = 0.001$
HDDM_W (Frías-Blanco et al., 2015)	$\alpha = 0.005, \beta = 0.001, \lambda = 0.05$
FHDDM (Pesaranghader and Viktor, 2016)	$\delta = 0.01$
MDDM (Pesaranghader et al., 2017)	$difference = 0.1, \delta = 0.01, \lambda = 0.05$
OASW (Yang and Shami, 2021)	$\alpha = 0.85, \beta = 0.95$

7.3.2 Concept Drift Detection Experiment and Evaluation

In this section, we use the pre-trained and fine-tuned network performing the concept drift detection on dataset with variant concept drift.

The detection comparison runs 2000 trails for each dataset. Since the benchmark methods in Section 7.3.1 only perform the detection by reporting the drift start point D_t, therefore, we conduct the comparison of the D_s for all the benchmark methods and QuadCDD. The result is shown in Table 7.4.

Table 7.4: Concept drift detection delay, denoted by $\left|D_s - \hat{D_s}\right|$, where D_s represents the ground truth of the concept drift start point, and $\hat{D_s}$ represents the detected drift start point. A smaller difference between these two variables indicates a stronger detection ability of the concept drift detection method.

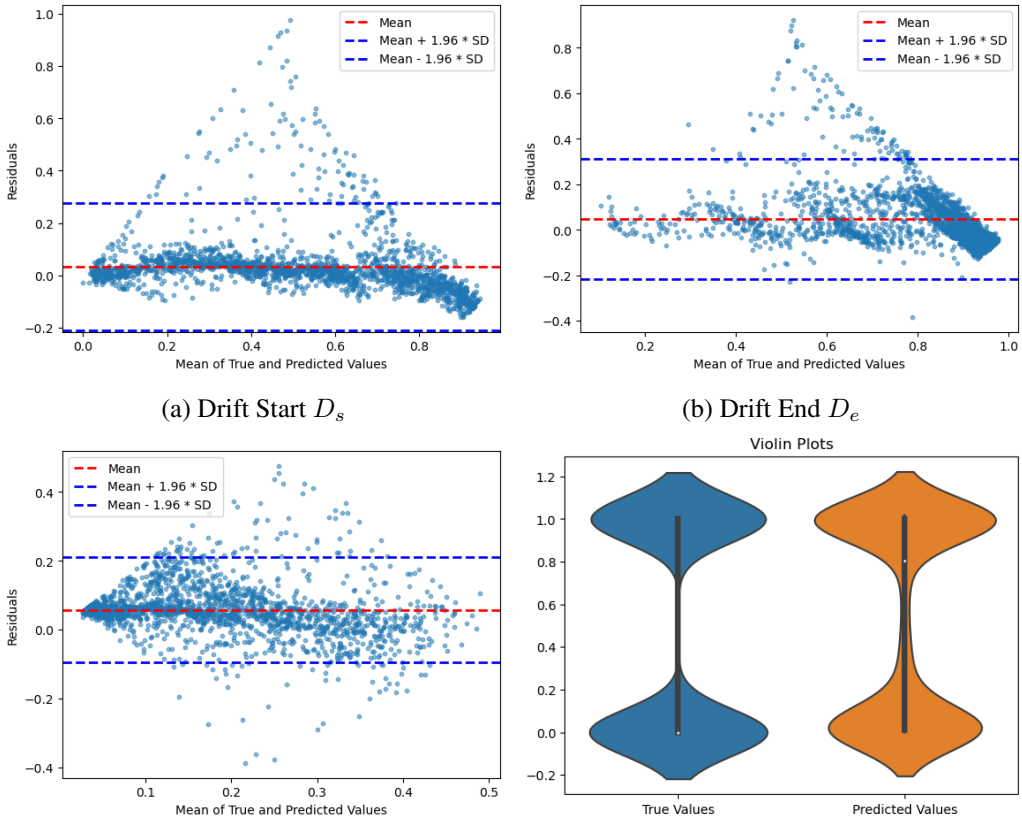
	DDM	**ADWIN**	**EDDM**	**PH**	**HDDM_A**	**HDDM_W**	**FDDM**	**MDDM**	**OASW**	**QuadCDD**
Circle	6.08	43.18	35.33	17.15	34.86	28.56	29.61	15.32	9.14	8.20
Sine	54.27	202.20	62.77	112.10	103.64	94.98	135.73	144.02	60.48	67.88
Hyperplane	43.85	179.17	53.85	100.87	93.68	106.82	142.72	128.28	77.51	80.25
RandomRBF	291.16	316.91	331.77	261.43	285.00	335.33	360.55	360.36	251.82	**176.26**
Posture	39.57	102.20	353.73	97.16	25.40	301.26	404.96	398.95	73.16	**12.38**
Powersupply	18.69	47.58	16.15	53.76	13.62	3.78	4.78	7.62	18.68	23.54
Average	75.60	148.54	142.27	107.08	92.70	145.12	179.73	175.76	81.80	**61.41**

Table 7.4 presents the performance of various drift detection methods, including our proposed method, QuadCDD, across six different datasets: Circle, Sine, Hyperplane, RandomRBF, Posture, and Powersupply. The performance metric used is the distance from the drift start point to the predicted drift start point, with lower values indicating better accuracy in drift detection. On average, QuadCDD outperforms all other methods, achieving the lowest average distance of 61.41, indicating its effectiveness in accurately predicting drift start points. Notably, QuadCDD demonstrates exceptional performance on the RandomRBF and Posture datasets, with distances of 176.26 and 12.38 respectively.

7.3.3 Understanding Concept Drift

In this section, we employ our pre-trained and adaptive fine-tuned model to generate quadruple parameters as said in Section 7.2.2, which are subsequently utilized to examine the concept drift in greater detail. To ensure the robustness of our findings, we conduct 2000 experimental trials on each dataset. Data for these experiments are generated using various quadruple parameter combinations, specifically has 1000 data points in each trail of data.

For illustrative purposes, we present the results derived from the Circle dataset as Figure 7.5.



(a) Drift Start D_s

(b) Drift End D_e

(c) Drift Severity D_v

(d) Drift Type D_t

Figure 7.5: Quadruple Understanding on Cricle Datasets, the first three plots shows the Bland-Altman plot of the parameters, the red dash line indicates the mean of the residuals, and the two blue dash line indicates the mean ± 1.96 standard deviations. The last plot is the violin plot visualized the binary classification.

D_s, D_e, and D_v The first three figures, Figure 7.5a, 7.5b, and 7.5c, display the residual values between the true and predicted outcomes for D_s, D_e, and D_v. The x-axis represents the mean of the true and predicted values, while the y-axis shows the absolute value of the residuals. These figures are constructed using the Bland-Altman plot methodology (Bland and Altman, 1986). Given that D_s, D_e, and D_v are randomly generated within a limited range, we treat these three parameters as continuous variables. The Bland-Altman plot is derived from the ground truth values and their corresponding predicted values obtained from the QuadCDD framework. Smaller residuals indicate better performance.

As observed, most residuals cluster around the line where the residual equals 0, which signifies a close alignment between the true and predicted values. To further investigate the agreement between these values, Bland-Altman lines are drawn in each subfigure. These lines represent the mean difference and the limits of agreement, defined as the mean difference ± 1.96 standard

deviations.

