UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Handling Concept Drift Using the Correlation between Multiple Data Streams

by Bin Zhang

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Dist. Prof. Jie Lu and A/Prof. Guangquan Zhang

University of Technology Sydney
Faculty of Engineering and Information Technology

October 2022

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Bin Zhang*, declare that this thesis is submitted in fulfillment of the requirements for the award of *Doctor of Philosophy*, in the *Faculty of Engineering and Information Technology* at the University of Technology Sydney

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

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

This research is supported by the Australian Research Council.

Signature: Production Note:
Signature removed prior to publication.

Date: October 24th, 2022

ABSTRACT

Machine learning has been widely applied to handle big data. In real-world applications, data are in the form of streams. Streaming data bring new challenges to machine learning. Concept drift is a major problem in handling streaming data. The newly arrived data have a different distribution from historical data. Hence, machine learning models do not work on the newly arrived data. Many methods have been proposed to detect whether concept drift occurs and to update the machine learning models to the drift.

To date, the research on concept drift considers data streams separately. That is, these methods handle data streams one by one, ignoring whether correlations exist between data streams. However, in real-world applications, correlations between data streams are widespread. If the correlations between data streams were available, this would support better decision making, rather than handling data streams separately. There is currently no research on how to model the correlations between data streams. Data streams are dynamic. Therefore, it is necessary to develop methods to track the correlations between data streams and to adapt to changes in correlations. In addition, after concept drift is detected, there is usually insufficient data to train a new model, which can lead to the over-fitting problem. How to deal with the over-fitting issue is still a challenge. Motivated by this, this research proposes several methods to overcome the aforementioned challenges.

To alleviate the over-fitting problem, a concept drift adaptation method, Drift Adaptation via Joint Distribution Alignment (DAJDA), is proposed. DAJDA performs a linear transformation to the drift instances instead of modifying the model. Instances are transformed into a common feature space, reducing the discrepancy of distributions before and after drift. Using additional historical data to train a new model can lead to better performance due to the increasing training set. Experimental studies show that DAJDA has the ability to improve the performance of the learning model under concept drift.

To model the correlations between data streams, we propose a novel Multi-stream Concept Drift Handling Framework via data sharing, containing fuzzy membership-based drift detection (FMDD) and fuzzy membership-based drift adaptation (FMDA) components, to train the new learning model for drifting streams by sharing weighted data from other non-drifting streams. A stream fuzzy set is defined with membership functions that measure the degree to which samples belong to a data stream. Our Concept Drift Handling Framework can detect when and in which streams concept drift occurs, and therefore, the over-fitting issue can be solved by adding the weighted data from non-drifting streams to train new learning models. Synthetic and real-world experiment results show that our method can help avoid the over-fitting issue caused by a lack of data and thereby significantly improve the prediction performance.

To track changes in correlations and adapt to correlation drift, we propose an ensemble chain-structured model, Evolutionary Regressor Chains. To provide the model with the ability to search the optimal order of the chain, we design a heuristic order searching strategy to be incorporated as part of the model's ongoing process. The heuristic order searching strategy can also update the chains as time passes, to track the dynamicity of the correlations. A diversity pruning method is also proposed to reduce computation complexity while retaining the diversity of the ensemble. We undertake a theoretical analysis, and give the dynamic regret bound of our method. Our experiment results show that our Evolutionary Regressor Chains method can track data stream correlations accurately. Chains can update themselves to adapt to both concept drift and correlation drift. The performance of the machine learning models on data streams is improved.

To learn meta knowledge across multiple data streams, a concept drift adaptation strategy - Learning to Fast Adapt in the Evolving Environment - is proposed for neural network classifiers in the non-stationary environment. A meta-level LSTM recurrent neural network is used to learn a proper parameter updating rule instead of updating methods that are traditional gradient descent-based (e.g. stochastic

gradient descent) during fine-tuning. A suitable updating step will be generated according to current loss and its gradient for the parameter of a neural network classifier. Experiments on both synthetic and real-world data sets show that our method can quickly adapt the neural network classifier to concept drift and help improve the performance of the classifier in a dynamic environment.

Dedication

I dedicate my dissertation work to my parents Xia Xiang and Xuxin Zhang.

Acknowledgements

It has been an indelible journey to complete my doctoral degree at the esteemed University of Technology Sydney. I am profoundly grateful to all those who have extended their assistance in myriad ways throughout this remarkable endeavor.

Foremost, I wish to express my sincerest appreciation to my principal supervisor, the highly esteemed Distinguished Professor Jie Lu. Her unwavering guidance has been an illuminating beacon, navigating me towards the successful culmination of my research. In addition to fostering an exceptional research environment, she bestowed upon me a generous scholarship that served as a solid bedrock for my four-year inquiry. Under her tutelage, I delved into uncharted research domains, identifying and addressing critical gaps with her expert counsel. Her unwavering dedication to scholarship served as a constant source of inspiration and solace during challenging times. Beyond research, her invaluable professional advice on life and potential career trajectories has shaped my path with utmost clarity. Moreover, her pursuit of innovation and unwavering commitment to scholarly rigor has left an indelible mark on my academic journey.

I also extend my deepest gratitude to my esteemed co-supervisor, Associate Professor Guangquan Zhang. His unwavering support and invaluable suggestions pertaining to literature review, manuscript writing, and paper revision have been instrumental in shaping the trajectory of my doctoral study. It is his trust, encouragement, and unwavering assistance that have played an integral role in my successful completion of this demanding program.

Joining the Decision Systems & e-Service Intelligence Lab has been an absolute pleasure, providing me with an enriching experience alongside numerous enthusiastic and dedicated colleagues. I wish to express my heartfelt gratitude to Dr. Anjin

Liu, Dr. Yiliao Song, Dr. Hang Yu, Dr. Bin Wang, and my fellow peers who have generously offered their guidance and suggestions throughout the course of my doctoral candidature.

Additionally, I would like to extend my sincerest appreciation to my parents, whose unwavering support has been a constant source of strength regardless of my location or pursuits. I am forever indebted to their encouragement and unwavering belief in my abilities. To my dear friends Donglai Yang and Shiyu Ma, I cherish the memories of our shared experiences and remain deeply grateful for your unwavering companionship during the challenging times imposed by the Covid-19 pandemic. I would like to express special appreciation to my best friend, Zihe Liu, whose presence has brought immeasurable joy and laughter to my life. Our bond is truly extraordinary, and I am immensely thankful for the camaraderie we share.

In summary, the successful completion of my doctoral journey would not have been possible without the support, guidance, and unwavering belief of those mentioned above. Their collective contributions have left an indelible mark on my academic and personal growth, and for that, I am eternally grateful.

Bin Zhang

Sydney, Australia, 2022.

Contents

List of Figures

List of Tables

Abbreviation

ARF: Adaptive Random Forest

DAJDA: Drift Adaptation via Joint Distribution Alignment

DDM: Drift Detection Method

ECDD: EWMA for Concept Drift Detection

ERC: Ensemble Regressor Chains

FMDA: Fuzzy Membership-based Drift Adaptation

FMDD: Fuzzy Membership-based Drift Detection

HAT: Heoffding Adaptive Tree

HDDM: Hoeffding's inequality based Drift Detection Method

HT: Heoffding Tree

JDA: Joint Distribution Adaptation

MMD: Maximum Mean Discrepancy

MOA: Massive Online Analysis

MTRS: Multi-target Regression Stacking

NSW: New South Wales

PAC: Probably approximately Correct learning

PCA: Principal Component Analysis

RC: Regressor Chains

SEA: SEA Moving Hyperplane Concepts

ST: Single Target

VIC: Victoria

Nomenclature and Notation

$X_t^{(i)}$ is the feature at time t of the ith stream.

$y_t^{(i)}$ is the label at time t of the ith stream.

S_i is the ith stream.

$P(\cdot)$ is the probability.

$E(\cdot)$ is the expect.

Chapter 1

Introduction

1.1 Background

It is witnessed that machine learning has achieved great success in handling big data. Massive amounts of streaming data are generated by smart phones, social networks, sensors, etc. Streaming data are different from other data, which are generated in sequence. Data streams introduce new challenges for machine learning models, including concept drift. Traditional machine learning algorithms are offline. Once the models have been trained using the given historical data set, they do not change after the training process ends. However, the world is dynamically changing and streaming data are not always stationary. New data might have a different distribution from historical data. Taking a recommender system as an example, customer preferences change over time, depending on the availability of alternatives, the inflation rate, and even personal emotions. It is impossible to consider all these factors in machine learning models. New methods are needed to recognize the patterns underlying these evolving streaming data. The phenomenon where the distribution change over time in data streams was first defined as concept drift in Widmer and Kubat (1996). Concept drift has been a major issue that influences the behavior of real-world machine learning algorithms in a dynamic and evolving environment. The phenomenon of concept drift is common in a number of fields, such as computer or telecommunication systems, traffic monitoring, system following personal interests, medical decision aiding, etc (Gama et al., 2014).

New challenges are introduced with the increased use of mobile devices and the

rise of the Internet of Things (IoT). Multiple sensors and mobile devices generate multiple data streams at the same time, known as multi-streams. The consequent unavoidable, rapid change in environments inevitably results in changes in the underlying data distribution in almost all data streams. These streams may be related. For example, a large-scale forest is monitored in real time by multiple devices, e.g. ground towers, unmanned aerial vehicles, and satellites. Multi-stream data is generated all the time. Each stream is composed of several channel signals. Significant changes in device signals as a result of drift may be an indication of fire in the monitored section, and the different sensitivities and the geographical relationships of multiple devices may mean that fires have broken out either simultaneously or sequentially. To quickly recognise drift in all devices and to generate early drift warning before fires take hold would support timely decision making to reduce damage to property and even save lives.

Most methods which handle concept drift in data streams follow a framework, which is shown in Figure 1.1. To deal with concept drift problems in an evolving environment, we first have to detect whether drift has occurred at all. Drift detection is vitally important but not simple. On the one hand, detection algorithms are supposed to be able to differentiate between concept drift and normal noise. On the other hand, drift detection methods should be fast enough so that we can update the model in a timely manner. Concept drift detection methods can be categorized into three groups: error rate-based methods, data distribution-based methods and multiple hypothesis test methods. Once drift has been detected, we have to update our learning models to suit the changes to the incoming data, which is called concept drift adaptation. Over the last few decades, several methods have been proposed to handle learning adaptation under concept drift. Most methods reconstruct the whole model when concept drift is detected. The whole model is eliminated and a new one is established. However, global adaptation strategies are a waste of computation.

Some methods leverage tree-based algorithms where each node separates the feature space into several parts and each leaf node represents a hyper-rectangle in the feature space. These methods identify the region where concept drift occurs and replace the model only in the drift region. Though local adaptation methods have the flexibility to adjust parts of rather than the whole model, they are restricted to a specific base learner. Furthermore, the popular methodology of ensemble learning has been used to address concept drift problems.

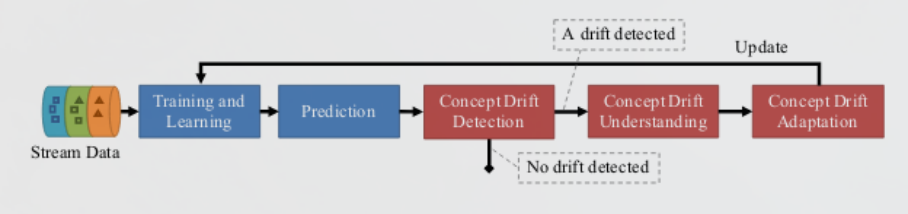


Figure 1.1 : Framework for handling concept drift

Although these methods have achieved great success in handling concept drift, they can only be applied to single data streams. In real-world applications, data streams exist concurrently. These methods handle streams separately. The correlation between multiple data streams are ignored. Motivated by this, our research aims to handle concept drift in a multi-stream scenario, tracking the correlation between data streams and better adapting to concept drift using the correlation.

1.2 Research Questions and Objectives

In Section 1.1, we described the concept drift problem in data stream mining. Motivated by the challenge of how to adapt machine learning models in a multi-stream non-stationary environment, four research questions are posed. In this section, we first list the four research questions that focus on the emergent problems in machine learning under concept drift. Then, we identify the research objectives corresponding to the research questions.

1.2.1 Research Questions

This study seeks to answer the following four research questions (RQ):

RQ1: How to model the correlation between two data streams?

RQ2: How to model the correlation between multiple data streams?

RQ3: How to track changes to the correlation between multiple data streams?

RQ4: How to adapt a machine learning model according to the correlation between multiple data streams?

1.2.2 Research Objectives

To answer these questions, we identified the following corresponding research objectives (RO).

RO1: To develop a methodology to model the correlation between two data streams. (to answer RQ1)

From a single stream to multi-streams, defining the correlation between data streams is the first step of our research. The correlation between data streams need to be investigated so it is possible to use knowledge from other data streams. Given two data streams, it is obvious that they should be handled separately if there is no correlation between them. Once the correlation is recognized, it is possible to use knowledge from one data stream to help handle concept drift in the other data stream. Therefore, how to model the correlation between two data streams is of vital importance.

RO2: To develop a network structure to model the correlation between multiple data streams. (to answer RQ2)

If we have multiple data streams, knowing the correlation between each pair of data streams is not enough. The correlation between each pair of data streams should be maintained in a graph structure. The graph is fundamental to our research. In the graph, each data stream is represented as a node. The edge of two nodes indicates the correlation between the two respective data streams. An intuitive illustration of all correlation of each pair of data streams can be given in the graph.

RO3: To develop an update strategy of the correlation graph between multiple data streams. (to answer RQ3)

Data streams are unstable and the correlation between streams in real-world scenarios is also unstable. The correlation between two data streams may disappear, while isolated nodes in the correlation graph may be connected. If the correlation become inaccurate, the corresponding drift adaptation consequences will start to deteriorate. Therefore, the correlation graph should have the ability to adapt to capture the new correlation.

RO4: To develop an adaption strategy for data streams using the correlation between data streams. (to answer RQ4)

With **RO1, RO2** and **RO3**, we can react to concept drift in one data stream using information from other data streams. If concept drift occurs in one node of the correlation graph, we can use information from the other connected nodes in the graph to help drift adaptation. In particular, if recurrent drift occurs in one data stream, i.e. an old concept reappears after a period of time when concept drift occurs, the correlation can remind the other data stream to reuse the old model. In

addition, since the two data streams are related, the newly arrived data of one data stream may help to construct new models for the other data stream.

1.3 Research Significance

In this section, we explain the significance and innovation of this research from the perspective of theory and application, based on the research contents.

1.3.1 Theoretical Significance

This research will make significant theoretical contributions to the concept drift field by addressing a very challenging issue: how to learn the correlation between two data streams. This is the fundamental component of this research. If this problem can be solved, we can construct a correlation graph to maintain the correlation between all data streams.

We also investigate how to react to concept drift using knowledge learnt from other data streams. This will help concept drift adaptation in a situation where newly arrived data do not have labels and will therefore save computation resources.

1.3.2 Practical Significance

In this research, we develop a correlation graph between multiple data streams. Multi-stream data is common in the real world, such as sensor data or marketing data. The correlation graph will help to quickly react to unavoidable drift in multiple data streams and generate integrated decision support.

1.4 Research Contributions

The contributions of this research are summarized as follows:

- It proposes a novel concept drift adaptation method which can overcome the insufficient training problem caused by scarce newly arrived data. We train

the classifier on a latent feature space using knowledge learnt from historical data to make predictions on the newly arrived data.

- It proposes a novel multi-stream Concept drift Handling Framework, which considers the correlation between multiple data streams rather than handling data streams separately.
  - The advantage of the framework is that the parameters of the membership functions are estimated using data from other streams. These data which have different distributions help to reach higher concept drift detection accuracy.
  - A new drift detection method, FMDD, is designed to detect when and in which streams concept drift occurs, dividing streams into drifting and non-drifting streams at each time. The parameters of the fuzzy membership functions are estimated using data from other data streams, which lead to a remarkably high true positive rate.
  - A new drift adaptation method, FMDA, is proposed using the correlation of multiple data streams to train a new model after concept drift is detected. By increasing the volume of training data, the over-fitting issue due to a lack of data is alleviated.
- It proposes a chain-structured model, Evolutionary Regressor Chains, to track the correlation among multiple data streams.
  - We firstly take the correlation between more than two data streams into account to improve the performance of the models in data stream regression, which has not been solved by existing research.
  - To overcome the drawback that the randomly generated chain order cannot track the correlation correctly, we design a heuristic order searching

strategy. The method updates the order iteratively to find an optimal order.

  - We also design an online updating strategy to update the models in the multi-stream regression scenario. This strategy can effectively adapt to both concept drift and correlation drift.
  - We propose a diversity pruning method to decrease the complexity of our method while maximizing the diversity of the ensemble. It can also increase the robustness of our method.
  - We analyze some theoretical properties of our method and give its dynamic regret bound.

- It proposes a concept drift adaptation strategy for neural network classifiers in the evolving environment. A parameter updating rule based on meta-level LSTM recurrent neural network replaces traditional gradient descent-based rules while rapidly adapting the classifier.

The publications related to this research are listed as follows:

Published

1. **B. Zhang**, J. Lu and G. Zhang, "Drift Adaptation via Joint Distribution Alignment," *2019 IEEE 14th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pp. 498-504, 2019.

Under Review

2 **B. Zhang**, J. Lu, Y. Song, G. Zhang, "A Multi-stream Concept Drift Handling Framework via Data Sharing," *IEEE Transactions on Cybernetics*.

3 **B. Zhang**, J. Lu, G. Zhang, A. Liu, X. Yao, "Evolutionary Regressor Chains for Multi-stream Regression," *IEEE Transactions on Cybernetics*.

4 **B. Zhang**, J. Lu, G. Zhang, "Learning to Fast Adapt in the Evolving Environment," *Nurocomputing*.

1.5 Thesis Organization

This thesis is organized as follows:

- **Chapter 2**: This chapter presents a literature review related to this research. Current research can be divided into two categories: concept drift detection and concept drift adaptation. In this chapter, concept drift detection methods are reviewed first. Then we provide a survey about concept drift adaptation. Finally, we review several related research areas, including transfer learning, multi-output learning and meta learning.

- **Chapter 3**: This chapter proposes a concept drift adaptation method, Drift Adaptation via Joint Distribution Alignment (DAJDA). The method performs a linear transformation to the drift instances instead of modifying the model. Instances are transformed into a common feature space, reducing the distribution discrepancy before and after concept drift. Experimental studies show that DAJDA has the ability to improve the performance of the learning model under concept drift.

- **Chapter 4**: This chapter proposes a novel multi-stream Concept Drift Handling Framework via data sharing. The framework contains fuzzy membership-based drift detection (FMDD) and fuzzy membership-based drift adaptation (FMDA) components, to train the new learning model for drifting streams by sharing weighted data from other non-drifting streams. A stream fuzzy set is defined with membership functions that measure the degree to which samples belong to a data stream. Our Concept Drift Handling Framework can detect when and in which streams concept drift occurs, and therefore, the over-fitting

issue can be solved by adding weighted data from non-drifting streams to train new learning models. Synthetic and real-world experiment results show that our method can help avoid the over-fitting issue caused by a lack of data and thereby significantly improve the prediction performance.

- **Chapter 5**: This chapter reports on the development of an ensemble chain-structured model, Evolutionary Regressor Chains, to track the correlation among data streams. To provide the model with the ability to search for the optimal order of the chain, we design a heuristic order searching strategy to be incorporated as part of the model's ongoing process. The heuristic order searching strategy can also update the chains as time passes, to track the dynamicity of the correlation. A diversity pruning method is also proposed to reduce the computational complexity while retaining the diversity of the ensemble. We undertake a theoretical analysis, and give the dynamic regret bound of our method. Our experiment results show that our Evolutionary Regressor Chains method can track data stream correlation accurately. Chains can update themselves to adapt to both concept drift and correlation drift. The performance of the machine learning models on data streams is improved.

- **Chapter 6**: In this chapter, a concept drift adaptation strategy - Learning to Fast Adapt in the Evolving Environment (LFAEE) - is proposed for neural network classifiers in the non-stationary environment. A meta-level LSTM recurrent neural network is used to learn a proper parameter updating rule instead of updating methods that are traditional gradient descent-based (e.g. stochastic gradient descent) during fine-tuning. A suitable updating step is generated according to current loss and its gradient for the parameter of a neural network classifier. Experiments on both synthetic and real-world data sets show that our method can quickly adapt the neural network classifier to

concept drift and help to improve the performance of the classifier in a dynamic environment.

- **Chapter 7**: A brief summary of the thesis contents and its contributions are given in the final chapter. There is also a discussion about research issues for further study.