By applying the Bland-Altman plot principles, we can evaluate the model's performance through the calculation of the percentage of data points falling within the limits of agreement. These findings are presented in Table 7.5. The table reveals that the QuadCDD framework provides stable predictions for these three parameters, with nearly 95% of values falling within ± 1.96 standard deviations. This indicates that the difference between the real values and the QuadCDD processed results is statistically negligible.

Table 7.5: The table presents the percentage of data points falling within the Bland-Altman lines for various categories of concept drift, including Circle, Sine, Hyperplane, RandomRBF, Posture, and Powersupply. The displayed values correspond to the measurements D_s, D_e, and D_v associated with each category, along with the average values across all categories. Notably, a significant majority of the data points (within the range of 95% $\pm$ 1.96 standard deviation) demonstrate a high level of agreement with the Bland-Altman analysis.

	D_s	D_e	D_v
Circle	95.19	94.95	94.25
Sine	96.27	95.15	96.43
Hyperplane	97.13	91.74	95.09
RandomRBF	93.12	95.63	97.10
Posture	95.81	97.15	94.81
Powersupply	95.39	92.18	95.72
Average	95.48	94.46	95.56

Table 7.6 shows the absolute residual values for the three parameters across different datasets. The table demonstrates a high concentration of QuadCDD-generated parameters. When combined with the results from Table 7.5, it is evident that the continuous parameters D_s, D_e, and D_v generated by QuadCDD exhibit high accuracy.

Table 7.6: Residual Analysis of Concept Drift. The table presents the absolute residual values denoted by D_s, D_e, and D_v for different categories of concept drift. D_s and D_e are measured in units of 1000 data points, while D_v is measured in percentages.

	D_s	D_e	D_v
Circle	0.017	0.010	0.046
Sine	0.17	0.059	0.021
Hyperplane	0.037	0.014	0.005
RandomRBF	0.088	0.018	0.005
Posture	0.013	0.021	0.01
Powersupply	0.038	0.027	0.009
Average	0.0605	0.024	0.016

D_t For the final parameter, D_t, which takes on binary values of 0 or 1, we employ a violin plot for visualization, as depicted in Figure 7.5d. The violin plot indicates that there is no substantial distinction between the two groups represented by the values of 0 and 1.

To further assess QuadCDD's capability in determining D_t, we present the accuracy and F1 score in Table 7.7. These findings demonstrate that the ability to distinguish D_t is satisfactory, with an average accuracy of 93.9% and an F1 score of 83.3%. Moreover, this supports the notion that assigning a smaller weight to D_t is adequate for generating accurate results.

Table 7.7: D_t results, The D_t is either 0 or 1 based on Equation. 7.4.

	Acc (%)	F1 (%)
Circle	93.2	82.8
Sine	94.2	83.2
Hyperplane	92.2	80.5
RandomRBF	94.2	83.4
Posture	94.1	85.3
Powersupply	94.1	84.3
Average	93.9	83.3

7.3.4 Decision-Making in Response to Concept Drift

In this section, we analyze the quadruple generated from Section 7.3.3 to make informed decisions in response to data streams with concept drift. Concept drift impacts the performance of trained models, necessitating adaptive strategies to maintain their performance. The degree of performance change largely depends on the severity D_v of the concept drift.

As an illustration, we employ the Circle dataset with a decision boundary defined as a straight line intersecting the circle. The initial slope of $-\infty$ is chosen for the training dataset. To simulate concept drift, we introduce varying slopes of $-1000, -100, -10, -5, -1, 0, 1, 5, 10, 100, 1000$. The drift starting point (D_s) and ending point (D_e) are pre-defined at data points 500 and 1000, respectively, within a total generated data length of 5000. Subsequently, we plot the accuracy of the model when no action is taken in response to the drift, as depicted in Figure 7.6. This analysis aims to highlight that larger differences between the original data and the drift data correspond to more significant changes in the model's performance.

It is evident that the model accuracy starts to decline from point 500, and the magnitude of the decline becomes significantly different from point 1000 onward. Upon overall observation, the

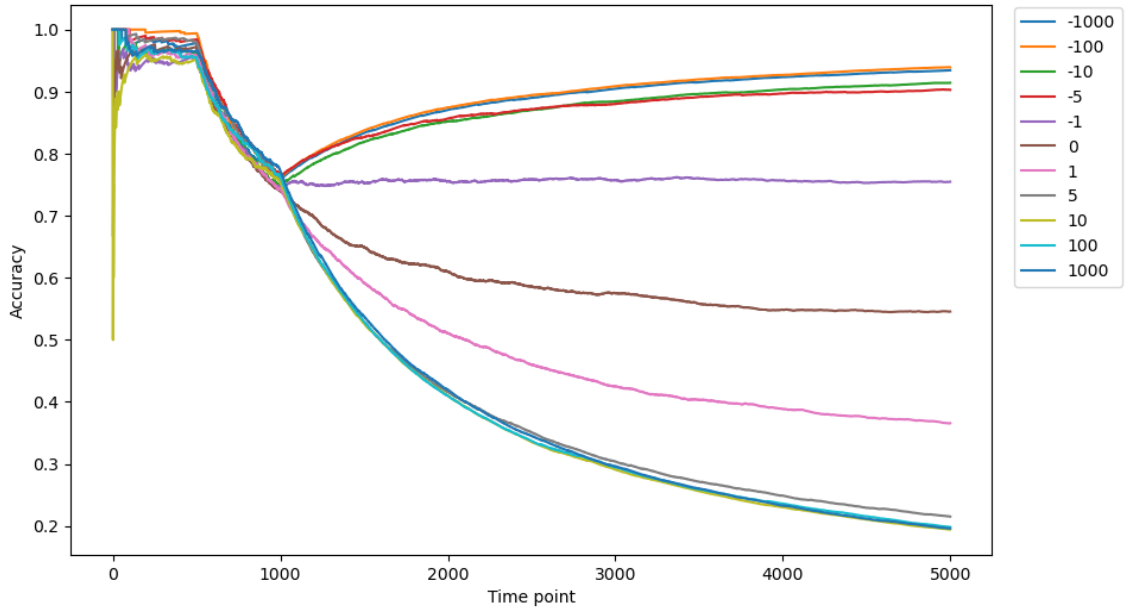


Figure 7.6: Effect of Concept Drift on Model Accuracy Across 11 Data Streams. Each data stream consists of two concepts: the first concept remains constant from 0 to point 500, which is consistent across all data streams. The second concept exhibits varying levels of drift severity, indicated by the slope of the decision boundary (a straight line for the Circle dataset). The initial slope for the first concept is set to $-\infty$, while the slopes for the subsequent concepts range from -1000 to 1000 (indicated in the legend at the right side). This figure illustrates that larger differences in slope lead to more significant declines in model accuracy. The trend becomes particularly evident after point 1000, emphasizing the impact of concept drift on model performance.

differentiation of the model accuracy is highly correlated to the D_v, which is induced by the magnitude of the slope change in this case. In other words, the higher the magnitude, the larger the decline. For instance, the accuracy with a slope of -1000 exhibits the best performance, whereas the worst accuracy occurs at a slope of 1000.

Informed decisions can be made using the quadruple generated by the QuadCDD framework. In this work, we employ two adaptive strategies based on the quadruple and compare the average accuracy before and after their application. Consequently, we obtain three accuracy metrics: $Acc_{original}$, $Acc_{incremental}$, and $Acc_{re-train}$. As illustrated in Figure 7.7, $Acc_{original}$ declines sharply when no action is taken, while $Acc_{incremental}$ follows a similar trend, with an even steeper decrease after a slope of 0. It is evident that higher severity levels correspond to lower accuracy for $Acc_{original}$ and $Acc_{incremental}$.

In contrast, the re-training strategy, $Acc_{re-train}$, consistently maintains a high and stable average accuracy. The significant difference between $Acc_{incremental}$ and $Acc_{re-train}$ begins around index

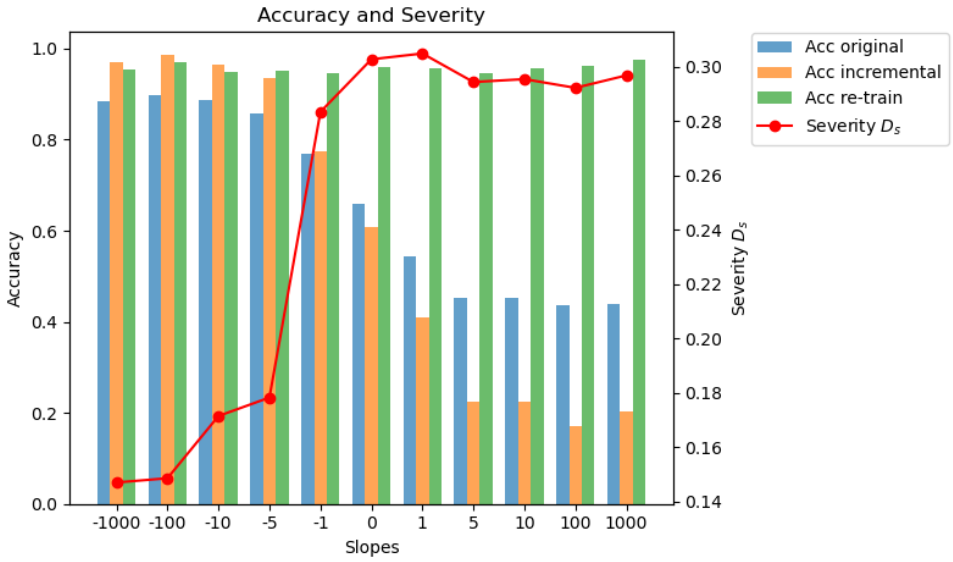
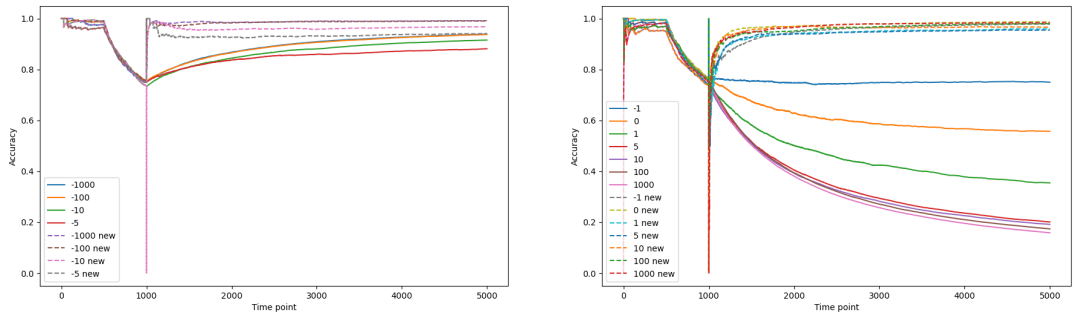


Figure 7.7: Comparison of accuracy for the three strategies: Acc_{original} represents the accuracy rate of the trained model without any action taken in response to concept drift, $Acc_{\text{incremental}}$ illustrates the performance when incremental learning is applied based on quadruple-informed decisions, and $Acc_{\text{re-train}}$ indicates the decision to re-train the model. The figure reveals that Acc_{original} experiences a sharp decline in accuracy, while $Acc_{\text{incremental}}$ maintains relatively high and stable performance until index -5. In most situations, $Acc_{\text{re-train}}$ demonstrates consistently high accuracy.

-1, which corresponds to a severity of 0.28. Beyond this point, $Acc_{incremental}$ is considerably lower than $Acc_{re-train}$. However, a trade-off between the strategies should be considered when selecting the appropriate decision.

By applying the decision-making algorithm, we adopt different strategies for data streams with concept drift and varying D_v, as depicted in Figure 7.8. The figure clearly shows two phases. In the first phase, before the drift ends, all accuracy lines share a similar pattern, with stable accuracy ranging from 0 to 500, where D_s starts. This is followed by a dramatic drop from 500 to 1000, which corresponds to the defined D_e. In the second phase, the accuracy rates vary for cases with and without reactions to drift. It is evident that the accuracy after taking adaptive actions (dash line) is consistently higher than the accuracy without any adaptation (solid line), indicating the effectiveness of our decision-making approach for handling concept drift.

As illustrated by the left sub-figure, the results from continuous learning show a rapid recovery of accuracy in response to concept drift and consistently stable performance thereafter. When compared to the $Acc_{original}$ variable, the recovery speed following the decision-making process is faster, and the accuracy at the end of the data stream is higher. These findings substantiate that



(a) Accuracy before and after decision-making with incremental learning based on QuadCDD

(b) Accuracy before and after decision-making with re-training based on QuadCDD

Figure 7.8: Accuracy before and after decision-making based on QuadCDD

quadruple-based decision-making is effective in enhancing the model's performance when faced with low severity levels.

In contrast, the right sub-figure reveals a dramatic decrease in accuracy when no action is taken, which corresponds to larger D_v values causing a greater degree of accuracy drop. This observation underscores the substantial performance improvement achieved through model retraining. As corroborated by Figure 7.7, continuous learning is not suitable in this particular scenario. Instead, retraining the model upon drift cessation presents a more viable option.

Table 7.8: Accuracy rate (%) comparison on six datasets, the first column is the accuracy rate without reaction to the concept, the second column is the accuracy rate after decision-making with our proposed QuadCDD framework

	No action	QuadCDD
Circle	61.20	93.51
Sine	62.35	94.83
Hyperplane	64.59	93.62
RandomRBF	75.42	95.72
Posture	67.19	95.35
Powersupply	76.13	97.72
Average	67.81	95.12

Table 7.8 displays the average accuracy obtained without taking action in response to data streams with concept drift (shown in the No action column) and the accuracy rate after informed decisions are made (shown in QuadCDD column). The decision could be either incremental learning or re-training, depending on the D_v value. It is clear that the model performance improves significantly by 27.31%, demonstrating that the decisions made effectively maintain high and stable model performance.

In summary, the impact of concept drift on the performance of data streams primarily depends on D_s. We can observe that the effect of D_s on performance is directly proportional to its magnitude. Our QuadCDD method enables a detailed understanding of concept drift through the detection of quadruples and facilitates tailored responses to different concept drifts. We adopt varying decisions based on different D_s values to ensure the model's stable performance. Our experiments demonstrate that our decision-making approach results in an average performance improvement of 27.31% compared to taking no countermeasures, achieving an average accuracy of 95.12%.