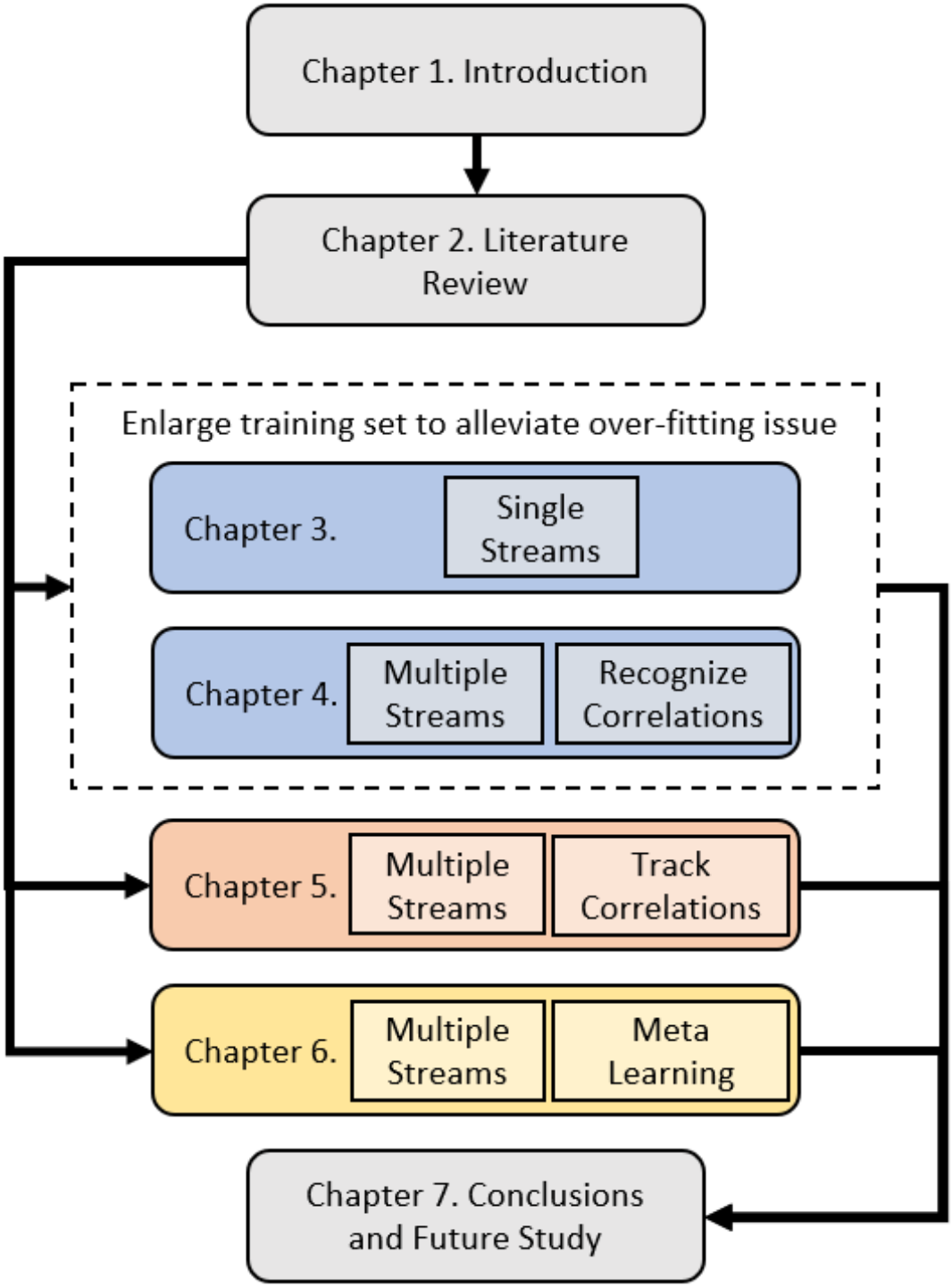


Figure 1.2 : The structure of the thesis.

Chapter 2

Literature Review

This chapter presents a survey of literature related to this research. As shown in Figure 1.1, Current literature focusing on concept drift can be divided into two categories: concept drift detection and concept drift adaptation. In Section 2.1, we give some setting and definitions. In Section 2.2, methods to detect concept drift are reviewed. In Section 2.3, we give a review of previous concept drift adaptation methods. Finally, other works related to this research are reviewed in Section 2.4, including transfer learning, multi-output learning and meta learning.

2.1 Setting and Definitions

Data streams are non-stationary. Given a data stream $S = \{(X_1, y_1), (X_2, y_2), \ldots, (X_t, y_t), \ldots\}$, concept drift refers to $\exists t_1, t_2$, such that the distributions

$$P(X_{t_1}, y_{t_1}) \neq P(X_{t_2}, y_{t_2}).$$

The distribution of data has changed over time. Considering the conditional probability, we have

$$P(X_t, y_t) = P(X_t)P(y_t|X_t).$$

Therefore, two types of concept drift should be distinguished: real concept drift and virtual concept drift Gama et al. (2014). The difference of two types of concept drift is illustrated in Figure 2.1. The virtual concept drift refers to situations where only the marginal probability $P(X_t)$ changes. The real concept drift refers to situations where the conditional probability $P(y_t|X_t)$ changes.

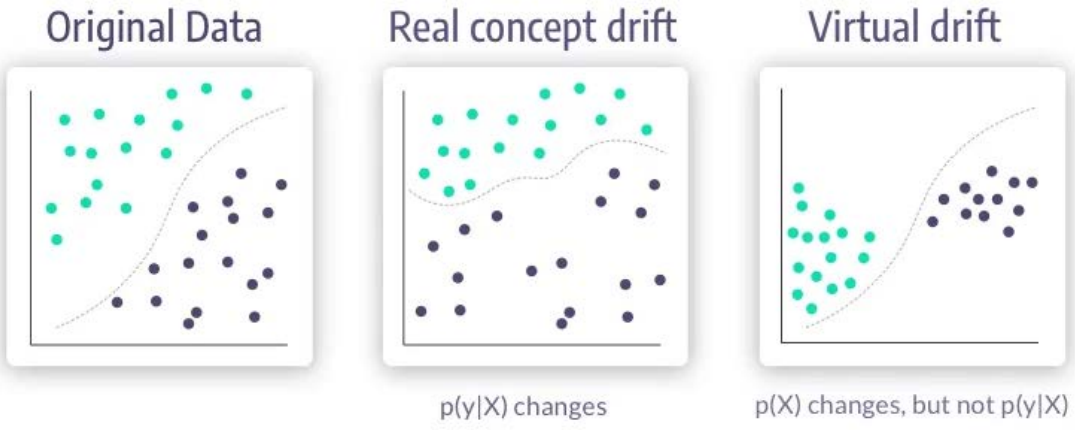


Figure 2.1 : Illustration of two types of concept drift.

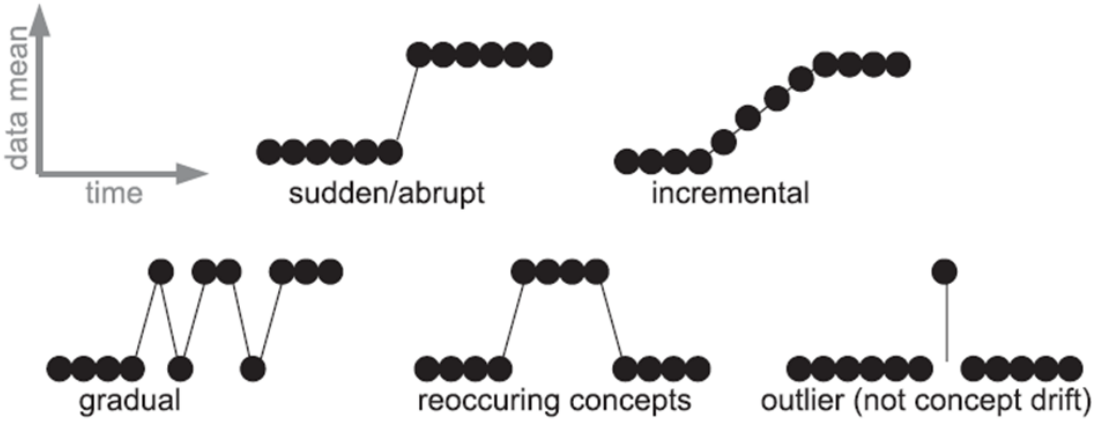


Figure 2.2 : Four patterns of changes in data distributions and outlier.

Changes in data distributions could be different. Four patterns of changes in data distributions are given in Gama et al. (2014), which is shown in Figure 2.2. The abrupt drift refers to that the change in data distribution is severe and rapid. The incremental drift refers to that the change in data distribution is slight and lasts for a period of time. The change in gradual drift is severe but data with changed distribution coexistence with the old data in a period of time. The reoccuring drift refers to the former distribution re-occurs after concept drift. It should be distinguished that the outlier is not concept drift.

2.2 Concept Drift Detection

Concept drift detection refers to the techniques and mechanisms to identify the change points or change time intervals of the distribution of the data Basseville and Nikiforov (1993). Next, we will introduce the famous algorithms of three kinds of detection methods.

2.2.1 Detection Methods based on Error Rate

The idea of error rate-based detection algorithms is very simple. These methods monitor the on-line error rate of the machine learning model and alarm once an increase of error rate is proven to be statistically significant. The first error rate-based algorithm, Drift Detection Method (DDM), was proposed in Gama et al. (2004). DDM define a warning level and a drift level for the error rate. When new instance is observed, if the error rate reaches the warning level, DDM will generate a warning signal. The next incoming data will be stored in a time window. A new model will be trained using data in the time window. If the error rate decreases afterwards, this signal will be considered as a false alarm and the new model will be discarded. If the error rate still increases and reaches the drift level, the new model will replace the old one.

Some improved algorithms were proposed afterwards, such as Learning with Local Drift Detection Gama and Castillo (2006), Early Drift Detection Method Baena-Garcıa et al. (2006), Heoffding's inequality based Drift Detection Method Frías-Blanco et al. (2015), Fuzzy Windowing Drift Detection Method Liu et al. (2017), Dynamic Extreme Learning Machine Xu and Wang (2017), Dynamic Classifier Selection Pinagé et al. (2020), etc.

Statistical Test of Equal Proportions Detection (STEPD), proposed in Nishida and Yamauchi (2007), is another error rate-based drift detection algorithm. Different

from DDM, STEPD compares incoming data with overall historical data using two time windows. However, the size of time window is defined by the users. Different sizes of time window may have different results. Bifet and Gavaldà (2007) developed a mechanism, Adaptive Windowing (ADWIN), to determine the size of time window adaptively. ADWIN does not require users to define the size of the time window. It traverses all possible time window size and computes the optimal window size based on the rate of change between two compared time windows.

STUDD Cerqueira et al. (2022) is proposed for the unsupervised setting. In STUDD, a teacher-student learning paradigm is performed, and the mimicking loss is monitored to detect whether concept drift occurs.

In addition, the robustness and efficacy of concept drift detection methods under poisoning attacks are investigated in Korycki and Krawczyk (2022a). A robust framework of concept drift detection is proposed and is evaluated under adversarial and poisoning attacks.

2.2.2 Detection Methods based on Data Distribution

The second category of drift detection methods is data distribution based detection algorithms. The idea of data distribution based detection algorithms is quite relevant to the two sample test problem. The two sample test is a classical problem in statistic area. According to Gretton et al. (2006), two sample test refer to the problem to determine that whether samples from two probability distributions are from the same distribution actually. Data distribution based detection algorithms deal with the same issue. Comparing the incoming data with the historical data, we have to determine whether they are from the same distribution. If not, we say that the concept has drifted. Data distribution based detection methods firstly define a distance function to measure the distance between two distributions. Distance functions used commonly include Chi-square distance, the Kullback-Leibler diver-

gence, the Hellinger distance, the Kolmogorov-Smirnov distance, Maximum Mean Discrepancy, etc Vayatis et al. (2009). Based on the distance function, a statistic is constructed. If the divergence between two samples is proven to be statistically significant, the concept drift is detected and the model need to be updated.

The first data distribution based detection algorithm was proposed in Kifer et al. (2004). This paper measures the discrepancy of distributions by so called Relativized Discrepancy. The distance of two distributions is defined as

$$d = \int |f_1(X) - f_2(x)| dx,$$

where f_1 and f_2 are the density functions of the two distributions. According to Krawczyk et al. (2017), similar algorithms include Wald-Wolfowitz test, two-sample Kolmogorov-Smirnov test, Wilcoxon rank sum test, two-sample t-test, etc. Hellinger Distance Drift Detection Method Ditzler and Polikar (2011) uses histograms to model the marginal distributions and compares the marginal distributions of two groups of samples. Change Detection Test Bu et al. (2018) uses RBF-networks to approximate the density functions. A fuzzy competence model is used in Lu et al. (2014) to model the distributions. In Dasu et al. (2006), kdqTrees are used to partion the feature space, then a Kullback-Leibler divergence is used to test whether the two groups of data are different. Similar method is proposed in Liu et al. (2018, 2021a), with a k-Means model to replace the kdqTrees. In Lobo et al. (2021), a cellular automata is used to monitor the distribution of data. The distribution is represented in the grid of the cellular automata. A paired student test is used to test the distribution of two groups of samples in Wang et al. (2021). In Hinder et al. (2020), a mathematical equivalence of the presence of concept drift and the dependency of a specific random variable is given. Based on the distribution of the random variable, a non-parametric independence test is performed for concept drift detection. Behavioral constraint templates are used in De Smedt et al. (2020) to detect concept drift in sequence analysis. An unsupervised discriminative

classifier Gözüaçık et al. (2019) is learned to discriminate whether the distribution has changed. Based on how much modification of the models after updating, a novel concept drift detection method is proposed in Yang et al. (2020). An adaptive minimax risk classifiers is proposed in Álvarez et al. (2022) for tracking multi-dimensional change over time.

CNF Density Estimation test proposed in Dries and Rückert (2009) is another data distribution based detection methods. Different from algorithms above, CNF Density Estimation test map the data into a binary feature space. Every attribute in the binary space only have two value, true and false. Each incoming instance is also represented as a binary vector. A Mann-Whitney test is processed to determine whether the difference between two samples is statistically significant. In Webb et al. (2018), a new setting, concept drift mapping, is introduced. Several quantitative concept drift mapping techniques are proposed for analysis of concept drift in data streams.

Compared with error rate-based detection algorithms, detection algorithms based on data distribution have its advantages and disadvantages. These algorithms address the concept drift problem based on the data distribution itself. Hence they are more accurate than error-based detection algorithms. However, error rate-based detection algorithms focus only on the error rate, while data distribution based detection algorithms need to calculate a new, complex statistic. As a consequence, the latter has higher time complexity than the former. As we discussed in the Section 2.1.1, data distribution-based or error rate-based, this is also a trade-off between accuracy and efficiency. In addition, it has been a problem in data distribution-based algorithms that which data is supposed to be chosen as the new data and which to be chosen as the historical data. In another word, how to construct the two samples in the two-sample test problem is still an open question.

2.2.3 Detection Methods based on Multiple Hypothesis Test

Multiple hypothesis test drift detection methods use similar techniques with those mentioned in the above sections. The difference is that multiple hypothesis test detection algorithms use multiple hypothesis tests to detect whether the concept drift has occurred. These algorithms can be divided into two groups: parallel multiple hypothesis test algorithms and hierarchical multiple hypothesis test algorithms.

The first parallel multiple hypothesis test algorithm, Just-In-Time adaptive classifier was proposed in Alippi and Roveri (2008). The authors gave four configurations, the features extracted by Principle Component Analysis (PCA), PCA extracted features plus one generic component of the original features, drift in each original feature and drift in all possible combinations of the feature space. Whether the concept drift occur is determined by all the four configurations. A similar algorithm is proposed in Wang and Abraham (2015). Linear Four Rate drift detection. Zhang et al. (2017) detects the label drift, feature space drift and decision boundary drift respectively based on Information value and Jaccard similarity.

Usually, the hierarchical multiple hypothesis test algorithms use an existing algorithm to detect whether a drift occurs, called the detection layer. Then another mechanism is developed to verify the detection process, called validation layer. Alippi et al. (2017) proposed the first hierarchical multiple hypothesis test drift detection algorithm, Hierarchical Change-Detection Trees. Existing drift detection algorithms chosen in detection layer is supposed to have a low drift rate and low computational cost. As we discussed above, error rate-based detection methods are competent. Whether the validation layer is activated depends on the results returned by the detection layer. Now that the accuracy is more import than efficiency in the validation layer, data distribution-based algorithms have more advantages over error rate-based

ones. The authors suggested two methods for validation layer: estimating the distribution of the test statistics by maximizing the likelihood or using an existing algorithms such as the Kolmogorov-Smirnov test or Cramer-Von Mises test. Similar algorithms include Two-Stage Multivariate Shift-Detection based on EWMA Raza et al. (2015), Hierarchical Linear Four Rate Yu et al. (2019) and Request-and-Reverify Yu et al. (2018). DriftSurf Tahmasbi et al. (2021) combine multiple drift detection methods, and divide into two types, stable-state and reactive-state. Different states are applied with different strategies.

2.3 Concept Drift Adaptation

This section concentrates on the strategies of updating learning models according to the drift.

2.3.1 Retraining Models

The most straightforward method to react to the concept drift is just to retrain a new machine learning model with the new coming data. Once a drift signal has been detected, a new model is supposed to be trained to replace the old models.

A window strategy is often utilized in these methods to preserve both the old data and the new data for drift detection and models retraining. Paired Learners Bach and Maloof (2008) followed this strategy and uses two learners, a stable learner for predicting and a reactive learner. If a drift has been detected, the stable learner is replaced by the reactive learner trained by the latest data. This method is simple and easy to implement. However, Bach and Maloof (2008) did not give the answer that how to determine the size of the window. A small size can reflect the information of the latest data, while a large one provides more data for training a new model. A popular window scheme approach is ADWIN Bifet and Gavaldà (2007). This paper gives an adaptive method to determine the size of the window.

Instead of directly retraining the model, some methods integrate the detection with the retraining process for some machine learning algorithms. DELM Xu and Wang (2017) extends the traditional ELM algorithm to address the concept drift by adaptively adjusting the number of the nodes of the hidden layer. When the error rate increases significantly, more nodes are added to the hidden layer of the network to improve the accuracy of the learning model. A parallel version of ELM-based method Han et al. (2015) has been proposed to deal with the high speed classification problems under concept drift.

Instance-based lazy learners for handling concept drift have also been investigated. The Just-in-time adaptive classifier Alippi et al. (2017) removes the old instance from the case base if a concept drift has been detected.

2.3.2 Adaptive Models

Rather than retraining the entire learner, adaptive models has abilities to partially update themselves when the concept drift is detected. This kind of method are more efficient when the drift occurs in only local region.

Numerous tree-based algorithms belong to this kind of adaptation methods because of the ability of tree model that could examine and adapt to each leaf node, i.e. local region of the whole feature space, separately. Very Fast Decision Tree classifier Domingos and Hulten (2000) uses Hoeffding bound to limit the number of instances required for node splitting. This approach does not store instances in memory and the cost of tree maintenance is very low. Hence VFDT is very popular. Hulten et al. (2001) extends the VFDT, develops a sliding window to maintain the latest data. A sub-tree is trained based on the data in the sliding window. The sub-tree will replace the nodes in the VFDT once the sub-tree outperforms its original counterpart. VFDTc Gama et al. (2003) is another improved version of VFDT. In VFDTc, when a drift is detected on a node, the node becomes a leaf node and its

descending sub-tree is removed. In Losing et al. (2018), a local statistics is used to predict the split time, rather than the global splitting scheme, which avoids unnecessary split-attempts. Usually, the global splitting scheme is of high computation complexity. The enhanced VFDT shows advantage on run-time without a loss of prediction performance. Another enhanced VFDT, Hoeffding Anytime Tree is proposed in Manapragada et al. (2018). It shows significantly superior prequential accuracy over the VFDT with only a small additional computation cost. Chordal Kernel Decision Tree Krawczyk (2021) is proposed for tensor data stream. The Chordal distance is used to calculate the similarities between tensors. A concept drift detection method is proposed based on the tensor represention. ERulesD2S Cano and Krawczyk (2019), a rule-based method, are proposed providing with more interpretability. Like tree-based methods, new rules are generated to adapt to concept drift.

Except for tree-based models, researchers proposed other adaptive models. KNN-PAW Bifet et al. (2013) uses a probabilistic adaptive window to maintain both the recent and older samples. SAM-kNN Losing et al. (2016) uses a long-term memory to store historical data and a short-term memory to store rencent data. NEFCS Lu et al. (2016) is a kNN-based adaptive model. NEFCS utilizes the output of the competence model-based drift detection method to locate instances in the case base that are affected by the concept drift, and a redundancy removal algorithm to remove redundant instances in a uniform way. A drift region-based data sample filtering method is propoed in Dong et al. (2022) to identify the region where concept drift occurs, and update only parts of the models.

Two varieties of support vector Machines (SVM) are proposed in Gâlmeanu and Andonie (2022); Yu et al. (2022a). The SVMs have ability to incrementally learn from recent data to adapt to concept drift.

In addition, the problem for learning discrete-time Markov chains under concept drift is investigated in Roveri (2019). A framework containing both detection and adaptation methods is proposed to address the problem.

Deep learning has achieved great success in various applications. The deep neural networks are updated by gradient descent-based optimization methods. It can be extend to data stream setting to update themselves online. A pre-trained convolutional neural network is proposed in Soleymani and Paquet (2020), and the parameters of the model is updated online to adapt to the concept drift. Then it is preferred that to use a meta learner to accelerate the updating in Ryan et al. (2019). A new online deep learning framework is proposed in Sahoo et al. (2018). A novel hedge backpropagation method is used to adapt to concept drift and update the parameters effectively. Other structure of deep models are also investigated to adapt to concept drift, including recurrent neural networks Fekri et al. (2021) and LSTM Yen et al. (2019). To avoid the high computation cost of updating the whole deep models, some methods update only parts of the parameters. In Yang et al. (2019); Kirkpatrick et al. (2017), a fisher information matrix is maintained to avoid the catastrophic forgetting problem. In Disabato and Roveri (2019), a two-layer hypothesis test is performed to test in which layer of the of the convolutional neural network the concept drift occurs and update the layer only.