7.4 Summary

In conclusion, our novel Quadruple-based Approach for Understanding Concept Drift framework in Data Streams addresses the limitations of existing concept drift detection methods that primarily rely on hypothesis testing frameworks. By providing a more detailed understanding of concept drift through the use of quadruples, our framework allows for a comprehensive analysis of drift start, drift end, drift severity, and drift type. This enables informed decision-making and the adoption of appropriate actions to handle various concept drifts effectively, resulting in high and stable performance in data streams with concept drift. Our experimental results demonstrate the effectiveness of the QuadCDD framework in accurately detecting and understanding concept drifts, as well as preserving the stability and performance of models when faced with such drifts. This research contributes to the field of data stream analysis by offering a more holistic approach to concept drift detection and understanding, ultimately benefiting a wide range of applications that depend on accurate and stable predictions in the presence of evolving data streams.

Chapter 8

Conclusion and Future Research

8.1 Conclusions

The rapid expansion of online applications such as sensor networks, telephony, mobile applications, and cloud computing has coincided with the rise in popularity of data stream mining, which has been one of the most notable implementations of this technology. Other applications that have benefited from this technology include cloud computing. These are only a few examples of the many different web apps that have been successful as a result of this trend. There are still a great many more. The data that is produced by these systems, which includes network logs and the flows of electronic transactions, is not only extensive but also always changing, which makes it prone to both quick development and large changes in the distributions that it is based on. This is because these systems are constantly generating new information. The flows of electronic transactions are included in this data. The problem that occurs as a consequence of such a dynamic environment is referred to as concept drift, and it presents major hurdles to the systems that are employed for online data mining and machine learning.

There are still major restrictions that need to be dealt with, despite the fact that a variety of methods for the identification and adaption of concept drift have been devised. These limits include the occurrence of complex data stream circumstances, the identification of erroneous drifts, and the infrequent discovery of these problems throughout the process of deploying a neural network.

In light of this, the purpose of this research was to develop innovative drift detection algorithms

with the ability to interpret and recognise various forms of concept drift in a more granular manner. This was done in an effort to improve the quality of drift identification already available. Within the scope of this investigation, it was planned that these algorithms would be put into action.

The findings of this study are summarized as follows:

- *The design and implementation of an innovative concept drift detection system for multi-label data stream classification was realized, allowing for concept drift detection in more intricate scenarios. This involved enhancing the capabilities of the Drift Detection Method (DDM) through the integration of a false positive rate, which resulted in an improved detection performance. (For details, refer to Objective 1, Chapter 3)*

  The innovation in this detection method lies in broadening the application of DDM to more complex scenarios, specifically those involving multi-labeled data streams. The incorporation of a false positive rate leads to a more sensitive reaction to concept drift, resulting in prompt and more accurate concept drift detection outcomes. This improvement also underscores the effect of indicator diversity on the performance of concept drift detection.

- *A novel concept drift detection method, referred to as Noisy Data Drift Detection Method (NTDDM), was developed for noisy data streams. The system showed exceptional performance in scenarios where the data stream had both concept drift and noise. (For details, refer to Objective 2, Chapter 4)*

  The initial step involved quantifying and defining two different types of noise in the data stream. Three metrics for a comprehensive measure of concept drift were then developed, facilitating a more thorough comparison of the concept drift detection performance. The proposed NTDDM detects concept drift by identifying potential drift using a detection function, then filters out the drift caused by noise with a validation function. Overall, the NTDDM showed strong performance in detecting concept drift in data streams with noise.

- *An innovative concept drift detection method was developed that leverages the prediction performance and stability of machine learning models, with the aim of improving detection performance by focusing more on the change in the model itself rather than its output. (For details, refer to Objective 3, Chapter 5)*

Changes in model performances, which are statistically significant, are usually used for drift detection. This method leverages changes in model stability to detect concept drift, where a vote-based incremental ensemble learning model could capture concept drift by monitoring a weighted indicator with prediction performance and stability. The experimental results demonstrated improved performance in drift detection.

- *A novel model-centric concept drift detection method based on transfer learning with deep neural networks was developed. This aims to accelerate the efficiency of concept drift detection by harnessing the power of deep neural networks. (For details, refer to Objective 4, Chapter 6)*

  This method explores the potential of deep neural networks to replace conventional machine learning methods for concept drift detection. While deep neural networks can handle more complex machine learning tasks, the considerable computational resources and time required are at odds with the immediate response requirement for concept drift. Thus, the use of transfer learning to freeze parts of the neural network helps reveal the capability of deep neural networks for prompt and accurate drift detection.

- *A novel concept drift detection and understanding method was developed to provide a comprehensive understanding of the concept. The result, represented by a quadruple comprising the drift start, drift end, drift severity, and drift type, provides a holistic perspective on concept drift. (For details, refer to Objective 5, Chapter 7)*

  The premise of this method is rooted in the observation that most concept drift detection methods focus heavily on drift start detection. However, drift start is just one aspect of the phenomenon. Leveraging the power of deep neural networks, a quadruple containing four pieces of information can be generated simultaneously. Experimentation shows that taking informed and instructive actions based on quadruple results leads to high and stable performance in data stream learning with concept drift.

8.2 Limitations

Despite the advancements made in this work on concept drift detection, several limitations have been identified, paving the way for future research directions. These are outlined as follows:

- **Lack of Complexity Analysis:** While our methods demonstrate effectiveness in detecting concept drift, an in-depth analysis of their computational complexity is lacking. This is crucial, especially for algorithms deployed in data stream mining, where efficiency and speed are paramount.
- **Noise Impact Study:** The influence of noise on concept drift detection has not been explored extensively. Our discussion primarily revolves around the impact of a single type of noise, despite the common occurrence of multiple noise types in real-world scenarios. Future work should delve into the combined effects of various noise types on concept drift detection.
- **Theoretical Basis for Weight Change Utilization:** Our approach uses weight changes in deep neural networks as an indicator for concept drift detection. However, this application lacks a solid theoretical foundation. Future research could benefit from an explanatory analysis from the perspective of explainable AI to deepen the understanding of this approach.
- **Consideration of Recurrent Concept Drift:** Among the four types of concept drift, recurrent concept drift was not addressed in our study. Given its prevalence in real-world applications, it is imperative to consider this type of drift in future research to enhance the robustness of detection methodologies.
- **Scalability and Real-World Applicability:** There is a need to test the scalability of our proposed methods in large-scale, real-world scenarios. This includes evaluating the methods' performance in rapidly changing and diverse data environments to ensure their practical utility.
- **Integration of Advanced Neural Architectures:** Future research should also explore the integration of more sophisticated neural network architectures and techniques, such as transformer, to improve the adaptability and efficiency of the concept drift detection methods in varying data stream scenarios.

8.3 Future Research

This research has successfully made significant advancements in the field of concept drift detection, yielding a collection of novel methodologies with demonstrated proficiency in complex

data stream classification scenarios. Notwithstanding these developments, there exist numerous research directions that could augment and expand the practicality of these methodologies. We intend to address these areas in our future work:

- **Incorporating Varied Data Modalities:** While our research has proficiently addressed the challenges associated with multi-label data streams and noisy data scenarios, it is crucial to expand our scope to encompass more complex and diverse data modalities. These would include high-dimensional data, imbalanced data streams, as well as multimodal data stemming from sources such as video and audio. Multimodal data offers a rich and complex representation of information, presenting novel challenges and opportunities for concept drift detection. Through the utilization of deep learning techniques, we aim to parse and analyze such data, detecting any underlying concept drift. Furthermore, by integrating data across multiple modalities, we aspire to develop robust models capable of functioning in diverse environments and scenarios. Therefore, the handling and understanding of multimodal data will form a critical component of our future work.

- **Enhancement of Performance Metrics:** While our current implementation of metrics for tracking concept drift has shown efficacy, there is a pressing need for a more universal and well-rounded performance assessment framework in the field. To date, the community has primarily relied on detection latency as a key indicator, but this singular dimension inadequately captures the multifaceted nature of concept drift detection. Therefore, future work should be dedicated to the development of a more comprehensive set of metrics. This framework should encapsulate a broader range of dimensions including, but not limited to, detection latency, false positive rate, false negative rate, and computational efficiency. A robust, holistic metric system will ensure a thorough evaluation of detection outcomes, pushing the boundaries of concept drift detection research.

- **Scalability and Efficiency Optimization in Real-World Contexts:** The experimental datasets employed in our studies, while significant, still present a considerable discrepancy compared to the sheer volume and velocity of real-world data streams. As such, a key challenge in future work is to rigorously test and measure the effectiveness of our methodologies in large-scale, rapidly evolving scenarios. Consequently, this necessitates an in-depth investigation into the scalability of our methods, alongside a concerted effort to optimize computational

efficiency. Our ambition is to empower these methodologies to perform real-time or near real-time concept drift detection within substantial and swiftly changing data environments typical of real-world scenarios. Achieving this would further underscore the practical utility of our concept drift detection methods.

- In light of the current research, future work with QuadCDD should focus on addressing certain limitations and areas for improvement. One aspect to consider is the adaptive fine-tuning of the model, including the adjustment of hyper-parameters to better accommodate various data streams with concept drift. This would enhance the model's adaptability and overall performance in handling diverse types of concept drifts.

  Furthermore, the drift type (D_t) is not extensively utilized in QuadCDD. Future research should explore additional potential applications of D_t to facilitate a deeper understanding of concept drift and its various manifestations. By addressing these limitations and expanding on the current framework, we can continue to advance the field of data stream analysis and develop more robust and effective models for handling concept drift in real-world applications.

- **Exploiting Deep Learning Techniques:** The promising outcomes achieved through the application of deep learning techniques in concept drift detection signal potential for further exploration. Upcoming research could focus on developing sophisticated deep learning architectures, refining existing models, and harnessing transfer learning and other strategies to augment model efficiency.

Addressing concept drift remains pivotal in adaptive machine learning systems, despite the challenges it poses. The ever-changing nature of data necessitates the development of robust, resilient machine learning models that can accommodate these shifts. The future of machine learning algorithms is significantly promising, particularly those that are capable of effectively handling concept drift. Such advancements will not only enhance predictive model reliability in evolving environments but also spur the development of sophisticated, adaptable systems. Ultimately, continued research in this domain is critical for the progression of machine learning and data science.

References

Ackermann, M. R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C. and Sohler, C. (2010), 'Streamkm++: A clustering algorithm for data streams', ACM J. Exp. Algorithmics **17**.

Alghushairy, O., Alsini, R., Soule, T. and Ma, X. (2020), 'A review of local outlier factor algorithms for outlier detection in big data streams', Big Data Cogn. Comput. **5**, 1.

Alippi, C., Boracchi, G. and Roveri, M. (2013), 'Just-in-time classifiers for recurrent concepts', IEEE Transactions on Neural Networks and Learning Systems **24**, 620–634.

Alippi, C., Boracchi, G. and Roveri, M. (2017), 'Hierarchical change-detection tests', IEEE Transactions on Neural Networks and Learning Systems **28**(2), 246–258.

Alkassar, S., Abdullah, M. A. M., Jebur, B. A., Abdul-Majeed, G. H., Wei, B. and Woo, W. L. (2021), 'Automated diagnosis of childhood pneumonia in chest radiographs using modified densely residual bottleneck-layer features', Applied Sciences **11**(23).

Andonovski, G., Lughofer, E. and Škrjanc, I. (2021), 'Evolving fuzzy model identification of nonlinear wiener-hammerstein processes', IEEE Access **9**, 158470–158480.

Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R. and Morales-Bueno, R. (2006), Early drift detection method, in 'Fourth international workshop on knowledge discovery from data streams', Vol. 6, pp. 77–86.

Baier, L., Schlör, T., Schöffer, J. and Kühl, N. (2021), 'Detecting concept drift with neural network model uncertainty', CoRR **abs/2107.01873**.

Basseville, M. and Nikiforov, I. V. (1993), Detection of Abrupt Changes: Theory and Application, Prentice-Hall, Inc., USA.

Bifet, A. and Gavalda, R. (2007), Learning from time-changing data with adaptive windowing, in 'Proceedings of the 2007 SIAM international conference on data mining', SIAM, pp. 443–448.

Bifet, A., Holmes, G., Kirkby, R. and Pfahringer, B. (2010), 'MOA: massive online analysis', J. Mach. Learn. Res. **11**, 1601–1604.

Bland, J. M. and Altman, D. (1986), 'Statistical methods for assessing agreement between two methods of clinical measurement', The lancet **327**(8476), 307–310.

Blažič, S. and Škrjanc, I. (2020), 'Incremental fuzzy c-regression clustering from streaming data for local-model-network identification', IEEE Transactions on Fuzzy Systems **28**(4), 758–767.

Buckland, M. and Gey, F. (1994), 'The relationship between recall and precision', Journal of the American society for information science **45**(1), 12–19.

Casilari, E., Lora-Rivera, R. and García-Lagos, F. (2020), 'A study on the application of convolutional neural networks to fall detection evaluated with multiple public datasets', Sensors **20**(5), 1466.

Chen, D., Yang, Q., Liu, J. and Zeng, Z. (2020), 'Selective prototype-based learning on concept-drifting data streams', Information Sciences **516**, 20–32.

Chen, Q., Chen, Z., Nafa, Y., Duan, T., Pan, W., Zhang, L. and Li, Z. (2022), 'Adaptive deep learning for entity resolution by risk analysis', Knowledge-Based Systems p. 110118.

Csiszár, O., Pusztaházi, L. S., Dénes-Fazakas, L., Gashler, M. S., Kreinovich, V. and Csiszár, G. (2023), 'Uninorm-like parametric activation functions for human-understandable neural models', Knowledge-Based Systems **260**, 110095.

Dawid, A. P. and Vovk, V. G. (1999), 'Prequential probability: principles and properties', Bernoulli **5**(1), 125 – 162.

Diez-Olivan, A., Ortego, P., Del Ser, J., Landa-Torres, I., Galar, D., Camacho, D. and Sierra, B. (2021), 'Adaptive dendritic cell-deep learning approach for industrial prognosis under changing conditions', IEEE Transactions on Industrial Informatics pp. 1–1.

Ditzler, G., Roveri, M., Alippi, C. and Polikar, R. (2015), 'Learning in nonstationary environments: A survey', IEEE Computational Intelligence Magazine **10**(4), 12–25.