Besides updating parameters, some methods update the structure of the neural networks to adapt to the concept drift. In Guo et al. (2021), a selective ensemble-based online adaptive deep neural network is proposed, provided with adaptive combination of shallow and deep features. In Rusu et al. (2016); Budiman et al. (2016), new branches are added to networks when concept drift occurs. However, this leads to high computational complexity. A Deep Evolving Denoising Autoencoder Ashfahani et al. (2020) has a flexible structure, where hidden units can be deleted or added to adapt to concept drift. It is pointed out that the model show

poor performance on unseen data. Thus it is assumed that the data follows the Gaussian distribution. Situations where the assumption is relaxed by Autonomous Gaussian Mixture Model are discussed in Pratama et al. (2019a,b). Some methods even adjust the depth of the neural networks to adapt to the concept drift Ashfahani and Pratama (2019); Pratama et al. (2020b). An overview of handling concept drift in deep learning is given in Yuan et al. (2022).

Besides convention neural networks, a new generation of artificial neural network, spiking neural network, is proposed. The spiking neural networks are considered to have ability to easily and fast adapt to non-stationary environment. In Lobo et al. (2018), Evolving Spiking Neural Networks are introduced for learning in non-stationary data streams. An overview of applying spiking neural networks in online learning is given in Lobo et al. (2020).

2.3.3 Adaptive Ensembles

Ensemble learning has received much attention in machine learning area in recent years Gomes et al. (2017a). Ensemble methods comprise a set of base classifiers that may have different types or different parameters. The output of each base classifier is combined using certain voting rules to predict the newly arrived data. Many adaptive ensemble methods have been developed that aim to deal with concept drift by extending classical ensemble methods or by creating specific adaptive voting rules.

Bagging, Boosting and Random Forests are classical ensemble methods used to improve the performance of single classifiers. They have all been extended for handling concept drift problems in streaming data. An online version of the bagging method was first proposed in Oza and Russell (2001) which uses each instance only once to simulate the batch mode bagging. In a later study Bifet et al. (2010a), this method was combined with the ADWIN drift detection algorithm Bifet and

Gavaldà (2007) to handle concept drift. When a concept drift is reported, the newly proposed method, called Leveraging Bagging, trains a new classifier on the latest data to replace the existing classifier with the worst performance. Similarly, an adaptive boosting method was developed in Chu and Zaniolo (2004) which handles concept drift by monitoring prediction accuracy using a hypothesis test, assuming that classification errors on non-drifting data should follow Gaussian distribution. In a recent work Gomes et al. (2017b), the Adaptive Random Forest (ARF) algorithm was proposed, which extends the random forest algorithm with a concept drift detection method, such as ADWIN Bifet and Gavaldà (2007), to decide when to replace an old tree with a new one. A similar work can be found in Li et al. (2015), which uses Hoeffding bound to distinguish concept drift from noise within trees.

Besides extending classical methods, many new ensemble methods have been developed to handle concept drift using novel voting techniques. Dynamic Weighted Majority (DWM) Kolter and Maloof (2007) is such an ensemble method that is capable of adapting to drifts with a simple set of weighted voting rules. It manages base classifiers according to the performance of both the individual classifiers and the global ensemble. If the ensemble misclassifies an instance, DWM will train a new base classifier and add it to ensemble. If a base classifier misclassifies an instance, DWM reduces its weight by a factor. When the weight of a base classifier drops below a user defined threshold, DWM removes it from the ensemble. The drawback of this method is that the adding classifier process may be triggered too frequently, introducing performance issues on some occasions, such as when gradual drift occurs. An Adaptive Chunk-based Dynamic Weighted Majority is proposed in Lu et al. (2020) later. Different sizes of chunks are used for training to sovle the imbalanced label problem. In order to deal with the imbalance problem, a very fast oversampling strategy, VFC-SMOTE, is proposed in Bernardo and Della Valle (2021), and a novel

Adaptive Rebalancing algorithm is proposed in Malialis et al. (2021). A well-known ensemble method, Learn++.NSE Elwell and Polikar (2011), mitigates this issue by weighting base classifiers according to their prediction error rate on the latest batch of data. If the error rate of the youngest classifier exceeds 50%, a new classifier will be trained based on the latest data. Learn++.NSE is a passive ensemble learning model which does not identify when a drift occurs. It assumes that data are incrementally received in batches. For each incoming data batch, this algorithm creates a new base classifier, and then dynamically adjusts each existing classifier voting weight based on its time-adjusted accuracy on the latest data batch. This method has several other benefits: it can easily adopt almost any base classifier algorithm; it does not store history data, only the latest batch of data, which it only uses once to train a new classifier; and it can handle sudden drift, gradual drift, and recurrent drift because underperforming classifiers can be reactivated or deactivated as needed by adjusting their weights. In Kappa Updated Ensemble Cano and Krawczyk (2020), A Kappa statistic is used for dynamic weighting the classifiers. The classifiers are trained on different subset of features for higher diversity.

A patching strategy is proposed in Kauschke and Fürnkranz (2018). A new classifier will be learned only on the region where the prediction performance is poor. A novel repair algorithm is proposed in Halstead et al. (2022). The repair algorithm detects the errors in concept drift adaptation and corrects them. The TORNADO framework proposed in Pesaranghader et al. (2018) wraps not only classifiers, but concept drift detectors into an ensemble. A CAR measure is introduced to select best detector-classifier pair for predicting. An ensemble architecture is proposed in Korycki and Krawczyk (2022b) for dealing with concept drift with sparse labeled data. Diversity and Transfer-based Ensemble Learning Sun et al. (2018) maintains ensemble of models according to not only performance but also diversity. Historical data are reused via transfer learning. In Liu et al. (2021b), a new meausre of diversity

is proposed to maintain the ensemble of classifiers. An active learning strategy is proposed in Krawczyk and Cano (2019) in semi-supervised learning setting. Only some models of the ensemble are accessed to label query. In Shan et al. (2019), the threshold of label query change over time to fast adapt to the concept drift. Similar methodology is performed in Pratama et al. (2020a), which reduces the complexity of the ensemble while keeping the diversity. In Zhao et al. (2020), a model reuse strategy is adopted to handle concept drift. In addition, the generalization error and the regret bound are also given. A Robust Online Self-Adjusting Ensemble Cano and Krawczyk (2022) is proposed to deal with the imbalance problem, by maintain a sliding window for each class. In Zhang et al. (2022), a reinforcement model is used for weighting the classifiers. A labeled data buffer is used to deal with the imbalance problem. A drift-gradient is defined in Song et al. (2022) to measure the increase of the distribution discrepancy. Then an adaptation method is proposed with low delay.

A number of research efforts have been made that focus on developing ensemble methods for handling concept drift of certain types. Accuracy Update Ensemble Brzezinski and Stefanowski (2014) was proposed with an emphasis on handling both sudden drift and gradual drift equally well. It is a batch mode weighted voting ensemble method based on incremental base classifiers. By doing re-weighting, the ensemble is able react quickly to sudden drift. All classifiers are also incrementally trained with the latest data, which ensures that the ensemble evolves with gradual drift. Two varieties of VIGO Nguyen et al. (2018) are proposed for abrupt and gradual concept drift respectively. The Optimal Weights Adjustment method Zhang et al. (2008) achieves the same goal by building ensembles using both weighted instances and weighted classifiers for different concept drift types. Evolutionary algorithms (Genetic Algorithm) are used in Ghomeshi et al. (2019) for adapting to different types of concept drift. In Chiu and Minku (2022), the diversity of ensemble

is used for dealing with different types of concept drift. A diversity ensemble can fast adapt to different types of concept drift. A special case of concept drift is considered in Sun et al. (2016), class evolution, the phenomenon of class emergence and disappearance. Recurring concepts are handled in Gama and Kosina (2014) and Gomes et al. (2014), which monitor concept information to decide when to reactivate previously stored obsolete models. Ahmadi and Kramer (2018) is another method that handles recurring concepts by refining the concept pool to avoid redundancy.

2.4 Other Related Work

2.4.1 Transfer Learning

Transfer learning has been an attractive research field recent years. A detailed review of transfer learning can be found in Pan and Yang (2010). The review divided the existing transfer learning algorithms into four categories: transferring knowledge of instances, transferring knowledge of feature representations, transferring knowledge of parameters and transferring relational knowledge. The feature-representation methods constitute major competent of transfer learning. The idea is to transfer data into a common space and training a common model. Thus, this paper only focuses on the second type. Some methods embed distributions as points in a Grassmann manifold, and generate a geodesic flow. This type of methods include Sampling Geodesic Flow Gopalan et al. (2011) and Geodesic Flow Kernel Gong et al. (2012). Some methods minimize the discrepancy of the distributions. Transfer Component Analysis (TCA) Pan et al. (2011) minimizes the distance of marginal distribution. Joint Distribution Adaptation (JDA) Long et al. (2013) improved the TCA and minimize both the marginal distribution and the conditional distribution jointly. Similar methods include Transfer Subspace Learning (TSL) Si et al. (2010) which replaces the Maximum Mean Discrepancy (MMD) by a Bregman divergences-based discrepancy.

Some methods reuse the data to train the models in target domain, which is named instance-transfer methods. Although the data from source domain cannot be used directly, parts of the data can be reused together with data in target domain. In TrAdaBoost Dai et al. (2007b), it is assumed that the source and target data use the same feature space, but the distributions are different. The data from source domain are weighted to reduce the effect of the "bad" source data and increase the effect of the "good" source data. Other instance-transfer methods include Liao et al. (2005); Dai et al. (2007a); Jiang and Zhai (2007).

Parameter-transfer methods make an assumption that models of related domains share the same parameters. Based on Gaussian Processes, MT-IVM Bonilla et al. (2007) aims to learn the parameters from multiple tasks. It is assumed that the prior distributions of Gaussian Processes of different tasks are the same. Other parameter-transfer methods include Schwaighofer et al. (2004); Evgeniou and Pontil (2004); Gao et al. (2008).

Relational-knowledge-transfer methods used prior knowledge to transfer across domains. The knowledge transferred is called relationship. TAMAR Mihalkova et al. (2007) assume that the source and target domains are related. A Markove Logic Network learned from source domain is used for learning in the target domain. Then two varieties Mihalkova and Mooney (2009); Davis and Domingos (2009) are proposed.

2.4.2 Fuzzy Systems in Machine Learning

Fuzzy systems have been widely used in machine learning research Deng et al. (2017); Lei et al. (2018, 2019, 2020). A small portion of information implies imprecision and vagueness. There exists clear correlation between the level of certainty and the amount of information Lu et al. (2015). Therefore, fuzzy systems have been applied in transfer learning research recently. In Behbood et al. (2011, 2013), a

fuzzy similarity measure is used to measure the distance between source and target domain. A survey of fuzzy systems in transfer learning is given in Lu et al. (2015).

Another area where fuzzy systems are used widely is clustering Song et al. (2020); Yu et al. (2022b). Fuzzy clustering is clustering methods where samples belong to two or more clusters. Because of its effectiveness and robustness, Fuzzy C-Means (FCM) Bezdek (2013) has been one of the most popular clustering methods. Compared with traditional nearest neighbors-based clustering methods, a sample belongs to all clusters rather than only a single cluster. Membership is used to measure how the sample belongs to the cluster. Some variants Zhou et al. (2021) are then developed to accelerate convergence or increase robustness. FRFCM proposed in Yang and Nataliani (2018), handling the fuzzy clustering problem in the high dimensional scenario using feature reduction. An ensemble fuzzy clustering method proposed in Rathore et al. (2018), EFCA, is designed for high dimensional clustering problem. In Shirkhorshidi et al. (2021), a data pruning technique is introduced for feature reduction in high dimensional fuzzy clustering. Besides FCM, new types of fuzzy clustering, such as anchor-based methods Nie et al. (2022) are attracting more attention recently.

2.4.3 Multi-Output Learning

Multi-output learning addresses the problem that the outputs of machine learning models are multiple. It can be divided into two categories according to tasks, multi-label classification and multi-target regression. A survey of multi-output learning is given in Zhang and Zhou (2014).

The key point of the multi-output learning problem is to track the correlation between multiple outputs, which is similar to tracking the correlation between multiple data streams. The idea of using a chain-structured model to track the correlation is first proposed in Read et al. (2011), and is more commonly referred to as Clas-

sifier Chains. The chain structure is very simple but useful nevertheless. The first label is predicted as per usual. Then the predicted label is used as a feature to predict the second label, and so on. All labels are predicted in sequence like a chain. Then two varieties, Probabilistic Classifier Chains Dembczyński et al. (2010) and Bayesian Classifier Chains Zaragoza et al. (2011), are proposed. The predicting of Classifier Chains is considered in a probabilistic and a Bayesian aspect respectively. It is pointed out that the order of chains is important in Read et al. (2011), but the solution to find a proper order is not given. It is suggested that using an ensemble of multiple chains will avoid obtaining a poorly ordered chain. No research has since focused on how to find the optimal order of the Classifier Chains.

As for the regression problem, the chain-structure method is adopted to address the multi-target regression problem in Spyromitros-Xioufis et al. (2016), known as the Regressor Chain. Notably, that both Probabilistic Classifier Chains and Bayesian Classifier Chains are feasible because the number of labels is finite. It is different from the regression problem because the target there is continuous. These two methods cannot be adopted directly to the regression problem. In Spyromitros-Xioufis et al. (2020); Read and Martino (2020), quantization strategies are used to constrain the continuous target to discrete values to transfer the problem into a discrete one.

2.4.4 Meta Learning

Another related research field is meta learning. In Schmidhuber (1987) it is firstly studied whether or not neural networks have the ability to modify their parameters automatically, which is considered as the first use of the term, meta learning, in computer science literature. Another common term refer to meta learning is "learning-to-learn" Thrun and Pratt (1998). Meta learning can be divided into three categories: model-based methods, optimization-based methods and metric learning

methods. A comprehensive survey about meta learning is given in Hospedales et al. (2022).

Optimization-based methods treat learning process as an optimization problem. A meta learner is used to improve part of the optimization procedure. A state-of-the-art example is Model-Agnostic Meta-Learning (MAML) Finn et al. (2017), using a neural network to generate proper initial parameters for very limited data, such that training from the generated parameters can converge quickly. Two improvement works Li et al. (2017); Antoniou et al. (2019) are proposed to learn proper step sizes, where a novelty parameter updating mechanism was proposed without using gradient. A more comprehensive of MAML is investigated in Li and Malik (2017). In Andrychowicz et al. (2016) a LSTM recurrent neural network is employed to generate proper parameter updating steps in batch learning. Then it is extended the application of this work to few shot learning problems in Ravi and Larochelle (2017).

Model-based methods, also known as black box methods, aim to embed the dataset into activation state with prediction for test data. Memory-augmented model, such as Neural Turing Machine Graves et al. (2014) and MANN Santoro et al. (2016) can rapidly encode new information, and make accurate predictions using the information. Compared with optimization-based methods, model-based methods have simple training procedure. However, model-based methods show poor performance on unseen tasks, and are efficient only when the size of datasets is small Finn and Levine (2018).

Metric learning methods refer to comparing the training points and the validation points and then predicting the matching training points. These methods include Koch et al. (2015); Vinyals et al. (2016); Sung et al. (2018).

The idea of meta learning was first introduced in few shot research in Lake

et al. (2016). A number of meta learning methods were proposed afterwards, including Vinyals et al. (2016); Finn et al. (2017); Ravi and Larochelle (2017).

2.5 Summary

In summary, although lots of methods have been proposed to handling concept drift in data streams, there are still the following research gaps exist in the previous research:

1. After concept drift is detected, the data is insufficient to train a good enough new model. Lots of methods are developed to solve the issue, but they are limited to specific model such as trees Bifet and Gavaldà (2009), kNN Losing et al. (2016) or ensemble models Sun et al. (2018). Few research concentrate on the general strategy.

2. Current research consider data streams separately, ignoring the correlation between multiple data streams. The correlation between data streams can improve the performance of machine learning models. However, no research focus on modeling the correlation between data streams.

3. Due to the dynamics of data streams, the correlation between data streams can change over time. Previous methods to handle concept drift cannot well tracking and handling the dynamic correlation between multiple data streams.

To fill the research gaps mentioned above, this research develop four methods. A drift adaptation method is proposed, in which historical data are used to augment the training set to train a better learning model. A fuzzy stream set is defined, with a fuzzy membership function to measure the correlation between data streams. The fuzzy membership is also used as weight of data to augment the training set to alleviate the over-fitting issue. An ensemble of regressor chains is proposed to track

the correlation between data streams and update itself to adapt to the correlation drift. A meta learning method is proposed to learn meta knowledge when data streams are absolutely different.

Chapter 3

Drift Adaptation via Joint Distribution Alignment

3.1 Introduction

Large-scale streaming data are generated as the development of the Internet. Statistical learning methods have shown advantages in recognizing the pattern hidden behind the data, and been applied in variety of fields, including email filtering, recommender systems, etc. Traditional statistical learning methods are under the stationary distribution assumption. which is not established in data stream cases. The distribution of the data changes by time, known as concept drift Widmer and Kubat (1996).

In the last decades, lots of concept drift adaptation methods have been proposed. A clear taxonomy of existing concept drift adaptation methods is given in Gama et al. (2014). Most methods reconstruct the classifier when concept drift is detected. However, drift only occurs in some regions rather than the whole feature space. Global adaptation strategies are waste of computation. Some methods leverage the property of tree-based algorithms that the feature space is separated as several hyper-rectangle and each one is represented by a leaf node. These methods identify the region where concept drift occurs and replace the classifier only in the drift region. Though local adaptation methods have flexibility to adjust rather than retrain the classifier, they are restricted to a specific base learner. You cannot have your cake and eat it too.

To break the learner limits of current local drift adaptation method, we divert the

focus from adjusting classifiers to transforming instances, and proposed a concept drift adaptation method based on JDA Long et al. (2013), named Drift Adaptation via Joint Distribution Alignment (DAJDA). JDA is well-known as an effective transfer learning method. Transfer learning hope to improve the learning model in a specific domain, which is usually noted as target domain, using the knowledge in a related source domain Pan and Yang (2010). Concept drift adaptation can be considered as to improve the learning model for newly arrived data using knowledge learnt from historical data.

JDA trains a classifier based on instances of source domain firstly and generates pseudo label on the instances of target domain. The discrepancy of different distributions is measured by Maximum Mean Discrepancy (MMD) Gretton et al. (2006). By jointly minimizing the MMD of both marginal distribution and conditional distribution, JDA gives the representation of instances from both domains in a new latent feature space. Different from transfer learning, concept drift adaptation usually assumes that the real label of the instance can be obtained a few moment after the prediction. As a consequence, DAJDA estimates the conditional distribution using real label. DAJDA reacts to the drift by transforming the data rather than modifying the model.

Our main contribution is to propose a novel concept drift adaptation method which can overcome the insufficient training problem caused by scarce newly arrived data. We train the classifier on a latent feature space using knowledge learnt from historical data to help predict on the newly arrived data. The rest of the chapter is organized as follows. In Section 3.2 we give a briefly reviews of concept drift adaptation and transfer learning. In Section 3.3, the detail of our method is given. In Section 3.4, we conducted several experiments to evaluate our approach. In Section 3.5, we summarized our proposed method.

3.2 Preliminaries

3.2.1 Maximum Mean Discrepancy

To measure the discrepancy of distributions of two groups of instances, MMD is introduced. MMD embeds each distribution into a Reproducing Kernel Hilbert Space. Let a linear transformation A be the kernel-induced map. Then we have the empirical estimate of MMD between $P(\mathbf{X})$ and $P(\mathbf{Y})$,

$$d_{MMD} = ||\frac{1}{n_1}\sum_{i=1}^{n_1} A\mathbf{X}_i - \frac{1}{n_2}\sum_{j=1}^{n_2} A\mathbf{Y}_j||^2_{\mathcal{H}}$$

where $||\cdot||_{\mathcal{H}}$ is the RKHS norm.