Dongre, P. B. and Malik, L. G. (2014), A review on real time data stream classification and adapting to various concept drift scenarios, in '2014 IEEE International Advance Computing Conference (IACC)', pp. 533–537.

dos Reis, D. M., Flach, P., Matwin, S. and Batista, G. (2016), Fast unsupervised online drift detection using incremental kolmogorov-smirnov test, in 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 1545–1554.

Du, H., Minku, L. L. and Zhou, H. (2020), Marline: Multi-source mapping transfer learning for non-stationary environments, in '2020 IEEE International Conference on Data Mining (ICDM)', pp. 122–131.

Du, L., Song, Q. and Jia, X. (2014), 'Detecting concept drift: An information entropy based method using an adaptive sliding window', Intell. Data Anal. **18**(3), 337–364.

Dubuc, T. T., Stahl, F. T. and Roesch, E. B. (2021), 'Mapping the big data landscape: Technologies, platforms and paradigms for real-time analytics of data streams', IEEE Access **9**, 15351–15374.

Fang, Z., Li, Y., Lu, J., Dong, J., Han, B. and Liu, F. (2022), 'Is out-of-distribution detection learnable?'.

Fang, Z., Lu, J., Liu, F., Xuan, J. and Zhang, G. (2020), 'Open set domain adaptation: Theoretical bound and algorithm', IEEE transactions on neural networks and learning systems **PP**.

Feng, G. (2019), Concept Drift Detection for Detection Machine Learning with Stream Data, PhD thesis, University of Technology Sydney.

Flach, P. (2012), Machine learning: the art and science of algorithms that make sense of data, Cambridge university press.

Frías-Blanco, I., Campo-Ávila, J. d., Ramos-Jiménez, G., Morales-Bueno, R., Ortiz-Díaz, A. and Caballero-Mota, Y. (2015), 'Online and non-parametric drift detection methods based on hoeffding's bounds', IEEE Transactions on Knowledge and Data Engineering **27**(3), 810–823.

Gaber, M. M. (2012), 'Advances in data stream mining', Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **2**.

Gama, J., Medas, P., Castillo, G. and Rodrigues, P. (2004), Learning with drift detection, in 'Brazilian symposium on artificial intelligence', Springer, pp. 286–295.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. and Bouchachia, A. (2014), 'A survey on concept drift adaptation', ACM computing surveys (CSUR) **46**(4), 1–37.

Gao, B., He, Y., Lok Woo, W., Yun Tian, G., Liu, J. and Hu, Y. (2016), 'Multidimensional tensor-based inductive thermography with multiple physical fields for offshore wind turbine gear inspection', IEEE Transactions on Industrial Electronics **63**(10), 6305–6315.

Gu, X., Angelov, P. and Zhao, Z. (2021), 'Self-organizing fuzzy inference ensemble system for big streaming data classification', Knowledge-Based Systems **218**, 106870.

Guo, H., Zhang, S. and Wang, W. (2021), 'Selective ensemble-based online adaptive deep neural networks for streaming data with concept drift', Neural Networks **142**, 437–456.

Halstead, B., Koh, Y. S., Riddle, P. J., Pears, R., Pechenizkiy, M., Bifet, A., Olivares, G. and Coulson, G. (2021), 'Analyzing and repairing concept drift adaptation in data stream classification', Machine Learning pp. 1–35.

Hamad, R. A., Yang, L., Woo, W. L. and Wei, B. (2022), 'Convnet-based performers attention and supervised contrastive learning for activity recognition', Applied Intelligence pp. 1–17.

Hamad, R., Kimura, M., Yang, L., Woo, W. L. and Wei, B. (2021), 'Dilated causal convolution with multi-head self attention for sensor human activity recognition', Neural Computing and Applications **33**.

Haque, A., Khan, L. and Baron, M. (2016), Sand: Semi-supervised adaptive novel class detection and classification over data stream, in 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 30.

Harel, M., Mannor, S., El-Yaniv, R. and Crammer, K. (2014), Concept drift detection through resampling, in 'International Conference on Machine Learning'.

Hashemi, S. and Yang, Y. (2009), 'Flexible decision tree for data stream classification in the presence of concept change, noise and missing values', Data Min. Knowl. Discov. **19**, 95–131.

Hashemi, S., Yang, Y. and Kangavari, M. (2007), To better handle concept change and noise: A cellular automata approach to data stream classification, in M. Orgun and J. Thornton, eds, 'Proc, 20th Aust. Joint Conf. on AI: Advances in AI', Vol. 4830, Springer-Verlag London Ltd., Germany, pp. 669 – 674.

Hernández, M. and Stolfo, S. (1998), 'Real-world data is dirty: Data cleansing and the merge/purge problem', Data Min. Knowl. Discov. **2**, 9–37.

Hu, B., Gao, B., Woo, W. L., Ruan, L., Jin, J., Yang, Y. and Yu, Y. (2021), 'A lightweight spatial and temporal multi-feature fusion network for defect detection', IEEE Transactions on Image Processing **30**, 472–486.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L. and Muller, P.-A. (2019), 'Deep learning for time series classification: a review', Data Mining and Knowledge Discovery **33**(4), 917–963.

Iwashita, A. S. and Papa, J. P. (2019), 'An Overview on Concept Drift Learning', IEEE Access **7**, 1532–1547.

Jain, L., Katarya, R. and Sachdeva, S. (2020), 'Opinion leader detection using whale optimization algorithm in online social network', Expert Systems with Applications **142**, 113016.

Jr., F. J. M. (1951), 'The kolmogorov-smirnov test for goodness of fit', Journal of the American Statistical Association **46**(253), 68–78.

Kaluža, B., Mirchevska, V., Dovgan, E., Luštrek, M. and Gams, M. (2010), An agent-based approach to care in independent living, in 'Proceedings of the First International Joint Conference on Ambient Intelligence', AmI'10, Springer-Verlag, Berlin, Heidelberg, p. 177–186.

Kammerer, K., Pryss, R., Hoppenstedt, B., Sommer, K. and Reichert, M. (2020), 'Process-driven and flow-based processing of industrial sensor data', Sensors **20**(18).

Kaur, A. and Jagdev, G. (2017), Analyzing working of fp-growth algorithm for frequent pattern mining.

Kauschke, S., Lehmann, D. H. and Fürnkranz, J. (2019), Patching deep neural networks for nonstationary environments, in '2019 International Joint Conference on Neural Networks (IJCNN)', pp. 1–8.

Kelly, M. G., Hand, D. J. and Adams, N. M. (1999), The impact of changing populations on classifier performance, in 'Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '99, Association for Computing Machinery, New York, NY, USA, p. 367–371.

Klinkenberg, R., Viii, L. I., Renz, I. and Ag, D.-B. (1998), Adaptive information filtering: Learning in the presence of concept drifts.

Korycki, L. and Krawczyk, B. (2021), Class-incremental experience replay for continual learning under concept drift, in 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 3649–3658.

Kovic, M. and Kovic, M. (2017), 'Why i write with latex (and why you should too)'.
URL: https://medium.com/@marko_kovic/why-i-write-with-latex-and-why-you-should-too-ba6a764fadf9

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J. and Woźniak, M. (2017), 'Ensemble learning for data stream analysis: A survey', Information Fusion **37**, 132–156.