3.3 Proposed Method

We refer to the idea of the JDA to handle concept drift adaptation. We hope to find a linear transformation matrix $A \in \mathbb{R}^{k\times k}$, such that though the distributions have changed, i.e.

$$P(\mathbf{S}_{(t_1,t)}) \neq P(\mathbf{S}_{(t+1,t_2)}).$$

the transformed data stream $\bar{\mathbf{S}}_{(t_1,t_2)} = (\bar{\mathbf{x}}_{(t_1,t_2)}, y_{(t_1,t_2)})$, where $\bar{\mathbf{x}}_{(t_1,t_2)} = \{A\mathbf{x}_{t_1}, A\mathbf{x}_{t_1+1}, \ldots, A\mathbf{x}_{t_2}\}$ has the property that,

$$P(\bar{\mathbf{S}}_{(t_1,t)}) \approx P(\bar{\mathbf{S}}_{(t+1,t_2)})$$

That is,

$$\min_A \quad d_{MMS}(P(\bar{\mathbf{S}}_{(t_1,t)}), P(\bar{\mathbf{S}}_{(t+1,t_2)})) \tag{3.1}$$

Solving the optimization problem in Equation (3.1) directly is not trivial. According to the definition of the conditional probability, $P(\mathbf{S}_{(t_1,t_2)}) = P(\mathbf{x}_{(t_1,t_2)})P(y|\mathbf{x}_{(t_1,t_2)})$. We separate the marginal distribution discrepancy and conditional distribution discrepancy. The optimization problem in Equation (3.1) is modified as follows,

$$\begin{aligned}\min_A \quad & d_{MMD}(P(\bar{\mathbf{x}}_{(t_1,t)}), P(\bar{\mathbf{x}}_{(t+1,t_2)})) \\ & + d_{MMD}(P(y|\bar{\mathbf{x}}_{(t_1,t)}), P(y|\bar{\mathbf{x}}_{(t+1,t_2)})).\end{aligned} \tag{3.2}$$

3.3.1 Marginal Distribution Discrepancy

Denote $\mathbf{X} = [\mathbf{x_{t_1}}, \mathbf{x_{t_1+1}}, \ldots, \mathbf{x_{t_2}}]$, We calculate the marginal distribution discrepancy firstly,

$$\begin{aligned} & d_{MMD}(P(\bar{\mathbf{x}}_{(t_1,t)}), P(\bar{\mathbf{x}}_{(t+1,t_2)})) \\ &= ||\frac{1}{n_1}\sum_{i=t_1}^{t} A\mathbf{x}_i - \frac{1}{n_2}\sum_{j=t+1}^{t_2} A\mathbf{x}_j||^2 \\ &= tr\left(AXM_0X^TA^T\right) \end{aligned} \tag{3.3}$$

where $n_1 = t - t_1 + 1$, $n_2 = t_2 - t$, M_0 can be calculated as follows

$$(M_0)_{ij} = \begin{cases} \frac{1}{n_1^2} & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \mathbf{x}_{(t_1,t)}, \\ \frac{1}{n_2^2} & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \mathbf{x}_{(t+1,t_2)}, \\ -\frac{1}{n_1 n_2} & \text{otherwise.} \end{cases} \tag{3.4}$$

3.3.2 Conditional Distribution Discrepancy

Note that in classification problems, all the labels is discrete. The empirical estimate of MMD calculates the discrepancy of average kernel embedding of each instance. However, the averaging discrete label does not make sense. In JDA, $P(\bar{\mathbf{x}}_{(t_1,t_2)}|y)$ is used to replace $P(y|\bar{\mathbf{x}}_{(t_1,t_2)})$ in Equation (3.2). We calculate conditional distribution $P(\bar{\mathbf{x}}_{(t_1,t_2)}|y = c)$ for each label in $\{1, 2, \ldots, C\}$ and then add all the class-conditional distribution discrepancies up as the total conditional distribution discrepancy,

$$\begin{aligned} & d_{MMD}(P(\bar{\mathbf{x}}_{(t_1,t)}|y), P(\bar{\mathbf{x}}_{(t,t_2)}|y)) \\ &= \sum_{c=1}^{C} d_{MMD}(P(\bar{\mathbf{x}}_{(t_1,t)}|y = c), P(\bar{\mathbf{x}}_{(t+1,t_2)}|y = c)) \\ &= \sum_{c=1}^{C} ||\frac{1}{n_{1,c}}\sum_{\mathbf{x}_i \in S_{1,c}} A\mathbf{x}_i - \frac{1}{n_{2,c}}\sum_{\mathbf{x}_j \in S_{2,c}} A\mathbf{x}_j||^2 \\ &= \sum_{c=1}^{C} tr\left(AXM_cX^TA^T\right) \\ &= tr\left(AX\sum_{c=1}^{C} M_cX^TA^T\right), \end{aligned} \tag{3.5}$$

where $S_{1,c} = \{\mathbf{x}_i : y_i = c \text{ and } t_1 \leq i \leq t\}$, $S_{2,c} = \{\mathbf{x}_j : y_j = c \text{ and } t+1 \leq j \leq t_2\}$, and $n_{1,c} = |S_{1,c}|, n_{2,c} = |S_{2,c}|$ are the cardinals of $S_{1,c}, S_{2,c}$ respectively. The elements in MMD matrix M_c for $c \in \{1, 2, \ldots, C\}$ can be computed as follows

$$(M_c)_{ij} = \begin{cases} \frac{1}{n_{1,c}^2} & \text{if } \mathbf{x}_i, \mathbf{x}_j \in S_{1,c}, \\ \frac{1}{n_{2,c}^2} & \text{if } \mathbf{x}_i, \mathbf{x}_j \in S_{2,c}, \\ -\frac{1}{n_{1,c}n_{2,c}} & \begin{cases} if\ \mathbf{x}_i \in S_{1,c}, \mathbf{x}_j \in S_{2,c} \\ or\ \mathbf{x}_i \in S_{2,c}, \mathbf{x}_j \in S_{1,c}, \end{cases} \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

Incorporating Equations (3.3) and (3.5), we have the total distribution discrepancy

$$tr\left(AX\sum_{c=0}^{C} M_c X^T A^T\right)$$

and the matrix form of Equation (3.2) is,

$$\min_A \quad tr\left(AX\sum_{c=0}^{C} M_c X^T A^T\right) + \lambda||A||_{\mathcal{F}}^2 \tag{3.7}$$

Note that a regularization term $||A||_{\mathcal{F}}^2$ is added in Equation (3.7), where λ controls the impact of the regularization term.

3.3.3 Preserve Data Variance

However, only minimizing the distribution discrepancy is not enough. SEA Moving Hyperplane Concepts (SEA) Street and Kim (2001) is a widely used synthetic dataset in concept drift area. Instances in SEA have three features x_1, x_2 and x_3. The label is determined by an inequality

$$ax_1 + bx_2 \leq \theta,$$

where a, b, θ are parameters. The third feature x_3 is a noisy feature and obeys uniform distribution. When concept drift occurs, i.e. some parameters changed, to reduce the discrepancy of distributions, the linear transformation matrix might give

the third feature higher weight. An extreme case is that if

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

we have

$$P(\bar{\mathbf{D}}_{(t_1,t)}) = P(\bar{\mathbf{D}}_{(t,t_2)}).$$

Instances before and after drift have the same distribution after transformed, but we lose all the information.

Hence, while minimizing the distribution discrepancy, the properties of data that contain the classification information have to been preserved. Both TCA and JDA adopt the idea of Principal Component Analysis (PCA) to maximize the data variance. Let the centering matrix $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}$, where $\mathbf{I}$ is the identity matrix, $\mathbf{1}$ is matrix of ones, and n is the quantity of samples. Now we have the covariance matrix $AXHX^TA^T$. The aim of PCA is to maximize the variance,

$$\max_{A} \quad tr\left(AXHX^TA^T\right) \tag{3.8}$$

We adopt the same optimization aim, to minimize the Equation (3.7) while preserve Equation (3.8) as much as possible. Generalized Rayleigh quotient theory shows that minimizing Equation (3.7) while maximizing Equation (3.8) is equivalent to the problem that minimizes Equation (3.7) with Equation (3.8) fixed. Now we have the final form of the optimization problem,

$$\begin{aligned} \min_{A} \quad & tr\left(AX\textstyle\sum_{c=0}^{C} M_c X^TA^T\right) + \lambda||A||_{\mathcal{F}}^2 \\ & s.t. \quad AXHX^TA^T = I \end{aligned} \tag{3.9}$$

3.3.4 Learning Algorithm

In this section, we state the procedure of DAJDA. Denote $\Phi \in \mathbb{R}^{k\times k}$ as the Lagrange multiplier matrix, where

$$\Phi = \begin{bmatrix} \phi_1 & 0 & \cdots & 0 \\ 0 & \phi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \phi_k \end{bmatrix}.$$

Then we have the Lagrange function

$$\begin{aligned}\mathcal{L}(A,\Phi) = tr\left(A\left(X\sum_{c=0}^{C} M_c X^T + \lambda I\right)A^T\right) \\ + tr\left(\left(I - AXHX^TA^T\right)\Phi\right).\end{aligned}$$

The optimization problem with an equality constraint in Equation (3.9) is converted into an unconstrained optimization problem

$$\begin{aligned}\min_{A,\Phi} \quad & tr\left(A\left(X\sum_{c=0}^{C} M_c X^T + \lambda I\right)A^T\right) \\ & + tr\left(\left(I - AXHX^TA^T\right)\Phi\right).\end{aligned} \tag{3.10}$$

To find the local minimum of $\mathcal{L}$, let $\frac{\partial \mathcal{L}}{\partial A} = 0$, and we have

$$A\left(X\sum_{c=0}^{C} M_c X^T + \lambda I\right) = \Phi AXHX^T. \tag{3.11}$$

Equation (3.11) shows that the proper linear transformation matrix A consists of the left generalized eigenvectors of $(X\sum_{c=0}^{C} M_c X^T, XHX^T)$. Thus, we can find the matrix A via applying generalized eigendecomposition to Equation (3.11).

To handle concept drift adaptation problem in data stream, we adopt a fixed-size window technique. Two fixed-size windows are maintained in our algorithm. A window, denoted as W_1 holds historical instances and another window, denoted as W_2 holds newly arrived instances. When new instance comes, the instance is added

into W_2 until W_2 is full. Then a linear transformation matrix A is learned by solving the generalized eigendecomposition problem in Equation (3.11). Both historical instance and new instances are transformed into a latent feature space. The labels of newly coming instances can be predicted via the knowledge learned from historical instance. The procedure of the DAJDA is summarized in Algorithm 1.

Algorithm 1 Drift Adaptation via Joint Distribution Alignment

Input: Training dataset $D_{(0,t)}$, Data stream $\mathbf{X}_{(t+1,\infty)}$, window size m, regularization parameter λ

Output: Labels $y_{(t+1,\infty)}$ of data stream $\mathbf{X}_{(t+1,\infty)}$

1: Train a base learner based on $D_{(0,t)}$
2: Predict the label $\hat{y}_i$ of the new arrived instance $\mathbf{x}_i$
3: Initial window $W_1 = D_{(0,t)}$, add the newly arrived instance $\mathbf{x}_i$ in window W_2 after the true label y_i is obtained
4: **while** the size of W_2 reach m **do**
5: Initial X as the the combine of instances in windows W_1 and W_2
6: Initial the MMD matrix M_0 and $\{M_c\}_{c=1}^{C}$ by Equations (3.4) and (3.6)
7: Solve Equation (3.11) to construct the transformation matrix A
8: Train a new model based on the transformed data $Z = AX$
9: Let $W_1 = W_2$ and W_2 be empty
10: Transform the newly coming instance $\mathbf{z}_j = A\mathbf{x}_j$ and predict the label $\hat{y}_j$by the new model
11: Add the newly arrived instance $\mathbf{x}_j$ in window W_2 after the true label y_j is obtained
12: **end while**

3.4 Experimental Study

We conducted several experimental studies to evaluate our method. We firstly verified the effectiveness of DAJDA. Then the performance is evaluated on some widely used real-world datasets.

3.4.1 Verification of the Effectiveness

To verify the effectiveness of DAJDA, we generate some synthetic data stream, and two strategies are conducted as the baseline:

- **Baseline 1**: The model trained from training set is used all the time.
- **Baseline 2**: Retrain the model in every time window.

To investigate the effectiveness of DAJDA, three types of synthetic data streams are gendered. Several types of synthetic data streams have been used in the previous research. Data streams used in this chapter is described as follows.

SEA

SEA was introduced firstly in Street and Kim (2001). Each instance has three features, $x_1, x_2,$ and x_3, all of which obey Uniform distribution from 0 to 10. The label is determined by

$$y = \begin{cases} 1 & \text{if } ax_1 + bx_2 \leq \theta, \\ 0 & \text{otherwise,} \end{cases}$$

where a, b, θ are parameters.

ROT

ROT is introduced in Brzezinski and Stefanowski (2014) to simulate the concept drift in which the decision boundary is rotated. Each instance has two features,

x_1 and x_2. ROT rotates the instance to simulate the decision boundary rotation. Each instance is rotated a certain angle

$$x = (x - a)\cos\theta - (y - b)\sin\theta + a,$$

$$y = (x - a)\cos\theta + (y - b)\sin\theta + b,$$

where a, b, θ are parameters.

CIR

CIR is introduced in Gama et al. (2004) to generate the concept drift in which the decision boundary is a sphere in the feature space. Considering the 2-dimensional case, each instance has two features, x_1 and x_2. The label is given by

$$y = \begin{cases} 1 & \text{if } (x_1 - a)^2 + (x_2 - b)^2 \leq \theta, \\ 0 & \text{otherwise,} \end{cases}$$

where a, b, θ are parameters.

In our simulation, some parameters are fixed and others are changed to simulate the concept drift. For SEA, we fixed $a = b = 1$ and changed θ every 1000 instances to simulate the movement of the decision boundary. For ROT, we fixed $a = b = 0$ and changed θ every 1000 instances to simulate the rotation of the decision boundary. For CIR, we fixed the radius of the ball θ and changed the center of the ball (a, b) every 1000 instances. The statistics of generated synthetic data stream are shown in Table 3.1. The value of parameters which have been changed to simulate concept drift are shown in Table 3.2.

We set the window size $m = 20$ and the regularization parameter $\lambda = 1$. And naive Bayes classifier is chosen as the base model. The experiment results are listed in Table 3.3. The results show that DAJDA performed steadily and obtained highest score for all metrics on all data stream except the precision score on data stream

Table 3.1 : Synthetic Data Streams to Evaluate DAJDA

Data Stream	#Example	#Feature	#Label
SEA1	10000	3	2
SEA2	10000	3	2
ROT1	10000	2	2
ROT2	10000	2	2
CIR1	10000	2	2
CIR2	10000	2	2

Table 3.2 : Parameters of the Synthetic Data Streams to Evaluate DAJDA

Stream	Value of Drift Parameters
SEA1	$10 \rightarrow 5 \rightarrow 13 \rightarrow 5 \rightarrow 11 \rightarrow 12 \rightarrow 14 \rightarrow 14 \rightarrow 8 \rightarrow 7$
SEA2	$10 \rightarrow 12 \rightarrow 9 \rightarrow 14 \rightarrow 14 \rightarrow 14 \rightarrow 12 \rightarrow 11 \rightarrow 9 \rightarrow 9$
ROT1	$0.78 \rightarrow 1.06 \rightarrow 1.38 \rightarrow 1.49 \rightarrow 1.80 \rightarrow 1.93 \rightarrow 2.14$ $\rightarrow 2.13 \rightarrow 2.41 \rightarrow 2.32$
ROT2	$0.78 \rightarrow 1.04 \rightarrow 1.22 \rightarrow 1.26 \rightarrow 1.18 \rightarrow 1.41 \rightarrow 1.42$ $\rightarrow 1.71 \rightarrow 1.78 \rightarrow 1.87$
CIR1	$(0,0) \rightarrow (-1.03, 0.01) \rightarrow (1.18, 0.49) \rightarrow (1.81, -2.08)$ $\rightarrow (1.85, -4.21) \rightarrow (1.47, -3.78) \rightarrow (1.47, -2.20)$ $\rightarrow (0.64, -3.08) \rightarrow (3.38, -1.14) \rightarrow (3.92, -2.01)$
CIR2	$(0,0) \rightarrow (-0.02, 0.63) \rightarrow (-0.17, 1.75) \rightarrow (2.12, 2.93)$ $\rightarrow (3.36, 2.09) \rightarrow (3.26, 1.83) \rightarrow (3.64, 3.02)$ $\rightarrow (3.76, 2.80) \rightarrow (5.13, 1.90) \rightarrow (6.10, 1.51)$

SEA2 and CIR2. It can be concluded that DAJDA has abilities to improve the performance of learning model under concept drift.

3.4.2 Evaluation on Real-world Datasets

In this section, we evaluated our proposed method in some real-world datasets. Three real-world datasets are included. The statistics of three datasets are shown in Table 3.4.

Several state-of-the-art concept drift methods are compared with DAJDA in the experiment: Drift Detection Method (DDM) Gama et al. (2004), EWMA for Concept Drift Detection (ECDD) Ross et al. (2012), Hoeffding's inequality based Drift Detection Method (HDDM) Frías-Blanco et al. (2015) and Heoffding Adaptive Tree (HAT) Bifet and Gavaldà (2009). All the algorithms are implemented in MOA framework Bifet et al. (2010b). For the sake of fairness, tree models are adopted as base learner for DDM, ECDD, HDDM and DAJDA. According to the experimental results, the window size m in DAJDA is set as 1000 and the regularization parameter λ is set as 1. The experiment results are listed in Table 3.6 and the average rank of performance on all datasets is listed in Table 3.5. Experiment results show that DAJDA has advantages over other methods on datasets Covertype Blackard (1998) and Weather Elwell and Polikar (2011). HAT obtained the highest scores on dataset Electricity Harries (1999). However, Table 3.5 shows that DAJDA has the best performance considering all three datasets and all metrics. We can conclude that DAJDA improve the precision significantly.

Table 3.3 : Experiment Results of Verification of the Effectiveness of DAJDA

Metrics	Stream	baseline 1	baseline 2	DAJDA
Accuracy	SEA1	0.7356	0.8739	**0.9079**
	SEA2	0.8133	0.8518	**0.9074**
	ROT1	0.5643	0.8800	**0.9232**
	ROT2	0.6674	0.8997	**0.9222**
	CIR1	0.9078	0.9310	**0.9431**
	CIR2	0.8990	0.9372	**0.9466**
F1 score	SEA1	0.7313	0.8742	**0.9082**
	SEA2	0.7852	0.7945	**0.8746**
	ROT1	0.5680	0.8806	**0.9236**
	ROT2	0.6713	0.9003	**0.9225**
	CIR1	0.9512	0.9632	**0.9699**
	CIR2	0.9462	0.9672	**0.9719**
Precision	SEA1	0.7187	0.8689	**0.9037**
	SEA2	**0.8974**	0.7539	0.8495
	ROT1	0.5688	0.8791	**0.9221**
	ROT2	0.6774	0.9038	**0.9233**
	CIR1	0.9767	0.9966	**0.9971**
	CIR2	0.9598	**0.9983**	0.9969
Recall	SEA1	0.7443	0.8796	**0.9128**
	SEA2	0.6979	0.8398	**0.9014**
	ROT1	0.5672	0.8822	**0.9252**
	ROT2	0.6652	0.8968	**0.9217**
	CIR1	0.9269	0.9329	**0.9442**
	CIR2	0.9331	0.9379	**0.9481**

Table 3.4 : Real-world Data Streams to Evaluate DAJDA

Data Stream	#Example	#Feature	#Label
Covertype	581012	54	7
Electricity	45312	8	2
Weather	18159	8	2

Table 3.5 : Average Rank of DAJDA on All Three Datasets

Method	Accuracy	F1 score	Precision	Recall
DAJDA	**1.67**	2	**2**	**1.33**
DDM	2.67	3	2.67	4
ECDD	6	6	6	6
HDDM-A	4.67	4.33	3.33	5
HDDM-W	3.67	4	4	5
HAT	2	**1.67**	**2**	2.67

Table 3.6 : Experiment Results of Evaluation of DAJDA on Real-world Dataset

Dataset	Method	Accuracy	F1 score	Precision	Recall
Covertype	DAJDA	**0.8457**	0.6286	**0.6424**	**0.6181**
	DDM	0.5974	0.4473	0.5106	0.3983
	ECDD	0.5321	0.3811	0.4305	0.3427
	HDDM-A	0.6068	0.4574	0.5290	0.4030
	HDDM-W	0.6403	0.4580	0.5251	0.4062
	HAT	0.7887	**0.6381**	0.6341	0.6040
Electricity	DAJDA	0.6860	0.6790	0.6786	0.6793
	DDM	0.6727	0.6837	0.6822	0.6082
	ECDD	0.4784	0.5105	0.5069	0.5014
	HDDM-A	0.6228	0.6401	0.6407	0.6036
	HDDM-W	0.6600	0.6624	0.6684	0.6051
	HAT	**0.7557**	**0.7502**	**0.7484**	**0.7051**
Weather	DAJDA	0.7255	**0.6826**	0.6816	**0.6836**
	DDM	**0.7435**	0.6710	**0.6900**	0.6054
	ECDD	0.6434	0.5936	0.5908	0.5094
	HDDM-A	0.6846	0.6540	0.6448	0.6064
	HDDM-W	0.7033	0.6431	0.6445	0.6049
	HAT	0.7160	0.6665	0.6654	0.6068

3.5 Summary

In this chapter, a concept drift adaptation method, Drift Adaptation via Joint Distribution Alignment (DAJDA), is proposed. The method transforms the data into common feature space. The Maximum Mean Discrepancy (MMD) is used to measure the distance of two groups of data on both marginal distribution discrepancy and conditional distribution discrepancy. The transform should satisfy minimum MMD of two groups of data after transformation. To preserve the classification information, the data variance is retained. The optimization problem is solved by generalized eigendecomposition. In addition, the proposed method is model-free. Experimental studies show that DAJDA has the ability to improve the performance of learning model in evolving environment.

Chapter 4

A Multi-stream Concept Drift Handling Framework via Data Sharing

4.1 Introduction

Concept drift, a phenomenon where data distribution changes over time, is one of the key challenges in data stream mining research Lu et al. (2019). When concept drift occurs, the model trained with previous data cannot make accurate prediction for the current case. For example, in a recommender system, a car advertisement could be a good recommendation when the user wants to buy a car, but a bad one after the user has bought a car. Formally, concept drift is defined as $\exists t_1, t_2$, such that $P(X_{t_1}, y_{t_1}) \neq P(X_{t_2}, y_{t_2})$. An *offline* learning model cannot obtain steady accuracy when applied to predict a data stream with concept drift Gama et al. (2014).

Currently, most methods handling concept drift follow a fixed paradigm as shown in Figure 4.1. Once concept drift is detected, the learning model will be updated. Many concept drift adaptation methods are developed in the previous literature Gama et al. (2014); Lu et al. (2019). Most methods proposed focus on updating part of the model to adapt to concept drift quickly, rather than updating the whole model. Therefore, these methods have advantages in that they can save computation resources. But these methods are all limited to being applied to one specific type of learning model, such as decision tree models Bifet and Gavaldà (2009), kNN-based models Losing et al. (2016) and gradient descend-based methods Ng et al. (2017). It is difficult for these methods to be migrated to other learning models. The most general strategy to handle concept drift is *re-training*, which is a

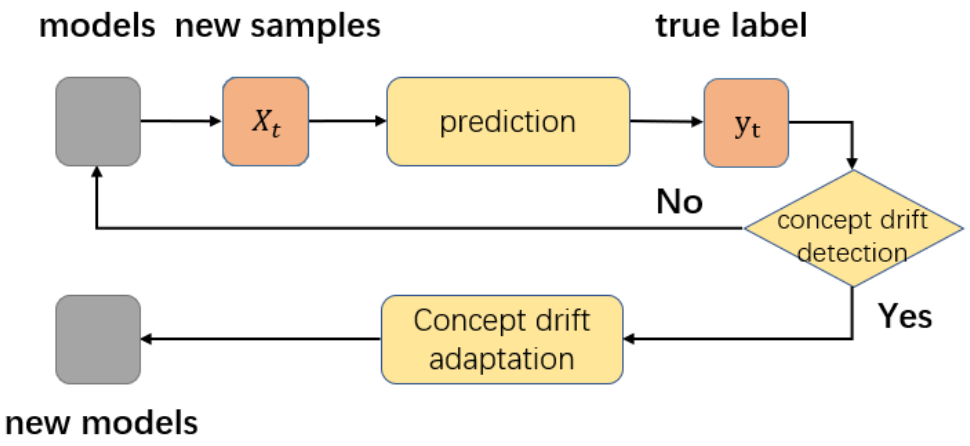


Figure 4.1 : The paradigm of handling concept drift for single data stream. After new data arrives, the learning model gives its prediction. The true label is then available. A concept drift detection procedure is conducted to detect whether concept drift occurs. If yes, a concept drift adaptation method is used to update the old learning model.

.

default option in lots of concept drift detection research Frías-Blanco et al. (2015); Liu et al. (2021a). That is to train a new learning model after concept drift is detected, which can be applied to all learning models.

A drawback of the re-training schema in Figure 4.1 is that it suffers from the over-fitting problem. Namely, a learned model will be over-fitted on the local pattern due to a limited amount of new data. This is because when concept drift is detected, data before drift should not be used for learning models intuitively, considering the data distribution has changed. In the recommender system example, recommendation of the same product to the same user can change to a complete opposite before and after concept drift occurs. If only using the newly arrived data after drift which is of a limited amount to re-train the model, the new model usually has pool performance. According to the PAC learnable theory Valiant (1984), the generalization error has

a negative correlation to the sample size. Usually, there exists bias between the distributions of samples and population. In addition, the noise of samples cannot be ignored when the size of sample is small. When the learned model is over-fitted on a local pattern, the performance of the learned model deteriorates when predicting labels due to the bias and noise.

To fix the over-fitting issue, using data from other data streams can be a solution. So far, the design and testing of most concept drift methods are conducted in a *single stream* scenario Song et al. (2021). However, practical applications often handle *massive* and *correlated* data streams. For example, Harris et al. Harries (1999) recorded the electricity price in both NSW and VIC in Australia, which has become a widely used dataset in concept drift research. The price trend is shown in Figure 4.2, where there is a significant correlation between the changing in prices. When concept drift is detected in one data stream, similar concept drift might have occurred in another correlated data stream a while ago, which means that data from these correlated data streams can reflect the new concept of the data after concept drift. But few research works leverage the information of the correlation between data streams to address the concept drift problem.

To alleviate the over-fitting problem when re-training the models, a stream fuzzy set is defined, and based on this stream fuzzy set, we develop a Multi-stream Concept Drift Handling Framework in this work. The developed framework contains a fuzzy membership-based Drift Detection (FMDD) component and a fuzzy membership-based Drift Adaptation (FMDA) component. Rather than handling data streams separately, the method is designed to also consider the correlation between streams in the multi-stream scenario. For each newly arrived batch of data, we first compute the fuzzy membership in the stream fuzzy set to present the correlation between streams. After that, FMDD uses the computed fuzzy membership to determine whether one data stream is a drifting stream. After separating streams into non-drifting and

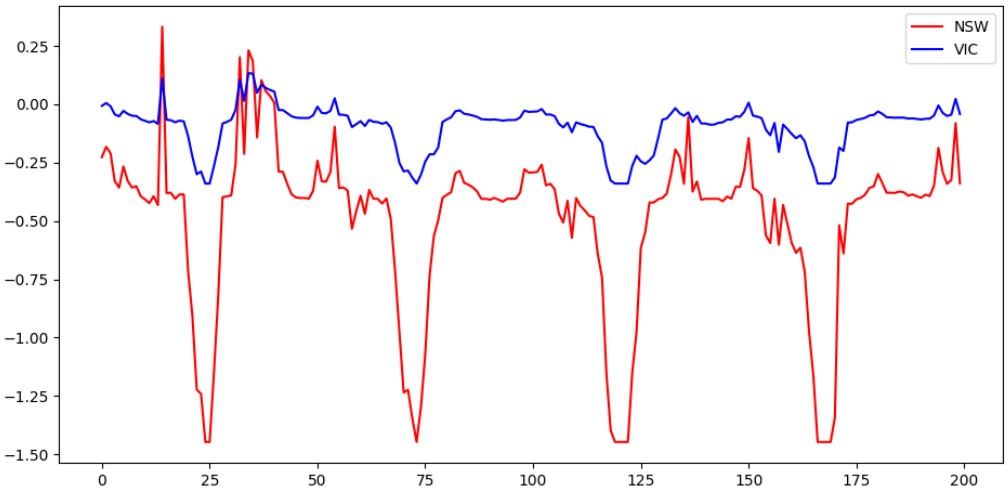


Figure 4.2 : Electricity price trend of NSW and VIC. The change in price in the two states shows a significant correlation. Negative data exist because the data set has been normalized

.

drifting streams, FMDA re-trains models for drifting streams by adding training samples from non-drifting streams. While, the models for non-drifting streams stay the same.

The defined *stream fuzzy set* measures how a sample belongs to a data stream. A large membership of a sample from stream A belonging to stream B means that these two data streams are highly correlated. FMDD identifies concept drift by comparing the membership of historical data with the membership of newly arrived data. If the membership of a newly arrived sample to the stream that this sample is from decreases significantly, meaning that this sample does not belong to this stream anymore, concept drift occurs in this stream. This is because if drift does not occurs, i.e., the data distribution does not change, the newly arrived sample should still belong to the stream that this sample is from. A Mann-Whitey U Rank Test Mann and Whitney (1947) is used in FMDD to test whether the membership

is significantly decreased. Given the non-drifting and drifting streams, FMDA enlarges the training data to re-train models for the drifting streams by using samples from both drifting and non-drifting streams. The non-drifting streams samples are weighted by the fuzzy membership. A larger membership means a larger weight. Though the distributions of these data are not completely the same, the new learning model can benefit from the proper weights.

From this, our contributions are summarized as follows:

- A novel multi-stream Concept drift Handling Framework is proposed which considers the correlation between multiple data streams rather than handling data streams separately. The advantage of the framework is that, the parameters of the membership functions are estimated using data from other streams. These data which have different distribution help to reach higher concept drift detection accuracy.

- A new drift detection method, FMDD is designed to detect when and in which streams concept drift occurs, dividing streams into drifting and non-drifting streams at each time. The parameters of fuzzy membership functions are estimated using data from other data streams, which lead to a remarkable high true pasitive rate.

- A new drift adaptation method, FMDA is proposed, using the correlation of multiple data streams to train a new model after concept drift is detected. By increasing the volume of training data, the over-fitting issue due to lack of data is alleviated.

- Experimental studies are conducted on both synthetic and real-world data sets, which show that when the batch size is small, i.e., the learners suffer from the over-fitting issue, our framework can successfully alleviate the issue and significantly improve prediction accuracy.

The rest of this chapter is organized as follows. In Section 4.2, we present an overview of related work. Section 4.3 demonstrates the details of our proposed method. In Section 4.4, we conduct an experimental study to evaluate our method. Our conclusions are presented in Section 4.5.

4.2 Proposed Method

In this section, we provide the details of our method. We give the problem setting in Section 4.2.1. The basic assumptions of this chapter are listed in Section 4.2.2. In Section 4.2.3, we define the stream fuzzy set. Then FMDD and FMDA are proposed in Section 4.2.4 and Section 4.2.5 respectively.

4.2.1 Problem Settings

We preface our discussion of concept drift, with a formal definition of data streams.

Definition 1 (Data Stream)**.** *A data stream* $S = \{(X_1, y_1), (X_2, y_2), \ldots, (X_t, y_t), \ldots\}$ *is a real-time, dynamic and infinite data sequence, where* (X_t, y_t) *is an instance at time* t*,* $X_t \in \mathcal{X}$ *is the feature and* $y_t \in \mathcal{Y}$ *is the target variable to be predicted.*

Then we have the first problem.

Problem 1. *Given* m *data streams* $\mathcal{S} = \{S_i\}_{i=1}^{m}$*, where* $S_i = \{(X_1^{(i)}, y_1^{(i)}), (X_2^{(i)}, y_2^{(i)}), \ldots, (X_t^{(i)}, y_t^{(i)}), \ldots\}$*. How do we recognize the correlation between data streams* $\mathcal{S}$ *automatically?*

Concept drift refers to the phenomenon where the distribution of data changes in the data streams over time Gama et al. (2014). Next, we re-state the formal definition of concept drift.

Definition 2 (Concept Drift)**.** *Concept drift occurs if* $\exists t_1, t_2$*, such that the distributions*

$$P(X_{t_1}, y_{t_1}) \neq P(X_{t_2}, y_{t_2}).$$

We mark the distribution of data before the concept drift as the old distribution, and the distribution of data after the concept drift as the new distribution. Now we have the second problem.

Problem 2. *Given* m *data streams* $\mathcal{S} = \{S_i\}_{i=1}^{m}$*, where* $S_i = \{(X_1^{(i)}, y_1^{(i)}), (X_2^{(i)}, y_2^{(i)}), \ldots, (X_t^{(i)}, y_t^{(i)}), \ldots\}$*, a concept drift is detected at time* t *on* S_i*. Only few data* $\{(X_s^{(i)}, y_s^{(i)}), \ldots, (X_t^{(i)}, y_t^{(i)})\}$ *are marked as data from the new distribution. Lack of data can lead to an over-fitting issue when re-training a new learning model. Based on the correlation recognized in Problem 1, how is it possible to solve the over-fitting issue using data from correlated data streams?*

4.2.2 Basic Assumptions

In this work, we make two assumptions:

- We assume that the data arrive batch after batch. If the data arrive one by one, we wait until the data constitute a batch. Samples of the ith stream $\{(X_{t_{j-1}+1}^{(i)}, y_{t_{j-1}+1}^{(i)}), \ldots, (X_{t_j}^{(i)}, y_{t_j}^{(i)})\}$ arrive at the same time. The size of the batch $t_j - t_{j-1}$ remains the same in the stream.

- We assume that the true label is available soon after predictions are made, such that the drift detection method can identify whether concept drift occurs.

4.2.3 Stream Fuzzy Set

In this section, we define the stream fuzzy sets. To measure the degree of any data sample belonging to a data stream, an intuitive method is to use distance, such as MMD Gretton et al. (2006), which measures the distance between distributions

of two groups of samples. However, the drawback of these methods is the high computation complexity, which is not suitable for a data stream mining scenario. Therefore, in this work, to measure the degree of how sample $X_t^{(i)}$ belongs to stream S_j, the membership is calculated based on the performance of the learning model, thereby simplifying the computation. If it produces a good prediction when using the learning model of S_j to predict $X_t^{(i)}$, the degree of that $X_t^{(i)}$ belongs to stream S_j should be large.

For Stream S_i, and a new batch of data of Stream $S_{i'}$, $\{X_{t_{j-1}+1}^{(i')}, \ldots, X_{t_j}^{(i')}\}$ arrive at time t_j. The learning model of Stream S_i makes its predictions $\{\hat{y}_{t_{j-1}+1}^{(i,i')}, \ldots, \hat{y}_{t_j}^{(i,i')}\}$. After the true labels $\{y_{t_{j-1}+1}^{(i')}, \ldots, y_{t_j}^{(i')}\}$ are available, the prediction error is obtained. The type of error depends on tasks, such as mean squared error for regression tasks,

$$\mathcal{L}_{t_j}^{(i,i')} = \frac{1}{t_j - t_{j-1}} \sum_{\tau=t_{j-1}}^{t_j} \left(y_\tau^{(i)} - \hat{y}_\tau^{(i,i')}\right)^2.$$

Then we define the stream fuzzy set.

Definition 3 (Stream Fuzzy Set)**.** *$M_i = \{\mathcal{X}, f_i\}$ is a stream fuzzy set of stream S_i, where $f_i : \mathcal{X} \to [0, 1]$ is its membership function. The membership function of a stream fuzzy set is defined as,*

$$f_i(X_t^{(i')}) = f_{\mathcal{M}}(\mathcal{L}_t^{(i,i')}; \theta_i),$$

where $f_{\mathcal{M}} : \mathbf{R} \to [0, 1]$ is monotonically decreasing function, and θ_i is its parameters.

For $X_t^{(i)}$ of Stream S_i, the prediction error $\mathcal{L}_t^{(i,i)}$ should be small if there is no concept drift occurring. Therefore, its membership calculated by $f_i(X_t^{(i)})$ is large. This is intuitive because $X_t^{(i)}$ is from Stream i. For $X_t^{(i')}$ of Stream $S_{i'}$, if the distribution of $S_{i'}$ is different from that of Stream S_i, the prediction error using learning model of Stream S_i to predict samples of Stream $S_{i'}$ should be large, which

leads to a small membership. Conversely, if the distribution of $S_{i'}$ is similar with that of Stream S_i, the membership calculated by $f_i(X_t^{(i')})$ should be large.

However, we do not know the correlation between data streams in advance. To estimate the parameters θ_i of f_i, we assume that the distribution of sample from other data streams is different. Therefore the empirical estimated membership function f_i should satisfy that, $f_i(X_t^{(j)})$ is closed to 1 if $i = j$, and $f_i(X_t^{(j)})$ is closed to 0 if $i \neq j$. Therefore, we have the membership function's parameter estimation algorithm. Let $\{X_{t_{j-1}+1}^{(i)}, \ldots, X_{t_j}^{(i)}\}$ be a batch of Stream S_i. Let $\{X_{t_{j-1}+1}^{(i')}, \ldots, X_{t_j}^{(i')}\}$ be a batch of data randomly sampled from other data streams. The parameters of f_i are solutions of the following optimization problem,

$$\theta_i = \arg\min_{\theta} \sum_{\tau=t_{j-1}}^{t_j} (f_i(X_\tau^i) - 1)^2 + \sum_{\tau=t_{j-1}}^{t_j} (f_i(X_\tau^{i'}) - 0)^2 \tag{4.1}$$

The parameter estimation algorithm using gradient descent is summarized in Algorithm 2.

4.2.4 Fuzzy Membership-based Drift Detection Method

In this section, we state our proposed method FMDD. Let $\{X_{t_{j-2}+1}^{(i)}, \ldots, X_{t_{j-1}}^{(i)}\}$ be a historical batch of stream S_i. Their membership is denoted as $C_{t_{j-1}}^{(i)}$. Let $\{X_{t_{j-1}+1}^{(i)}, \ldots, X_{t_j}^{(i)}\}$ be a newly arrived batch of streams S_i. Their membership is denoted as $C_{t_j}^{(i)}$. If $C_{t_j}^{(i)}$ is significantly smaller than $C_{t_{j-1}}^{(i)}$, it is alerted that concept drift occurs. This is because $C_{t_j}^{(i)}$ is smaller than $C_{t_{j-1}}^{(i)}$, means that $\mathcal{L}_{t_j}^{(i,i)}$ is larger than $\mathcal{L}_{t_{j-1}}^{(i,i)}$. The increasing prediction error implies the distribution of newly arrived samples has changed, and the learning model needs to be updated.

A Mann-Whitney U Rank Test Mann and Whitney (1947) is performed to detect whether there are significant differences between $C_{t_{j-1}}^{(i)}$ and $C_{t_j}^{(i)}$. The U statistic is defined as

$$U = \sum_{\tau_1=t_{j-2}+1}^{t_{j-1}} \sum_{\tau_2=t_{j-1}+1}^{t_j} \mu(C_{\tau_1}^{(i)}, C_{\tau_2}^{(i)}), \tag{4.2}$$

Algorithm 2 Parameter Estimation using Gradient descent

Input: batch of data $\{X^{(i)}_{t_{j-1}+1}, \ldots, X^{(i)}_{t_j}\}$ and $\{X^{(i')}_{t_{j-1}+1}, \ldots, X^{(i')}_{t_j}\}$

Parameter: learning rate α, iteration number T, terminating bound ϵ

Output: parameter θ_i

Initialize θ_i randomly.
for $t = 1$ to T **do**
  Let $J = \sum_{\tau=t_{j-1}}^{t_j} (f_i(X^i_\tau) - 1)^2 + (f_i(X^{i'}_\tau) - 0)^2$
  $G = \frac{\partial J}{\partial \theta_i}$
  if $G < \epsilon$ **then**
    Break
  end if
  Update $\theta_i = \theta - \alpha G$
end for
return θ_i

where

$$\mu(x, y) = \begin{cases} 1 & \text{if } x > y \\ 0.5 & \text{if } x = y \\ 0 & \text{if } x < y \end{cases}$$

The U statistic's distribution approximates to Normal distribution. Whether the null hypothesis is rejected or not depends on the p value.

The fuzzy membership-based Drift Detection Method is summarized in Algorithm 3.

4.2.5 Fuzzy Membership-based Drift Adaptation Method

In this section, we state our proposed method FMDA. Let $\{S_i\}_{i=1}^m$ be m data streams. New batches of data arrive at time t_j. Then these data streams are divided into two groups after the drift detection procedure, $\{S_{i_1}\}_{i_1 \in \mathcal{I}_D}$ and $\{S_{i_2}\}_{i_2 \in \mathcal{I}_N}$,

Algorithm 3 Fuzzy Membership-based Drift Detection Method

Input: membership of historical batch of data $C^{(i)}_{t_{j-1}}$, membership of newly arrived batch of data $C^{(i)}_{t_j}$

Parameter: significant level α

Output: drifting state

1: Calculate U by Equation (4.2).
2: Calculate the p value of U.
3: **if** $p < \frac{1}{2}\alpha$ or $p > 1 - \frac{1}{2}\alpha$ **then**
4: **return** 1
5: **else**
6: **return** 0
7: **end if**

where $\mathcal{I}_D$ is the index set of drifting streams and $\mathcal{I}_N$ is the index set of non-drifting streams. In traditional single-stream methods, data streams are handled separately. For Streams in $\{S_{i_2}\}_{i_2 \in \mathcal{I}_N}$, nothing happen. For Streams in $\{S_{i_1}\}_{i_1 \in \mathcal{I}_D}$, a new learner is trained. To re-train a new learner to adapt to the concept drift, only $\{(X^{(i)}_{t_{j-1}+1}, y^{(i)}_{t_{j-1}+1}), \ldots, (X^{(i)}_{t_j}, y^{(i)}_{t_j})\}$ are used. Usually, the batch size of $Z^{(i)}_j$ is small. Lack of data leads to the problem of over-fitting and a large generalization error.

To manage this issue, we hope to use additional data from $\{S_{i_2}\}_{i_2 \in \mathcal{I}_N}$ to train a new learning model. However, the data distribution of streams is different. Using data directly from other data streams, which have different distributions, can lead to the low performance of the new learning model. The intuitive idea is to assign weights to data from other data streams. If the data distribution is similar, the weights should be large. Conversely, if the data distribution is different, the weights should be small, or zero if the data distribution is irrelevant. Then the weight of data from S_{i_2} in re-training of S_{i_1} is the average membership that $Z^{(i_1)}_j$ belong to

Algorithm 4 Fuzzy Membership-based Drift Adaptation Method

Input: new batch of data

Output: weights

1: Drift detection for all data streams using FMDD.
2: Divide streams into two groups
$\{S_{i_1}\}_{i_1 \in \mathcal{I}_D}$ and $\{S_{i_2}\}_{i_2 \in \mathcal{I}_N}$.
3: **for** $S_{i_1} \in \{S_{i_1}\}_{i_1 \in \mathcal{I}_D}$ **do**
4: **for** $S_{i_2} \in \{S_{i_2}\}_{i_2 \in \mathcal{I}_N}$ **do**
5: Calculate weights by Equation (4.3).
6: **end for**
7: **end for**
8: **return** $w^{(i_1,i_2)}$

stream S_{i_2}.

$$w^{(i_1,i_2)} = \frac{1}{N} \sum_{\tau=t_{j-1}+1}^{t_j} f_{i_2}(X_\tau^{(i_1)}). \tag{4.3}$$

where $N = t_j - t_{j-1}$. The fuzzy membership-based Drift Adaptation is summarized in Algorithm 4. The weights are used to re-train a new learning model. For example, in some gradient descent-based methods, such as linear regression and neural networks, the weights are used as the amplification factor of learning rate in gradient descent.

The entire procedure of Multi-stream Concept Drift Handling Framework is illustrated in Figure 4.3.