Krawczyk, B., Stefanowski, J. and Wozniak, M. (2015), 'Data stream classification and big data analytics', Neurocomputing **150**, 238–239.

L. Lobo, J., Del Ser, J. and Herrera, F. (2021), 'Lunar: Cellular automata for drifting data streams', Information Sciences **543**, 467–487.

Laborieux, A., Ernoult, M., Hirtzlin, T. and Querlioz, D. (2021), 'Synaptic metaplasticity in binarized neural networks', Nature communications **12**(1), 1–12.

Lamport, L. (1994), LATEX: a document preparation system: user's guide and reference manual, Addison-wesley.

Leite, D., Andonovski, G., Škrjanc, I. and Gomide, F. (2020), 'Optimal rule-based granular systems from data streams', IEEE Transactions on Fuzzy Systems **28**(3), 583–596.

Leite, D. and Gomide, F. (2020), 'An overview on evolving systems and learning from stream data', Evolving Systems **X**, 1–18.

Li, J., Jiang, H., Mei, X., Hu, C. and Zhang, G. (2020), 'Dynamical analysis of rumor spreading model in multi-lingual environment and heterogeneous complex networks', Information Sciences **536**, 391–408.

Lin, C.-C., Deng, D.-J., Kuo, C.-H. and Chen, L. (2019), 'Concept drift detection and adaption in big imbalance industrial iot data using an ensemble learning method of offline classifiers', IEEE Access **7**, 56198–56207.

Liu, A., Lu, J., Liu, F. and Zhang, G. (2018), 'Accumulating regional density dissimilarity for concept drift detection in data streams', Pattern Recognition **76**, 256–272.

Liu, A., Lu, J., Song, Y., Xuan, J. and Zhang, G. (2022), 'Concept drift detection delay index', IEEE Transactions on Knowledge and Data Engineering pp. 1–1.

Liu, A., Song, Y., Zhang, G. and Lu, J. (2017), Regional concept drift detection and density synchronized drift adaptation, in 'International Joint Conference on Artificial Intelligence'.

Liu, F., Yu, Y., Song, P., Fan, Y. and Tong, X. (2020), 'Scalable kde-based top-n local outlier detection over large-scale data streams', Knowledge-Based Systems **204**, 106186.

Liu, S., Feng, L., Wu, J., Hou, G. and Han, G. (2017), 'Concept drift detection for data stream learning based on angle optimized global embedding and principal component analysis in sensor networks', Computers & Electrical Engineering **58**, 327–336.

Liu, Z., Loo, C. K. and Seera, M. (2019), 'Meta-cognitive recurrent recursive kernel os-elm for concept drift handling', Applied Soft Computing **75**, 494–507.

Long, M., Cao, Z., Wang, J. and Jordan, M. I. (2018), Conditional adversarial domain adaptation, in 'Proceedings of the 32nd International Conference on Neural Information Processing Systems', NIPS'18, Curran Associates Inc., Red Hook, NY, USA, p. 1647–1657.

Losing, V., Hammer, B. and Wersing, H. (2016), Knn classifier with self adjusting memory for heterogeneous concept drift, in '2016 IEEE 16th International Conference on Data Mining (ICDM)', pp. 291–300.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. and Zhang, G. (2019), 'Learning under concept drift: A review', IEEE Transactions on Knowledge and Data Engineering **31**(12), 2346–2363.

Lu, N., Lu, J., Zhang, G. and Lopez de Mantaras, R. (2016), 'A concept drift-tolerant case-base editing technique', Artificial Intelligence **230**, 108–133.

Luengo, J., Sánchez-Tarragó, D., Prati, R. C. and Herrera, F. (2021), 'Multiple instance classification: Bag noise filtering for negative instance noise cleaning', Information Sciences **579**, 388–400.

Lughofer, E. (2017), 'On-line active learning: A new paradigm to improve practical useability of data stream modeling methods', Information Sciences **415-416**, 356–376.

Lughofer, E. and Angelov, P. (2011), 'Handling drifts and shifts in on-line data streams with evolving fuzzy systems', Applied Soft Computing **11**(2), 2057–2068. The Impact of Soft Computing for the Progress of Artificial Intelligence.

Lughofer, E., Pratama, M. and Skrjanc, I. (2018), 'Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation', IEEE Transactions on Fuzzy Systems **26**(4), 1854–1865.

Lughofer, E., Pratama, M. and Škrjanc, I. (2021), 'Online bagging of evolving fuzzy systems', Information Sciences **570**, 16–33.

Lughofer, E., Weigl, E., Heidl, W., Eitzinger, C. and Radauer, T. (2016), 'Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances', Information Sciences **355-356**, 127–151.

Luo, Q., Gao, B., Woo, W. and Yang, Y. (2019), 'Temporal and spatial deep learning network for infrared thermal defect detection', NDT & E International **108**, 102164.

Mahan, F., Mohammadzad, M., Rozekhani, S. M. and Pedrycz, W. (2021), 'Chi-mflexdt:chi-square-based multi flexible fuzzy decision tree for data stream classification', Applied Soft Computing **105**, 107301.

Mahdavi, E., Fanian, A., Mirzaei, A. and Taghiyarrenani, Z. (2022), 'Itl-ids: Incremental transfer learning for intrusion detection systems', Knowledge-Based Systems **253**, 109542.

Manias, D. M., Shaer, I., Yang, L. and Shami, A. (2021), Concept drift detection in federated

networked systems, in '2021 IEEE Global Communications Conference (GLOBECOM)', IEEE Press, p. 1–6.

Masud, M. M., Gao, J., Khan, L. and Han, J. (2009), Classifying evolving data streams for intrusion detection.

Mittelbach, F., Goossens, M., Braams, J., Carlisle, D. and Rowley, C. (2004), The LATEX companion, Addison-Wesley Professional.

Montiel, J., Halford, M., Mastelini, S. M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H. M., Read, J., Abdessalem, T. and Bifet, A. (2021), 'River: Machine learning for streaming data in python', J. Mach. Learn. Res. **22**(1).

Montiel, J., Read, J., Bifet, A. and Abdessalem, T. (2018), 'Scikit-multiflow: A multi-output streaming framework', Journal of Machine Learning Research **19**(72), 1–5.

Orozco-Arias, S., Piña, J. S., Tabares-Soto, R., Castillo-Ossa, L. F., Guyot, R. and Isaza, G. (n.d.), 'Measuring performance metrics of machine learning algorithms for detecting and classifying transposable elements'.

Page, E. S. (1954), 'Continuous inspection schemes', Biometrika **41**(1/2), 100–115.

Pang, Z.-H., Fan, L.-Z., Sun, J., Liu, K. and Liu, G.-P. (2021), 'Detection of stealthy false data injection attacks against networked control systems via active data modification', Information Sciences **546**, 192–205.

Pesaranghader, A. and Viktor, H. L. (2016), Fast hoeffding drift detection method for evolving data streams, in P. Frasconi, N. Landwehr, G. Manco and J. Vreeken, eds, 'Machine Learning and Knowledge Discovery in Databases', Springer International Publishing, Cham, pp. 96–111.