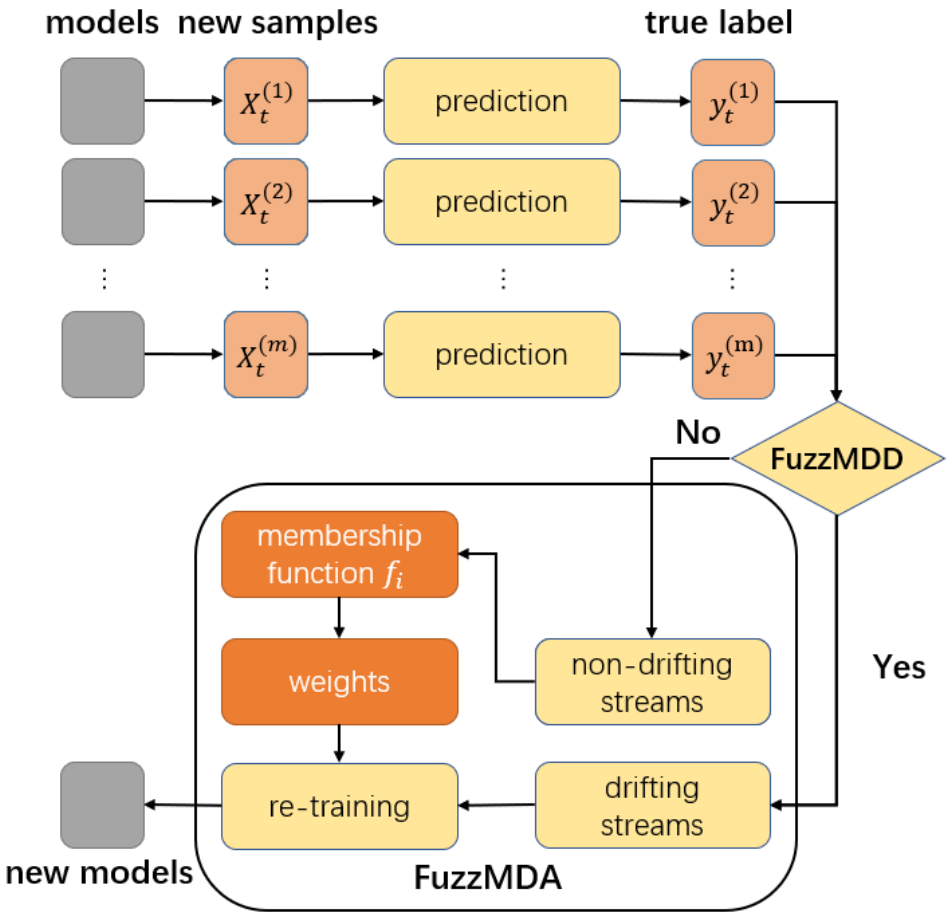


Figure 4.3 : Flow chart of Multi-stream Concept drift Handling Framework

.

4.3 Experimental Study

In this section, we conduct a series of experimental studies to evaluate our Multi-stream Concept Drift Handling Framework. We give a comprehensive evaluation from the following aspects.

1. We investigate the correlation between the fuzzy membership and distribution discrepancy. The results are shown in Figure 4.4.

2. Next, we evaluate the detection accuracy of our drift detection method FMDD.

The results are summarized in Table 4.2.

3. We compare the prediction accuracy of learners trained by adding weighted data from different data streams on twelve synthetic data sets. The results are shown in Figure 4.5.

4. We evaluate prediction accuracy of learners using FMDA on two real-world data sets. The results are summarized in Table 4.3 and Table 4.4.

5. Next, we conduct a batch size study to investigate the effect of the batch size. The results are summarized in Figure 4.6 and Figure 4.7.

6. Finally, we give a visualization explanation of stream correlation. The visualization is shown in Figure 4.8.

All experiments were conducted on a computing cluster with an INTEL Xeon Gold 6126 CPU @ 2.60GHz and 192G memory, and the operating system is Red Hat Enterprise Linux*.

4.3.1 Data Sets

Our method is evaluated on both synthetic and real-world data sets. A synthetic data set Synthetic00 is generated to evaluate FMDD. and twelve synthetic data sets Synthetic01-Synthetic12 are generated to evaluate FMDA. Then two real-world data sets are used in our experimental study.

Synthetic01

In Synthetic01, we generated 100 streams of samples. The feature $x \in \mathbf{R}^{10}$ is generated from a 10-dimension multivariate standard Gaussian distribution. The

*https://www.redhat.com

label $y \in \mathbf{R}$ is generated by a linear model,

$$y = \omega x + b + \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, 0.1)$ is noise. Each stream contains 1000 samples, which are split into two subgroups. The first 500 samples and the last 500 samples are generated from different distributions, which means that the concept drift occurs between the first and the last 500 samples. The differences are parameters ω and b. The parameters satisfy

$$\omega_2 - \omega_1 \sim \mathcal{N}(0, 1), \quad b_2 - b_1 \sim \mathcal{N}(0, 1).$$

Synthetic02

The generation method of Synthetic02 is similar to Synthetic01's. In Synthetic02, we generate ten synthetic data streams. Different from Synthetic01, to simulate different degree of concept drift, the parameters ω and β before and after concept drift satisfy

$$\omega_2 - \omega_1 \sim \mathcal{N}(0, \delta), \quad b_2 - b_1 \sim \mathcal{N}(0, \delta),$$

where δ represent the degree of concept drift, which varies from 0.1 of Stream 1 to 1 of Stream 10.

Synthetic03-14

The generation methods of Synthetic03 to Synthetic14 are the same. Each data set includes four data streams, one major stream and three auxiliary streams. Each stream has 1000 samples and the feature is 10-dimension. In the major stream, the feature $x_1 \in \mathbf{R}^{10}$ is generated from a 10-dimension multivariate standard Gaussian distribution. The label $y_1 \in \mathbf{R}$ is generated by a linear model,

$$y = \omega x + b + \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, 0.1)$ is noise. For the other three auxiliary streams, the generation methods are similar. The differences are parameters ω and b. The parameters satisfy

$$\omega_2 - \omega_1 \sim \mathcal{N}(0, 0.1), \quad b_2 - b_1 \sim \mathcal{N}(0, 0.1),$$

$$\omega_3 - \omega_1 \sim \mathcal{N}(0, 0.1), \quad b_3 - b_1 \sim \mathcal{N}(0, 0.1),$$

$$\omega_4 - \omega_1 \sim \mathcal{N}(0, 1), \quad b_4 - b_1 \sim \mathcal{N}(0, 1),$$

and $\exists 1 \leq i \leq d$, such that $\omega_3[i] - \omega_1[i] \sim \mathcal{N}(0, 3)$. The first auxiliary stream has approximate distribution with the major stream. The second is similar except for a feature with different distribution. The difference between the third and the major stream is significant.

Synthetic15

The generation method of Synthetic15 is similar to Synthetic03-14's. In Synthetic15, we generate five synthetic data streams. Different from Synthetic03-14, Stream 1-3 are generated from the same distribution, and Stream 4 and 5 are generated from the same distribution. The difference of the two distributions are the parameters ω and β, which satisfy

$$\omega_2 - \omega_1 \sim \mathcal{N}(0, 1), \quad b_2 - b_1 \sim \mathcal{N}(0, 1).$$

Real-world Data sets

The two real-world data sets are used in our experimental study.

Train The Train data set Yu et al. (2020) records train dwelling information of eight stations in New South Wales. There are eight data streams in the data set. Each data stream is corresponding to a train station. The task is to predict the load change of carriages during the train dwelling time, so that staff can organize the crowd and avoid congestion.

Table 4.1 : Summary of Datasets to Evaluate the Multi-stream Concept Drift Handling Framework

Dataset	#streams	#sample	#feature
Synthetic01	100	1000	10
Synthetic02	10	1000	10
Synthetic03-Synthetic14	4	1000	10
Synthetic15	5	1000	10
Train	8	373327	18
Weather	10	49728	8

Weather The Weather data set Wang et al. (2019) records weather data from ten weather stations near Beijing China. There are ten data streams in the data set. Each data stream is corresponding to a weather station. The weather data includes temperature, pressure, humidity, etc.. The task is to predict the accumulated rainfall in one hour.

The details of synthetic data sets and both real-world data sets are summarized in Table 4.1.

4.3.2 Evaluation of Stream Fuzzy Set

We use Synthetic00 to show how our proposed Stream Fuzzy Set can measure the degree to which samples belong to a data streams. For each stream, we use the first 500 samples to build the stream fuzzy set and estimate the parameters of the responding membership function. We calculate the fuzzy membership of the last 500 samples. Then the MMD of two group of samples are also calculated. The correlation between fuzzy membership and MMD are shown in Figure 4.4. Each point represent a stream in Synthetic01. The x-axis of points represent the reciprocal

of fuzzy membership. The y-axis of points represent the MMD. The orange dashed line is the linear least-squares regression estimated from the data.

It is clear that there exists a positive correlation between the reciprocal of our proposed fuzzy membership and MMD. However, the average computing time of fuzzy membership is 1.039e-4 seconds, while the average computing time of MMD is 2.419e-2 seconds.

Thus, there exists a correlation between the fuzzy membership and the distribution discrepancy. The fuzzy membership stand out due to its low computing time, which is suitable in the data stream mining scenario.

4.3.3 Evaluation of FMDD

In this section we evaluate the drift detection accuracy of our drift detection method FMDD on Synthetic02. FMDD determine whether concept drift occurs on streams in Synthetic02. The ground truth of concept drift occurring or not is available in Synthetic02. Then the detection accuracy is obtained based on the ground truth.

First a learner is built for each stream. All learners are linear models with an L2 regularization term on ω and b, implemented in Scikit-Learn[†] of version 0.24.2. The used data are randomly sampled from the former 500 samples. There are two ways to generate a testing set: sample from the former 500 samples, which means no concept drift occurs, or from the latter 500 samples, which implies a concept drift. Once FMDD determine the result, concept drift occurring or not, we can then know whether the result is correct or not. Two state-of-the-art drift detection methods, DDM Gama et al. (2004) and HDDM Frías-Blanco et al. (2015) are used as the baseline. There are two varieties of HDDM, HDDM-a and HDDM-w. The results

[†]https://scikit-learn.org

are listed in Table 4.2.

Table 4.2 : Results of evaluation of FMDD on Synthetic Streams

Metrics	Stream	FMDD	DDM	HDDM-a	HDDM-w
True positive	1	**0.644**	0.28	0.257	0.305
	2	**1.0**	0.25	0.276	0.28
	3	**1.0**	0.223	0.232	0.244
	4	**1.0**	0.194	0.204	0.215
	5	**1.0**	0.191	0.184	0.214
	6	**1.0**	0.184	0.175	0.21
	7	**1.0**	0.168	0.182	0.194
	8	**1.0**	0.179	0.165	0.195
	9	**1.0**	0.188	0.186	0.206
	10	**1.0**	0.174	0.165	0.191
True negative	1	0.832	0.794	**0.9**	0.828
	2	0.811	0.806	**0.91**	0.825
	3	0.813	0.814	**0.889**	0.829
	4	0.801	0.792	**0.893**	0.825
	5	0.796	0.81	**0.912**	0.863
	6	0.804	0.796	**0.893**	0.821
	7	0.788	0.805	**0.896**	0.814
	8	0.815	0.795	**0.905**	0.821
	9	0.795	0.798	**0.88**	0.818
	10	0.781	0.787	**0.897**	0.836
False positive	1	0.168	0.206	**0.1**	0.172

Continued on next page

Continued from previous page

Metrics	Stream	FMDD	DDM	HDDM-a	HDDM-w
False positive	2	0.189	0.194	**0.09**	0.175
	3	0.187	0.186	**0.111**	0.171
	4	0.199	0.208	**0.107**	0.175
	5	0.204	0.19	**0.088**	0.137
	6	0.196	0.204	**0.107**	0.179
	7	0.212	0.195	**0.104**	0.186
	8	0.185	0.205	**0.095**	0.179
	9	0.205	0.202	**0.12**	0.182
	10	0.219	0.213	**0.103**	0.164
False negative	1	**0.356**	0.72	0.743	0.695
	2	**0.0**	0.75	0.724	0.72
	3	**0.0**	0.777	0.768	0.756
	4	**0.0**	0.806	0.796	0.785
	5	**0.0**	0.809	0.816	0.786
	6	**0.0**	0.816	0.825	0.79
	7	**0.0**	0.832	0.818	0.806
	8	**0.0**	0.821	0.835	0.805
	9	**0.0**	0.812	0.814	0.794
	10	**0.0**	0.826	0.835	0.809
Accuracy	1	**0.793**	0.576	0.720	0.639
	2	**0.841**	0.563	0.754	0.615
	3	**0.842**	0.545	0.676	0.588

Continued on next page

Continued from previous page

Metrics	Stream	FMDD	DDM	HDDM-a	HDDM-w
	4	**0.834**	0.483	0.656	0.551
	5	**0.831**	0.501	0.676	0.610
	6	**0.836**	0.474	0.621	0.540
Accuracy	7	**0.825**	0.463	0.636	0.511
	8	**0.844**	0.466	0.635	0.521
	9	**0.830**	0.482	0.608	0.531
	10	**0.820**	0.450	0.616	0.538

As shown in the table, our method FMDD has a remarkably *high true positive rate*, especially when the distribution changes dramatically (stream 2-10), and all concept drifts are detected. On the other hand, our method also has a *high false positive rate*, which means that when there is no concept drift, a concept drift is still alerted. Nevertheless, the overall accuracy still stands out from other methods. A probable reason is that data from other data streams are used to estimate the parameters of fuzzy membership functions. Other methods only monitor the performance of learners. Once this decreases, other methods cannot determine whether the decrease is affected by concept drift or noise. Thus these methods favor notifying the presence of a concept drift when the decrease is dramatic, while our method can properly recognize the difference. At last, we can conclude the effectiveness of our drift detection method FMDD and its superior performance.

4.3.4 Evaluation of FMDA on Synthetic Data Sets

In this section, we evaluate the average mean absolute percentage error of learners using FMDA on Synthetic03 to Synthetic14.

In each stream, all 1000 samples are split into two sets: the first 900 samples are used to train and the last 100 samples are used to test. Learners trained with and without data from auxiliary streams are evaluated. The first learner (with_n) is trained using only data from the major stream. Then three learners (with_i) are trained by employing data from one of the auxiliary streams. The last learner (with_a) is trained using data from all auxiliary streams. Specifically, the learners are evaluated under the varied size of the training set. All learners are linear models with an L2 regularization term on ω and b, implemented in Scikit-Learn[‡] of version 0.24.2. The used data are randomly sampled from the training set. The experiment is conducted 5 times. To avoid effect of different scales from different data streams, we choose mean absolute percentage error to measure learners' performance. The average mean absolute percentage errors are listed in Figure 4.5

As shown in Figure 4.5, the average mean absolute percentage error decreases as the the size of training set grows. The over-fitting problem only exists when the training size is small. It is exactly the situation where concept drift occurs. Only a few data can be used to train a new learner. When the size exceeds 50, there is no significant difference between the prediction performance of learners. This is because the data are sufficient for training a new learner, so the influence is minor in determining whether to use data from auxiliary streams or not. When the training size is less than 50, it is obvious that learners trained using data from the first, second, and that all three auxiliary streams have advantages over the learner that only uses data from the major stream. The learner trained using data from the

[‡]https://scikit-learn.org

third auxiliary stream has approximate performance or even worse. This is because the distribution of the major stream is significantly different from that of the third auxiliary stream. Rather than being beneficial, using data from the third auxiliary stream is harmful in training a new learner. Specifically, a learner trained using data from all three auxiliary streams has the best prediction performance. Using data with a similar distribution can offset the negative impact of using data with a different distribution.

We can conclude that our method of using weighted data from other data streams can effectively reduce the prediction errors and solve the over-fitting problem when training a new learner after concept drift occurs. The result shows that if the distribution of the drifting stream is completely different from that of other streams, it does not help to use data from other streams to train a new learner.

4.3.5 Evaluation of FMDA on Real-world Data Sets

In this section, we evaluate the average mean squared error of learners using FMDA on two real-world data sets, Train and Weather.

To evaluate the effectiveness of FMDA on real-world data sets, we choose four concept drift adaptation methods: re-training the whole model (**Re-train**), two tree-based methods as per the Hoeffding Tree (**HT**) Domingos and Hulten (2000), and Hoeffding Adaptive Tree (**HAT**) Bifet and Gavaldà (2009), which can update part of the model, and an ensemble method Adaptive Random Forest (**ARF**) Gomes et al. (2017b). All the methods are implemented in scikit-multiflow[§] of version 0.5.3, and all are conducted on the two real-world data sets independently. FMDD is used to detect whether concept drift occurs. Once alerted that a concept drift has occurred, the model is re-trained or partially trained on the new data. Then FMDA is used to support these four methods. In re-training or partial training, FMDA determines

[§]https://scikit-multiflow.readthedocs.io

the correlation of data streams and gives corresponding fuzzy membership. The membership is used as the training weight of data from other data streams. The batch size is set as 50. The batch size is a very important parameter in our method. If the amount of data is sufficient for re-training, the over-fitting problem does not exists. We give a detail batch size study in the following section. The results are summarized in Table 4.3 and Table 4.4.

The results show that our method can significantly improve the prediction accuracy of Re-train and ARF. However, the effectiveness of two tree-based methods, HT and HAT, is noteworthy. For Hoeffding Tree, on Streams 3, 6, 7 and 8 of the Train data set, Streams 1, 3, 4, 6, 9 and 10 of the Weather data set, FMDA leads to larger errors. For Hoeffding Adaptive Tree, only on Stream 4 and 8 of the Train data set, Stream 1, 6 and 10 of the Weather data set, FMDA shows its advantages. This is because, in these two tree-based methods, new data are only used to train a new sub-tree. The complexity of the sub-tree is usually small, and the over-fitting problem is not obvious. Data from other streams can lead to negative effects instead. The result validates the mechanism of our method again. FMDA uses data from other data streams to solve the over-fitting problem. If the problem is not obvious, our method still shows few advantages, even negative effects.

In summary, the efficiency of FMDA is validated by the experiments.

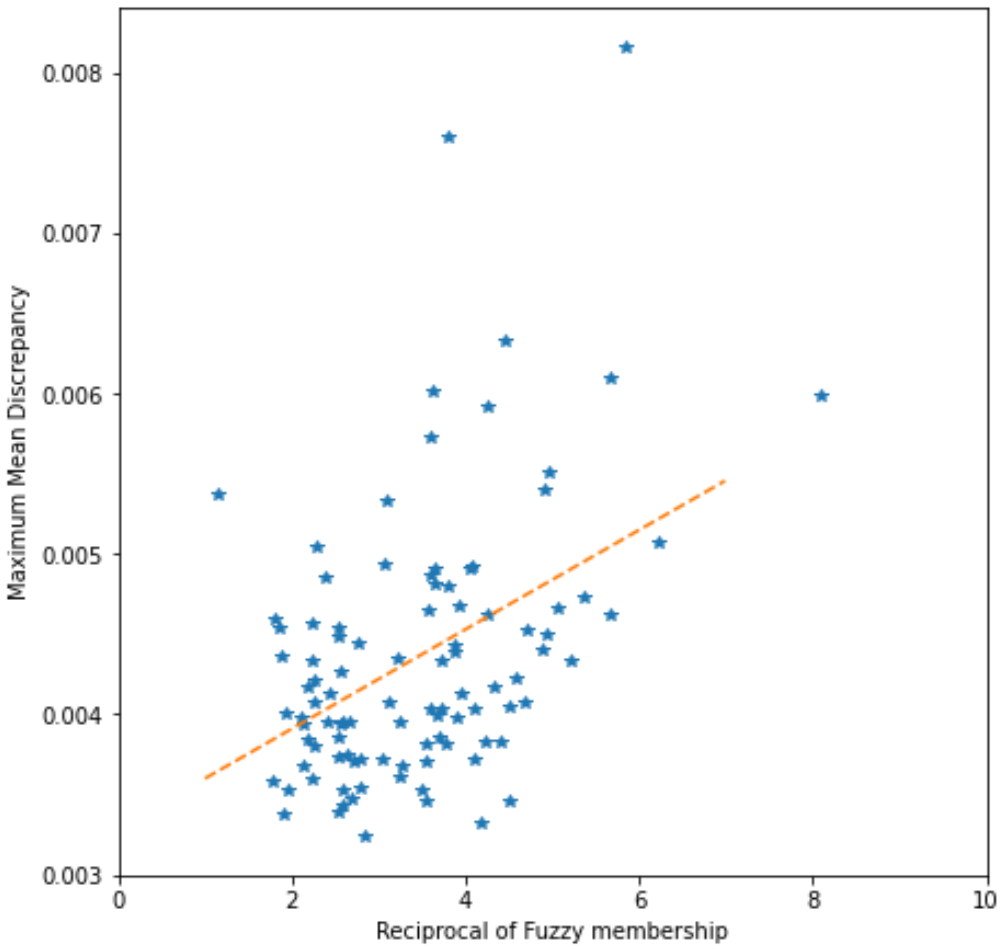


Figure 4.4 : Correlation between the MMD and the reciprocal of fuzzy membership. There exists a positive correlation between the reciprocal of our proposed fuzzy membership and MMD. However, the average computing time of fuzzy membership is 1.039e-4 seconds, while the average computing time of MMD is 2.419e-2 seconds.

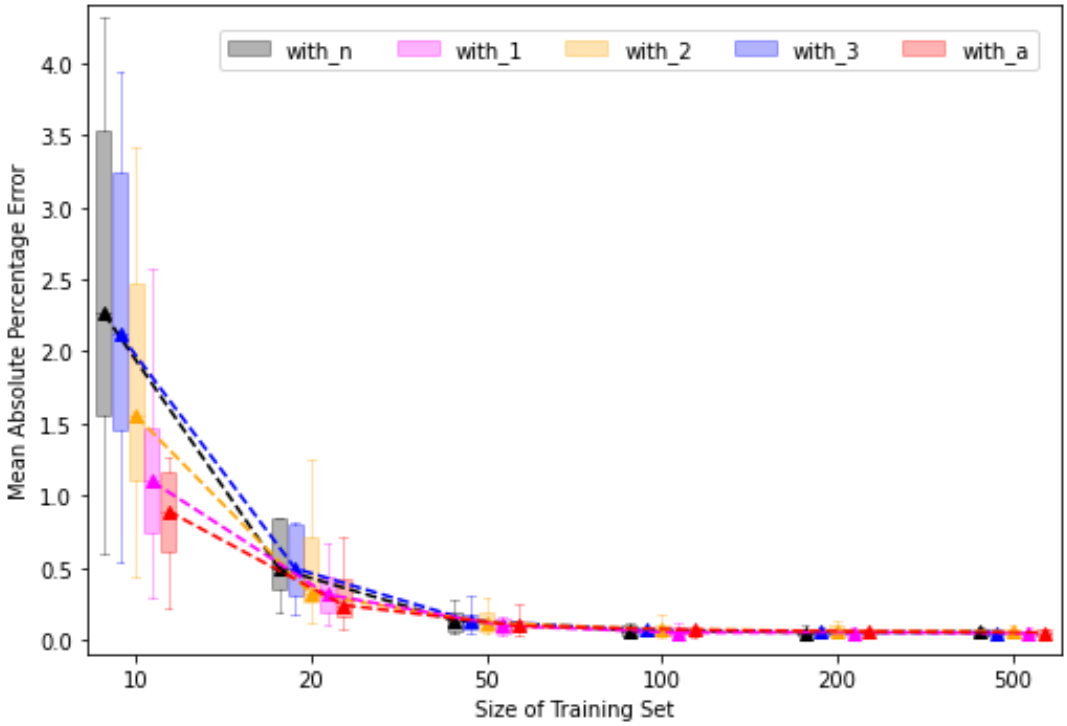


Figure 4.5 : Results of evaluation of FMDA on synthetic data sets. The x-axis is the size of training set and the y-axis is the average mean absolute percentage error. The box plot show the medians and quartiles of average mean absolute percentage errors of twelve data sets. The medians are connected by dashed lines.

Table 4.3 : Results of evaluation on Train

Method	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7	Stream 8
Re-train	2.4093	1.0914	0.9810	0.9201	1.2001	1.0416	1.3787	**0.6097**
Re-train + FMDA	**2.3324**	**0.7526**	**0.8493**	**0.8737**	**0.9777**	**0.9610**	**1.0439**	0.6624
HT Domingos and Hulten (2000)	**2.6466**	2.3151	**2.3194**	2.2406	2.2602	**2.2974**	**2.2571**	**2.2601**
HT + FMDA	3.0263	**2.2914**	2.3339	**2.2200**	**2.2490**	2.3401	2.2639	2.2760
HAT Bifet and Gavaldà (2009)	**2.7727**	**2.3383**	**2.3473**	2.2798	**2.2636**	**2.3502**	**2.2659**	2.3434
HAT + FMDA	3.2551	2.3722	2.3815	**2.2599**	2.3040	2.3963	2.3180	**2.3368**
ARF Gomes et al. (2017b)	2.8249	**0.4744**	2.2816	2.2144	**2.2304**	2.3270	2.2586	2.2452
ARF + FMDA	**2.7209**	0.6567	**2.0314**	**2.0089**	0.3760	**2.0555**	**2.0247**	**2.1167**

Table 4.4 : Results of evaluation on Weather

Method	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7	Stream 8	Stream 9	Stream 10
Re-train	0.1724	0.1349	0.2304	**0.0916**	0.1597	0.1608	0.2615	0.4146	0.2364	0.2302
Re-train + FMDA	**0.1608**	**0.1303**	**0.1949**	0.0976	**0.1023**	**0.1271**	**0.2322**	**0.4035**	**0.1801**	**0.1882**
HT	**0.0696**	0.0648	**1.5707**	**0.0505**	0.0651	**1.4788**	0.1703	0.1436	**0.1041**	**0.1032**
HT + FMDA	0.4334	**0.0487**	2.1482	0.0840	**0.0590**	4.5118	**0.1222**	**0.1249**	2.2996	1.8641
HAT	0.1038	**0.0470**	**0.3813**	**0.0547**	**0.0706**	2.7442	**0.1464**	**0.1421**	**0.1014**	0.0859
HAT + FMDA	**0.0722**	0.0589	2.1503	0.2118	0.1119	**0.1177**	0.1944	0.1645	2.6239	**0.0755**
ARF	0.0346	0.0293	0.0292	**0.0156**	**0.0526**	0.2393	0.3285	0.1031	0.0323	0.0471
ARF + FMDA	**0.0242**	**0.0277**	**0.0228**	0.2036	0.0828	**0.0291**	**0.0415**	**0.0630**	**0.0248**	**0.0340**

4.3.6 Batch Size Study on Real-world Data Sets

As shown in the above section, the over-fitting problem exists when the data to train a new learner are insufficient. Under our assumption, data arrive batch by batch. When drift is detected, only the current batch of data can be used to re-train. Hence the size of batch is key to our method. If the the batch is large enough, the over-fitting problem does not exist. Data in the current batch is sufficient to train a new learner. It is not necessary to use data from other data streams. Like the situation in Figure 4.5 when the training set size is greater than 50, there is no significant difference in whether or not to use data from auxiliary streams. Therefore, in this section, we conduct a study of the impact of the batch size on two real-world data sets.

Two methods are compared in this section:

- **Baseline**: Only data in the current batch of drifting streams are used to train new learners after concept drift is detected.
- **FMDA**: Once concept drift is detected, both data of drifting streams and weighted data of non-drifting streams are used to train new learners.

The learners in both methods are decision trees implemented in Scikit-Learn of version 0.24.2. The batch size varies from 20 to 500. The experiment is conducted 5 times. The results are shown in Figure 4.6 and Figure 4.7.

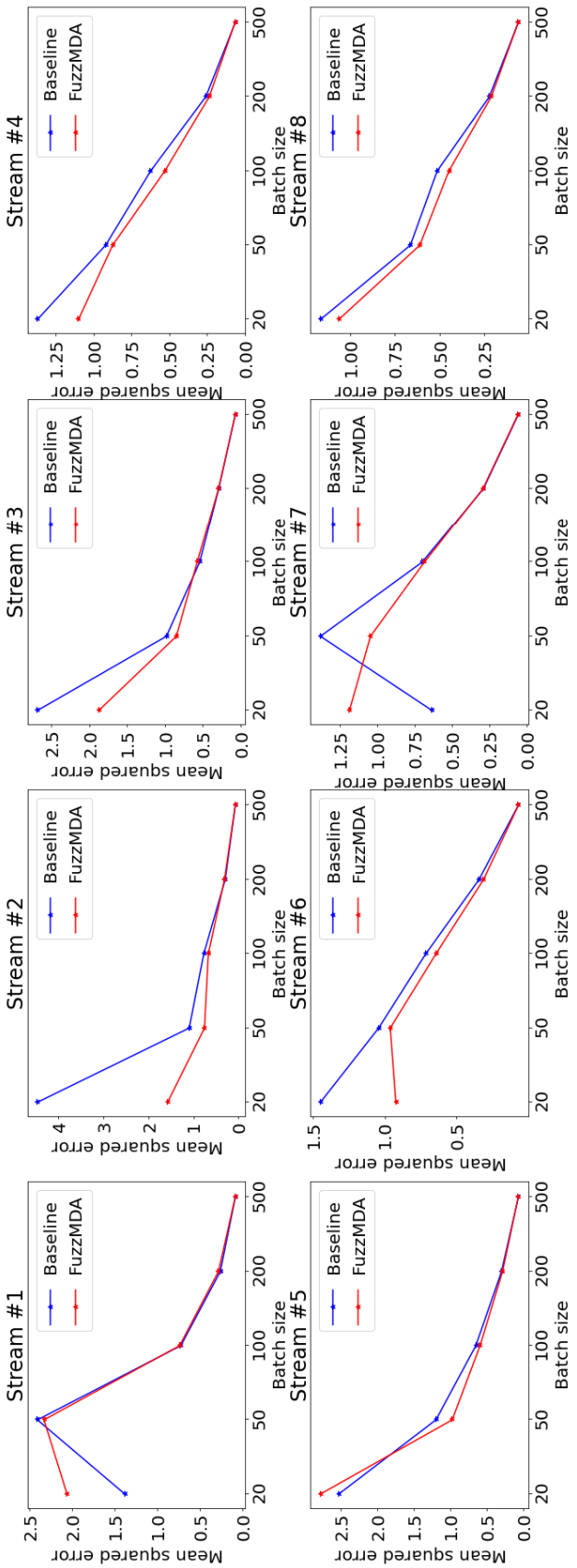


Figure 4.6 : Results of batch study on Train data set. Each sub-figure shows result of a data stream. There are eight streams in total. The x-axis is the size of batch and the y-axis is the average mean squared error.

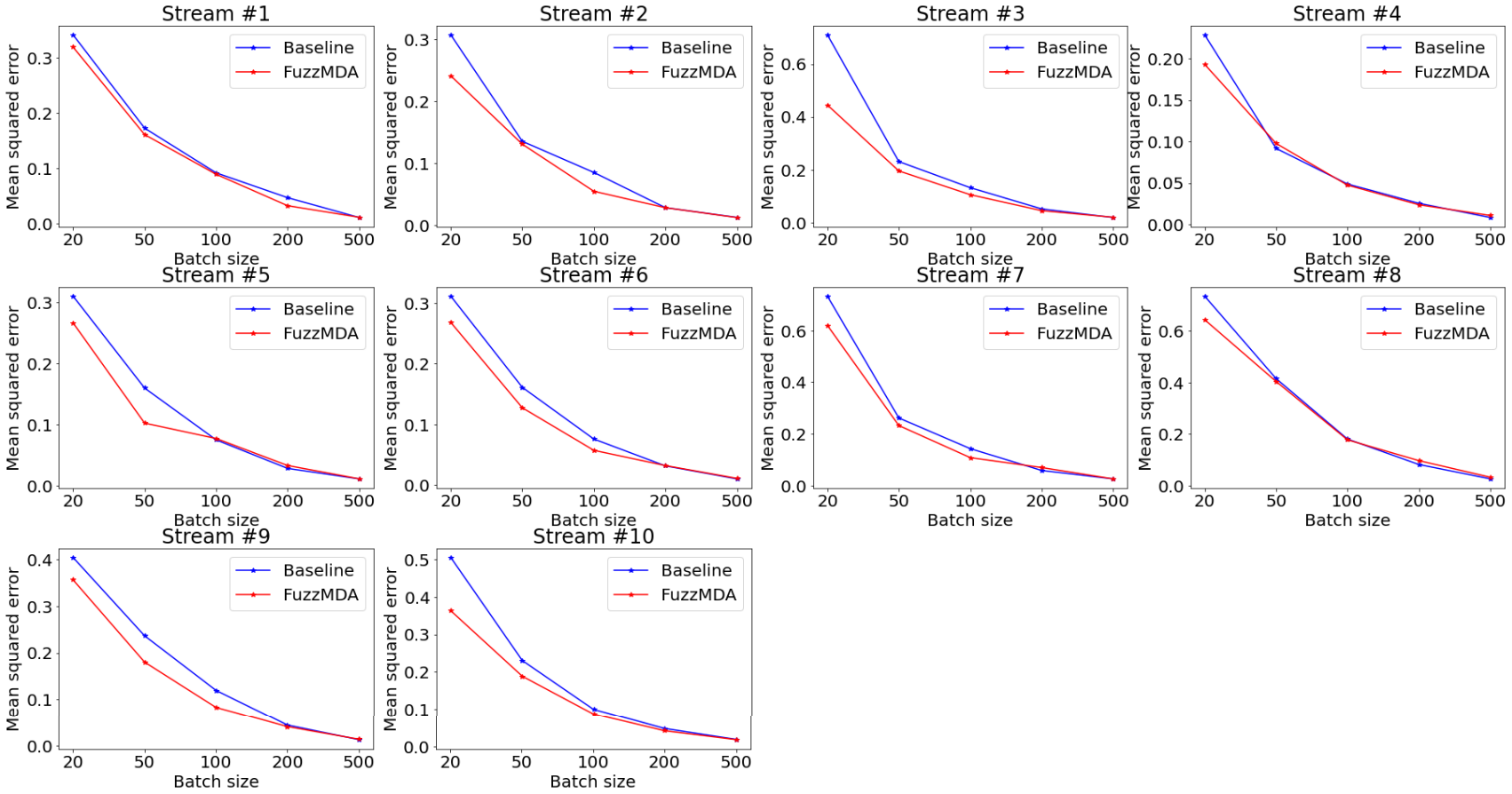


Figure 4.7 : Results of batch study on Weather data set. Each sub-figure shows result of a data stream. There are ten streams in total. The x-axis is the size of batch and the y-axis is the average mean squared error.

As shown in Figure 4.6, in the Train data set, the prediction errors decrease as the batch size grows. When the batch size is small, the improvement of our method is significant. Just like results on synthetic data sets, when the batch size is small, there are not sufficient data to train new learners. The over-fitting problem is encountered. Our method uses weighted data from non-drifting streams to solve the over-fitting problem. With the increase in the amount of data, the prediction performance of new learners is improved. Currently, data from other streams lead to poor performance due to the different distribution. Note that in Stream 1 and Stream 7, the prediction errors are significantly smaller than in our method when the batch size is 20. This is because the batch size is too small. There exists some noise in data, which cannot be ignored in this situation. The maximum mean squared error of Stream 1 is 922.2667, while 0.1028 of average mean squared error. Occasional extreme values cause the results.

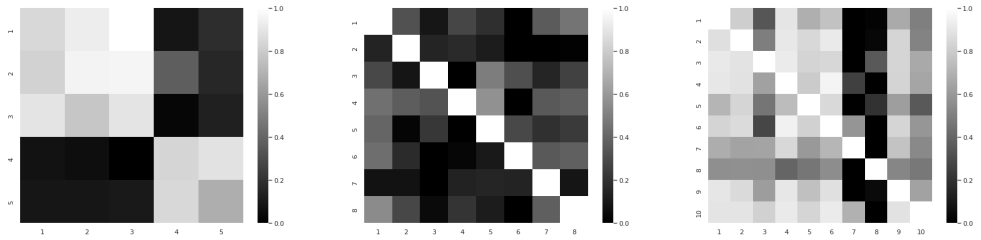
In Figure 4.7, all ten data streams show the same pattern. Reduction in prediction errors is also shown compared with the baseline. And the prediction errors decrease as the batch size increases. When the batch size is large, data from other streams show a disadvantage compared with the baseline. When the batch size is larger than 200, the curves become flat, which means that the over-fitting problem does not exist. This is different from the Train data set.

This experiment provides a reference for us to choose a proper batch size for these two real-world data sets. It also validates the effectiveness of FMDA when the batch size is small. Therefore, we can conclude when the batch size of the data stream is small, the over-fitting problem can be alleviated by our method, using weighted data from non-drifting data streams to train a new learner. Meanwhile, the situation of Stream 1 and Stream 7 in the Train data set implies that FMDA is not suitable when the batch size is extremely small, smaller than 20 in Train data set for example. The occasional extreme values can cover the improvement of FMDA.

4.3.7 Visualization of the Correlation between Data Streams

In this section, we give a visualization study of fuzzy membership.

One of advantages of fuzzy technique is its interpretability. Machine learning models are usually considered as a black box lack of interpretability. In our Multi-stream Concept Drift Handling Framework, the membership can interpret the correlation between data streams to support decision making. We first visualize the correlation between five data streams in Synthetic15, which is shown in Figure 4.8a. Each block reflects the correlation of two streams. The lighter the block is, the more significant the correlation is. In Synthetic15, all Stream 1-3 are generated from a distribution, and both Stream 4 and Stream 5 are generated from another distribution. The Figure 4.8a show that the nine blocks of the top left corner are light, which implies the correlation of Stream 1-3 are significant. The four blocks of the lower right corner are light, which implies the correlation of Stream 4 and Stream 5 is significant. This is consistent with the true correlation. As for real-world data sets, we do not know the real correlation between data streams, the visualization of batches of samples in Train and Weather data sets are shown in Figure 4.8b and Figure 4.8c. The visualization is clear for recognize the correlation between data streams and to trace the changes of the correlation, which is useful for artificial decision making.



(a) Synthetic15 (b) Train (c) Weather

Figure 4.8 : Visualization of the correlation between data streams. The color of block means the correlation between two data streams represented by fuzzy membership.

4.4 Summary

In this chapter, we propose a Multi-stream Concept Drift Handling Framework, to alleviate the over-fitting issue in the re-training procedure after concept drift. We design a stream fuzzy set with membership to measure the degree to which the samples belong to a data stream. Based on the membership of samples, a Mann-Whitney U Rank Test is used to detect when and in which stream concept drift occurs. To manage the over-fitting issue due to a lack of data when re-training a new learning model after concept drift occurs, we design a weighting method based on the membership. Weighted data from other non-drifting data streams are added to the training set. By increasing the volume of data in the training set, the over-fitting problem is alleviated. Experiments on both synthetic and real-world data sets show that our method can improve the performance of the new learning model being re-trained after concept drift occurs.

Chapter 5

Evolutionary Regressor Chains for Multi-stream Regression

5.1 Introduction

Currently, most data stream regression researchers treat each data stream independently. However, in many real-world scenarios, data streams exist simultaneously. There are usually some correlation between multiple data streams. For instance, Harries et al. Harries (1999) recorded the electricity price in both NSW and VIC in Australia. The trend of price is shown in Figure 4.2. It is obvious that there exists a significant correlation between the electricity prices of the two states. However, traditional methods for data stream regression always build an exclusive model for each single data stream. The exclusive model has no ability to receive information from other correlational data streams to utilize the correlation between data streams. This lack of information from other data streams leads to poor performance.

Chandra et al. Chandra et al. (2016) are the first to take the correlation into account. It is assumed that in one data stream, called the target data stream, the labels are not available in time. Another labeled data stream, called the source data stream, is needed to help train a model on the target data stream and update it. However, in many real-world applications, more than two data streams exist simultaneously. In this situation, all data streams are target data streams. It is hard to determine which data streams should be chosen as source data streams for other data streams. Hence the correlation of data streams should be considered

in an overall view. An intuitive idea is to treat the output of the model on one data stream as the input of the model on another data stream. If these two data streams are correlational, this method introduces information from one data stream to a correlational data stream in the form of additional features in the model, which can lead to improvement in the performance of the model. However, it is an $O(m^2)$ problem to track all pair-wise correlation between data streams, which is impossible when the number of data streams m is large.

Another challenge of data stream regression is its dynamicity compared with traditional regression problems. The properties of data streams can change over time. The dynamicity is manifested in two main aspects: data dynamicity and correlation dynamicity.

- Data dynamicity refers to the fact that the distribution of data is non-stationary. Models built on historical data do not work on newly arriving data when the distribution of the data has changed. This phenomenon is usually referred to as concept drift Lu et al. (2019) in the literature.

- In the multi-stream regression scenario, the correlation of data streams can also change, which reflects the dynamicity of the correlation. We call this phenomenon correlation drift.

Thus, we should not only recognize the correlation between data streams, but also track the changes in the correlation and adapt to them.

Now we have three problems to solve: 1) how to recognize the correlation between data streams in an overall view; 2) how to track the changes in the correlation in data streams; and 3) how to adapt to concept drift. To solve these three problems in the multi-stream regression scenario, we propose an ensemble chain-structured model named Evolutionary Regressor Chains.

Regressor Chains As mentioned above, it is impossible to track all correlation when the number of data streams is large. A similar dilemma is faced in multi-target regression. Xioufis et al. Spyromitros-Xioufis et al. (2016) proposed a chain structure, named Regressor Chains, to track the correlation between multiple targets. Inspired by Regressor Chains, we also use a chain structure to track the correlation between data streams. All data streams are shuffled into a chain structure in a random order. The corresponding machine learning models are built in sequence. The model of the first data stream is built as per usual. Predictions of the first model are treated as additional features for the model of the second data stream. The second model treats both its original features and the additional features, i.e., the predictions from the first model, as input. Information from the first data stream can be tracked by the second model through the additional feature if the first two data streams are correlational. The third model is built with an additional feature, which is the predictions of the model of the second data stream, and so on, with all models being organized in a chain structure in the same order as the data streams.

Heuristic Order Searching The main drawback of Regressor Chains is that the performance is sensitive to the order of the chain. The order of the chain is key to tracking the correlation, and the method suffers from a poorly ordered chain. However, the order is generated randomly; it should not be expected to track the correlation correctly. It is impossible to search all possible order and choose the best order because it is an $O(m!)$ problem. We design a heuristic order searching strategy to find the optimal order of the chain. Multiple chains are generated as part of the initialization procedure. The initial order is generated randomly as in Regressor Chains. The performance of the models under the order is recorded. Then a new order of the chain is iteratively generated based on the performance of the models. If the performance of the model on Stream i, which treats the output of the

model on Stream j as input, is high, it is more likely that Stream j follows Stream i in the new order. A better order can be obtained after several iterations. All the chains constitute an ensemble and the average prediction is considered as the final prediction. We name the Regressor Chains with heuristic order searching strategy as Evolutionary Regressor Chains.

Adaptation to Dynamicity To adapt to the dynamicity in data streams, the models are updated online. Once new data arrive, a gradient descent-based method is utilized to update the parameters of the models in the Evolutionary Regressor Chains. The model is updated online to adapt to the change in data distribution, i.e. concept drift. The order of chains is also updated online. Heuristic order searching is utilized to update the order of the chains once new data arrive, which enables Evolutionary Regressor Chains to adapt to correlation drift.