Pesaranghader, A., Viktor, H. L. and Paquet, E. (2017), 'Mcdiarmid drift detection methods for evolving data streams', 2018 International Joint Conference on Neural Networks (IJCNN) pp. 1–9.

Pratama, M., Lu, J., Lughofer, E., Zhang, G. and Er, M. J. (2017), 'An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks', IEEE Transactions on Fuzzy Systems **25**(5), 1175–1192.

Priya, S. and Uthra, R. A. (2021), 'Deep learning framework for handling concept drift and class imbalanced complex decision-making on streaming data', Complex & Intelligent Systems pp. 1–17.

Pukelsheim, F. (1994), 'The three sigma rule', The American Statistician **48**(2), 88–91.

Rezk, H., Aly, M. and Fathy, A. (2021), 'A novel strategy based on recent equilibrium optimizer to enhance the performance of pem fuel cell system through optimized fuzzy logic mppt', Energy **234**, 121267.

Rohlf, F. J., Sokal, R. R. et al. (1995), Statistical tables, Macmillan.

Ryan, S., Corizzo, R., Kiringa, I. and Japkowicz, N. (2019), 'Deep learning versus conventional learning in data streams with concept drifts', 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA) pp. 1306–1313.

Schweppe, H., Zimmermann, A. and Grill, D. (2010), 'Flexible on-board stream processing for automotive sensor data', IEEE Transactions on Industrial Informatics **6**(1), 81–92.

Shaker, A. and Lughofer, E. (2014), 'Self-adaptive and local strategies for a smooth treatment of drifts in data streams', Evolving Systems **5**, 239–257.

Sobolewski, P. and Wozniak, M. (2013), 'Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors.', J. Univers. Comput. Sci. **19**(4), 462–483.

Song, Y., Zhang, G., Lu, H. and Lu, J. (2019), A noise-tolerant fuzzy c-means based drift adaptation method for data stream regression, in '2019 IEEE International Conference on Fuzzy Systems', pp. 1–6.

Souza, V. M., dos Reis, D. M., Maletzke, A. G. and Batista, G. E. (2020), 'Challenges in benchmarking stream learning algorithms with real-world data', Data Mining and Knowledge Discovery **34**(6), 1805–1858.

Teinemaa, I., Dumas, M., Leontjeva, A. and Maggi, F. M. (2018), 'Temporal stability in predictive process monitoring', Data Mining and Knowledge Discovery **32**, 1306–1338.

Teinemaa, I., Dumas, M., Rosa, M. L. and Maggi, F. M. (2019), 'Outcome-oriented predictive process monitoring: Review and benchmark', ACM Trans. Knowl. Discov. Data **13**(2).

Tengtrairat, N., Woo, W. L., Parathai, P., Aryupong, C., Jitsangiam, P. and Rinchumphu, D. (2021), 'Automated landslide-risk prediction using web gis and machine learning models', Sensors **21**(13).

Tengtrairat, N., Woo, W. L., Parathai, P., Rinchumphu, D. and Chaichana, C. (2022), 'Non-intrusive fish weight estimation in turbid water using deep learning and regression models', Sensors **22**(14).

Tharwat, A. (n.d.), 'Classification assessment methods'.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P. and SciPy 1.0 Contributors (2020), 'SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python', Nature Methods **17**, 261–272.

Wang, H. and Abraham, Z. (2015), Concept drift detection for streaming data, in '2015 International Joint Conference on Neural Networks (IJCNN)', pp. 1–9.

Wang, P., Jin, N., Davies, D. and Woo, W. L. (2023), 'Model-centric transfer learning framework for concept drift detection', Knowledge-Based Systems **275**, 110705.

Wang, P., Jin, N., Woo, W. L., Woodward, J. R. and Davies, D. (2022), 'Noise tolerant drift detection method for data stream mining', Information Sciences **609**, 1318–1333.

Wang, S., Minku, L. L. and Yao, X. (2018), 'A systematic study of online class imbalance learning with concept drift', IEEE Transactions on Neural Networks and Learning Systems **29**(10), 4802–4821.

Wares, S., Isaacs, J. P. and Elyan, E. (2019), 'Data stream mining: methods and challenges for handling concept drift', SN Applied Sciences **1**.

Wei, L., Liu, L., Qi, J. and Qian, T. (2020), 'Rules acquisition of formal decision contexts based on three-way concept lattices', Information Sciences **516**, 529–544.

Widmer, G. and Kubat, M. (1994), 'Learning in the presence of concept drift and hidden contexts', Machine Learning **23**.

Wu, X., Zhu, X., Wu, G.-Q. and Ding, W. (2014), 'Data mining with big data', IEEE Transactions on Knowledge and Data Engineering **26**(1), 97–107.

Xiong, H., Pandey, G., Steinbach, M. and Kumar, V. (2006), 'Enhancing data analysis with noise removal', IEEE Transactions on Knowledge and Data Engineering **18**(3), 304–319.

Xu, Y., Sun, G., Geng, T. and He, J. (2019), 'Low-energy data collection in wireless sensor networks based on matrix completion', Sensors **19**(4), 945.

Xu, Y. and Wilson, K. (2021), Early alert systems during a pandemic: A simulation study on the impact of concept drift, in 'LAK21: 11th International Learning Analytics and Knowledge Conference', LAK21, Association for Computing Machinery, New York, NY, USA, p. 504–510.

Yang, L. and Shami, A. (2021), 'A lightweight concept drift detection and adaptation framework for iot data streams', IEEE Internet of Things Magazine **4**(2), 96–101.

Yang, Y., Zheng, X., Guo, W., Liu, X. and Chang, V. (2019), 'Privacy-preserving smart iot-based healthcare big data storage and self-adaptive access control system', Information Sciences **479**, 567–592.

Yu, H., Zhang, Q., Liu, T., Lu, J., Wen, Y. and Zhang, G. (2022), 'Meta-add: A meta-learning based pre-trained model for concept drift active detection', Information Sciences **608**, 996–1009.

Yu, S. and Abraham, Z. (2017), Concept drift detection with hierarchical hypothesis testing, in 'Proceedings of the 2017 SIAM International Conference on Data Mining', SIAM, pp. 768–776.

Yu, S., Abraham, Z., Wang, H., Shah, M., Wei, Y. and Príncipe, J. C. (2019), 'Concept drift detection and adaptation with hierarchical hypothesis testing', Journal of the Franklin Institute **356**(5), 3187–3215.

Yu, S., Wang, X. and Príncipe, J. (2018), Request-and-reverify: Hierarchical hypothesis testing for concept drift detection with expensive labels, in 'IJCAI'.

Zhang, Y., Gao, B., Yang, D., Woo, W. L. and Wen, H. (2022), 'Online learning of wearable sensing for human activity recognition', IEEE Internet of Things Journal **9**(23), 24315–24327.

Zyblewski, P., Sabourin, R. and Woźniak, M. (2021), 'Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams', Information Fusion **66**, 138–154.

Škrjanc, I. (2020), 'Cluster-volume-based merging approach for incrementally evolving fuzzy gaussian clustering—egauss+', IEEE Transactions on Fuzzy Systems **28**(9), 2222–2231.
■Škrjanc et al.

Škrjanc, I., Iglesias, J. A., Sanchis, A., Leite, D., Lughofer, E. and Gomide, F. (2019), 'Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey', Information Sciences **490**, 344–368.