Diversity Pruning In addition, we propose a diversity pruning method. A large number of chains are generated to search for the optimal order; However, it is unwise to use all these chains, as it would lead to high computation complexity. Diversity is important in ensemble learning, because nothing can be obtained if all members of an ensemble are the same. Thus, we prune the chains according to diversity. In our method, not all chains in the model, but chains with maximum diversity are used when predicting. This also makes the model more robust in resisting the potential impact from minor drift.

The main contributions of our work in this sphere can be summarized as follows:

- We propose a chain-structured model, Evolutionary Regressor Chains, to track the correlation among multiple data streams. We firstly take the correlation between more than two data streams into account to improve the performance of the models in data stream regression, which has not been solved by existing

research.

- To overcome the drawback that the randomly generated chain order cannot track the correlation correctly, we design a heuristic order searching strategy. The method updates the order iteratively to find an optimal order.
- We design an online updating strategy to update the models in the multi-stream regression scenario. This strategy can effectively adapt to both concept drift and correlation drift.
- We propose a diversity pruning method to decrease the complexity of our method while maximizing the diversity of the ensemble. It can also increase the robustness of our method.
- We analyze some theoretical properties of our method and give its dynamic regret bound.
- We conduct an experimental study on some real-world data sets. The results show the efficiency of our method compared with other state-of-the-art methods.

The rest of this chapter is organized as follows: in Section 5.2, we give a brief review of the related work. The problem settings of multi-stream mining are presented in Section 5.3. The detail of our method is given in Section 5.4. We describe some theoretical analysis in Section 5.5. In Section 5.6, we discuss the experimental results. Our conclusions are put forward in Section 5.7.

5.2 Problem Settings

Firstly, we re-state the formal definition of data streams give in Definition 1.

Definition 4. *A data stream* $S = \{(X_1, y_1), (X_2, y_2), \dots, (X_t, y_t), \dots\}$ *is a real-time, dynamic and infinite data sequence, where* (X_t, y_t) *is an instance at time* t*,* $X_t \in \mathcal{X}$ *is the feature and* $y_t \in \mathcal{Y}$ *is the target variable to be predicted.*

In a supervised learning setting, given a data stream S, X_t and y_t obey a specific distribution $P(X_t, y_t)$. A machine learning model is a map $f : \mathcal{X} \to \mathcal{Y}$. Given the feature X_t of an instance, $f(X_t)$ is to approximate the conditional expectation $E(y_t|X_t)$.

Next we give the definition of correlational data streams.

Definition 5. *Given two data streams,* $S_1 = \{(X_1^{(1)}, y_1^{(1)}), (X_2^{(1)}, y_2^{(1)}), \dots, (X_t^{(1)}, y_t^{(1)}), \dots\}$ *and* $S_2 = \{(X_1^{(2)}, y_1^{(2)}), (X_2^{(2)}, y_2^{(2)}), \dots, (X_t^{(2)}, y_t^{(2)}), \dots\}$*,* S_1 *and* S_2 *are correlational at time* t*, if*

$$P(y_t^{(1)}, y_t^{(2)}) \neq P(y_t^{(1)})P(y_t^{(2)}).$$

The correlation of data streams must be taken into account to enable the machine learning model to predict more precisely. If S_1 and S_2 are correlational, the machine learning model of S_1 must be to approximate the conditional expectation $E(y_t^{(1)}|X_t^{(1)}, y_t^{(2)})$.

Given m data streams, based on the correlation of each two data streams, we obtain the correlation graph of all m data streams.

Definition 6. *Given* m *data streams* $\mathcal{S} = \{S_i\}_{i=1}^m$*, where some data streams are correlational, the correlation graph at time* t *is a graph* $\mathcal{G}_t = (\mathcal{S}, \mathcal{E})$*, where the vertex* $S_i \in \mathcal{S}$ *is a data stream, and the edge* $e_{i,j} \in \mathcal{E}$ *means that the two corresponding data streams* S_i *and* S_j *are correlational at time* t*.*

Remark 1. *Given* m *data streams* $\mathcal{S} = \{S_i\}_{i=1}^m$ *with the correlation graph* $\mathcal{G}_t = (\mathcal{S}, \mathcal{E})$*, the challenge of the multi-stream mining problem is how can the model of* S_i

take as much information as possible into account from $S_j \in \mathcal{S}$ *that is connected with* S *in* $\mathcal{G}_t$.

Dynamicity is a distinctive feature of data streams. That is to say, the properties of data streams can change over time. The phenomenon that $P(X_t, y_t)$ change over time is called concept drift Gama et al. (2014). Likewise, the correlation of data streams can also be non-stationary, which we name correlation drift.

Definition 7. *Given* m *data streams* $\mathcal{S} = \{S_i\}_{i=1}^m$, *where some data streams are correlational, correlation drift occurs if* $\exists\ t_1, t_2$, *such that*

$$\mathcal{G}_{t_1} \neq \mathcal{G}_{t_2}.$$

Remark 2. *Given* m *data streams* $\mathcal{S} = \{S_i\}_{i=1}^m$ *with the correlation graph* $\mathcal{G}_t = (\mathcal{S}, \mathcal{E})$, *the multi-stream mining model must be adaptive, such that the model can update itself when* $\mathcal{G}_{t_1} \neq \mathcal{G}_{t_2}$.

5.3 Proposed Method

This section details our method, Evolutionary Regressor Chains.

5.3.1 Regressor Chains

Inspired by the Regressor Chains Spyromitros-Xioufis et al. (2016), we propose a chain structure to model as much stream correlation as possible. Given m data streams $\{S_i\}_{i=1}^m$, a Regressor Chain is defined as $C = \{F, O\}$, where $F = \{f^{(i)}\}_{i=1}^m$ is a collection of machine learning models, and $O = (o_1, o_2, \ldots, o_m)$ is an m-permutation. An m-permutation is a permutation from 1 to m. For example, $(5, 2, 1, 3, 4)$ and $(1, 4, 3, 5, 2)$ are two 5-permutations. All data streams are sorted in order of O at initialization, $\{S_{o_1}, S_{o_2}, \ldots, S_{o_m}\}$. Then m machine learning models $\{f^{(i)}\}_{i=1}^m$ are built in sequence. The first model $f^{(1)}$ is trivial, and takes $X_t^{(o_1)}$ as input and $y_t^{(o_1)}$ as output. After this, the predictions of $f^{(1)}$ are

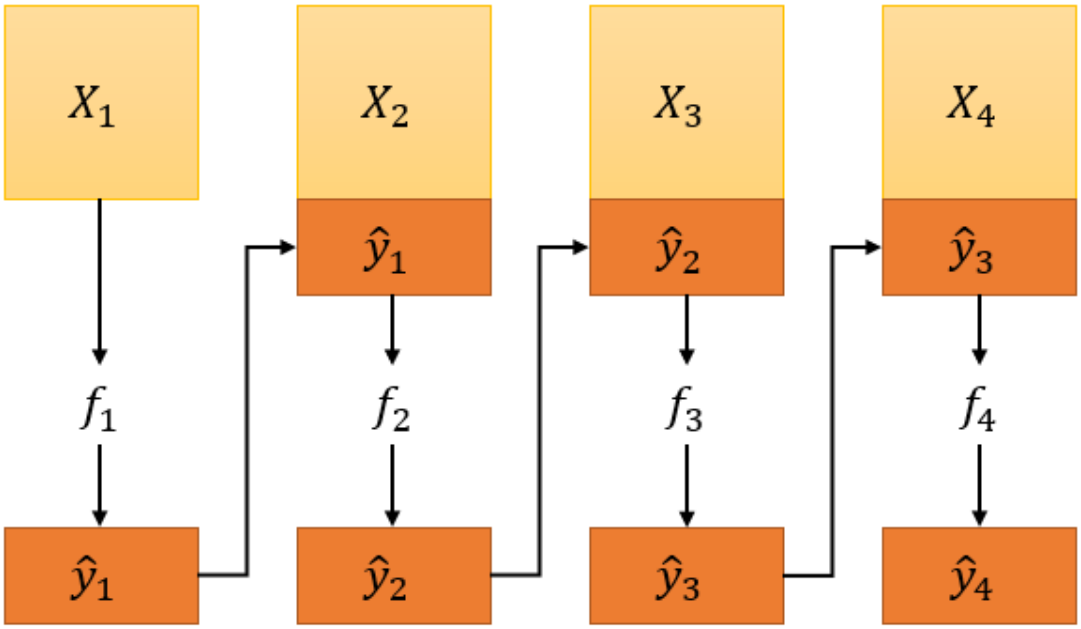


Figure 5.1 : An illustration of a basic chain structure built on four data streams

obtained as $\hat{y}_t^{(o_1)} = f^{(1)}(X_t^{(o_1)})$. The aim of the second model $f^{(2)}$ is to approximate $P(y_t^{(o_2)}|X_t^{(o_2)}, y_t^{(o_1)})$. Therefore both $X_t^{(o_2)}$ and $\hat{y}_t^{(o_1)}$ are considered as the input of $f^{(2)}$. Let $X_t'^{(o_2)} = \{X_t^{(o_2)}, \hat{y}_t^{(o_1)}\}$. $f^{(2)}$ takes $X_t'^{(o_2)}$ as input and $y_t^{(o_2)}$ as output. The third model $f^{(3)}$ treats $\hat{y}_t^{(o_2)}$ as an additional features. Let $X_t'^{(o_3)} = \{X_t^{(o_3)}, \hat{y}_t^{(o_2)}\}$. Then $f^{(3)}$ takes $X_t'^{(o_3)}$as input and $y_t^{(o_3)}$ as output, and so on. The last model $f^{(m)}$ treats $\hat{y}_t^{(o_{m-1})}$ as an additional feature. Let $X_t'^{(o_m)} = \{X_t^{(o_m)}, \hat{y}_t^{(o_{m-1})}\}$. Then $f^{(m)}$ takes $X_t'^{(o_m)}$ as input and $y_t^{(o_m)}$ as output. Finally, m models $\{f^{(i)}\}_{i=1}^m$ are constructed, organized as a chain. Figure 5.1 demonstrates how the models are organized in a chain structure.

5.3.2 Heuristic Order Searching

It is obvious that the order of the chain is essential to our model. If models of two data streams with no correlation are adjacent in the chain, the additional input from the former irrelevant data stream does not help the prediction of the latter

model. Moreover, the models of two related data streams must be adjacent in the chain for the latter model to take advantage of information from the former. A proper order means that it can learn as much correlation as possible from all data streams. Hence, the next important step is to find a proper order for our model.

In Spyromitros-Xioufis et al. (2016) one suggested solution is to generate multiple chains randomly and aggregate them into an ensemble. The averaging prediction of a randomly generated ensemble can improve the results of chains with a bad order, but it also weakens the result of those with a proper order. To find the proper chain order, an optimization problem needs to be solved,

$$C = \{F, \underset{O \in \mathcal{P}_m}{\operatorname{argmin}} \sum_{i=1}^{m} \mathcal{L}(f^{(i)})\}, \tag{5.1}$$

where $\mathcal{P}_m$ is the set of all m-permutations, and $\mathcal{L}(f^{(i)})$ is the loss of model $f^{(i)}$. For a chain of length m, there are $m!$ different orders. It is impossible to enumerate all possible orders to find the best when m is large. It becomes an NP-complete problem.

As a consequence, we turn our attention to heuristic algorithms. Inspired by the Max-Min Ant System Stützle and Hoos (2000), a variant of Ant Colony Optimization (ACO) Dorigo et al. (1996), we propose a heuristic order searching strategy to find a suitable order for our model. First, we generate $m \times m$ machine learning models $\{g_{i,j}\}_{i,j=1}^{m}$, where $g_{i,j}$ takes both $X_t^{(j)}$ and $y_t^{(i)}$ as input and $y_t^{(j)}$ as output, and $g_{i,i}$ takes only $X_t^{(i)}$ as input without additional features and $y_t^{(i)}$ as output. Let $C = \{F, O\}$ be a Regressor Chain. The order $O = (o_1, o_2, \ldots, o_m)$ is generated randomly. The first model $f^{(1)}$ is assigned to g_{o_1,o_1}. The jth model $f^{(j)}$ is assigned to g_{o_{j-1},o_j}. Note that we use two kinds of symbols to refer to machine learning models. The symbol $f^{(i)}$ refers to the ith model of the chain, while the symbol $g_{i,j}$ refers to the model which takes $X_t^{(j)}$ and $y_t^{(i)}$ as input and $y_t^{(j)}$ as output. It should not be too difficult to distinguish them. An $m \times m$ table Q is maintained to

record the performance of these $m \times m$ models, and all values in Q are initialized as 1. The role of the values in the Q table is similar to the pheromone in an ant colony system. For iteration t, for the convenience of the demonstration, we denote the Evolutionary Regressor Chain as $C_t = \{F_t, O_t\}$, where $F_t = \{f_t^{(i)}\}_{i=1}^m$. The ith model of the chain $f_t^{(i)}$ can give the prediction $\hat{y}_t^{(o_i)} = f_t^{(i)}(X_t^{(o_i)})$, and the loss $\mathcal{L}(f_t^{(i)})$ is obtained. Then a new order $O'_t = (o'_1, o'_2, \ldots, o'_m)$ is generated. Firstly o'_1 is chosen from a multinomial distribution $\mathcal{MN}(p_1)$. The probability of choosing i is calculated by

$$p_1[i] = \frac{Q[i,i]}{\sum_{k=1}^m Q[k,k]}.$$

The rest element o'_i is generated from a multinomial distribution $\mathcal{MN}(p_i)$. The probability of choosing j as o_i is calculated by

$$p_i[j] = \frac{Q[o_{i-1}, j]}{\sum_{k \neq o_{i'}, i' < i} Q[o_{i-1}, k]}.$$

Then all the models are assigned to new ones by $f_t'^{(1)} = g_{o'_1, o'_1}$ and $f_t'^{(j)} = g_{o'_{j-1}, o'_j}$ for $j > 1$. With the new order, the loss of models $\mathcal{L}(f_t'^{(i)})$ can be obtained. In this iteration, the order of Evolutionary Regressor Chain O_{t+1} will be updated to O'_t if the new order O'_t takes advantage of the old one O_t,

$$O_{t+1} = \begin{cases} O'_t & \text{if} \sum_{i=1}^m \mathcal{L}(f_t^{(i)}) > \sum_{i=1}^m \mathcal{L}(f'^{(i)}_t), \\ O_t & \text{otherwise.} \end{cases} \tag{5.2}$$

Then we can update the Q table by

$$Q_{t+1}[i,j] = \begin{cases} \min\{(1-\rho)Q_t[i,j] + \rho, q_{\max}\} & \text{if} g_{i,j} \in F_t \\ \max\{(1-\rho)Q_t[i,j], q_{\min}\} & \text{otherwise.} \end{cases} \tag{5.3}$$

where ρ is the evaporation factor that controls the evaporation speed of the values in the Q table. In addition, the values in the Q table are bounded in the interval $[q_{\min}, q_{\max}]$, which is introduced in Stützle and Hoos (2000) to make the algorithm

more robust. For the first model of the chain, the Q table is updated by

$$Q_{t+1}[i,i] = \begin{cases} \min\{(1-\rho)Q_t[i,i] + \rho, q_{\max}\} & \text{if} f_t^{(1)} = g_{i,i} \\ \max\{(1-\rho)Q_t[i,i], q_{\min}\} & \text{otherwise.} \end{cases} \tag{5.4}$$

According to Equation (5.2), the performance is always increased or at least the same after updating the order. Thus, for $g_{i,j} \in F_t$, the corresponding value in Q the table should increase, so that it is more likely to be chosen in the next iteration. The other values in the Q table all evaporate according to ρ. The random choice enables the Evolutionary Regressor Chain to explore other unseen models with lower loss. This can avoid the problem of being trapped in the local optimum.

To find the optimum as soon as possible, like most heuristic optimization algorithms, we generate multiple Evolutionary Regressor Chains $\{C_i\}_{i=1}^{n}$. All chains are with the heuristic order searching procedure at the same time.

5.3.3 Online Training Procedure

To deal with both concept drift and correlation drift, the training procedure is processed in an online way. After new data arrived at time t, the predictions can be obtained by

$$\hat{y}_t^{(o_i)} = f_t^{(i)}(X_t^{(o_i)}),$$

and the loss $\mathcal{L}(f_t^{(i)})$ is obtained by

$$\mathcal{L}(f_t^{(i)}) = ||\hat{y}_t^{(o_i)} - y_t^{(o_i)}||.$$

The parameters $\omega_t^{(i)}$ of model $f_i^{(i)}$ are updated as follows,

$$\omega_{t+1}^{(i)} = \omega_t^{(i)} - \eta \nabla \mathcal{L}(f_t^{(i)}), \tag{5.5}$$

where $\nabla \mathcal{L}(f_t^{(i)})$ is the gradient, and η is the learning rate to control the step length towards the direction of the gradient descent. The whole training procedure is summarized in Algorithm 6. The time complexity of every training step is

Algorithm 5 Procedure to generate a new order for an Evolutionary Regressor Chain based on table Q

Require: m, table Q

Ensure: $O' = (o'_1, o'_2, \ldots, o'_m)$

Initialize an array p of size m with 1s
for $i = 1$ **to** m **do**
  for $j = 1$ **to** m **do**
    if $p[j] \neq 0$ **then**
      if $i == 1$ **then**
        Let $p[j] = \frac{Q[j,j]}{\sum_{k=1}^{m} Q[k,k]}$
      else
        Let $p_i[j] = \frac{Q[o'_{i-1},j]}{\sum_{k \neq o'_{i'}, i'<i} Q[o'_{i-1},k]}$
      end if
    end if
  end for
  Update o'_i with a random number from a multinomial distribution $\mathcal{MN}(p)$
  Let $p[o'_i] = 0$
end for

$\Theta(MN(D + P + M^2))$, where M is the number of data streams, N is the number of chains, D is the number of data features, and P is the number of parameters in a machine learning model.

In our method both the training procedure and heuristic order searching are processed online. The former deals with concept drift, while the latter handles both optimal order searching and correlation drift.

5.3.4 Diversity Pruning

In the training procedure, n chains are generated. The more chains that are produced, the higher the probability of finding the optimal order. However, in the predicting procedure, it is not necessary to use all the n chains. Thus, we design a pruning method based on the diversity of the ensemble of chains. There are two main reasons for choosing diversity:

- Diversity is an important measure in ensemble learning. It loses its significance if all chains in the ensemble make the same predictions; in that case it makes no difference if only one chain in the ensemble is used. The diversity is key to improving the performance of an ensemble of chains.

- Diversity can make the model more robust. If the diversity is not large enough, all chains make similar predictions. A minor correlation drift will lead to a significant reduction in performance. Increasing diversity can avoid this.

Given n Evolutionary Regressor Chains $\{C_i\}_{i=1}^{n}$, and new data $(X_t^{(1)}, y_t^{(1)}), \ldots,$ $(X_t^{(m)}, y_t^{(m)})$ at time t, let $\hat{Y}_t^{(i)}$ be the prediction of the ith chain C_i. We use the norm of difference between two predictions as the measure of the diversity of two chains

$$\text{Diversity}_t(i,j) = ||\hat{Y}_t^{(i)} - \hat{Y}_t^{(j)}||. \tag{5.6}$$

If the norm of the difference between two predictions is large, i.e. the two models give totally different predictions, it is obvious that the diversity is large. If it is small, the two models are similar. Then we can calculate $\text{Diversity}_t(i,j)$ for all $1 \leq i < j \leq n$. To choose n' chains in n Evolutionary Regressor Chains, two chains with maximum diversity C_{n_1} and C_{n_2} are chosen firstly.

$$\text{Diversity}_t(n_1,n_2) = \max_{1\leq i<j\leq n} \text{Diversity}_t(i,j). \tag{5.7}$$

Let $\mathcal{C}_{\text{chosen}} = \{C_{n_1}, C_{n_2}\}$ be the set of all chosen chains. The next chain C_{n_3} is chosen by

$$n_3 = \underset{C_i \notin \mathcal{C}_{\text{chosen}}}{\text{argmax}} \text{Diversity}_t(n_1, i) + \text{Diversity}_t(n_2, i). \tag{5.8}$$

Add C_{n_3} to $\mathcal{C}_{\text{chosen}}$, and so on until the n'th chain is chosen. Then n' chains with maximum diversity are chosen. Only these chains are used in the predicting procedure. The procedure of diversity pruning is summarized in Algorithm 7. The time complexity of the pruning procedure is $\Theta(M^2N^2)$.

Algorithm 6 The Online Training Procedure of Evolutionary Regressor Chains

Require: m data streams $\{S_i\}_{i=1}^{m}$, learning rate η, evaporation factor ρ, Q bounds $q_{\min}$ and $q_{\max}$

Initialize $m \times m$ machine learning models $\{g_{i,j}\}_{i,j=1}^{m}$
Initialize n Evolutionary Regressor Chains $\{C_i\}_{i=1}^{n}$
Initialize a $m \times m$ table Q with 1s
loop
  Get data $(X_t^{(1)}, y_t^{(1)}), \ldots, (X_t^{(m)}, y_t^{(m)})$
  for all $C_t = \{F_t, O_t\}$ in C_i **do**
    for $i = 1$ **to** m **do**
      Calculate the prediction by $\hat{y}_t^{(o_i)} = f_t^{(i)}(X_t^{(o_i)})$
      Calculate the loss by $\mathcal{L}(f_t^{(i)}) = ||\hat{y}_t^{(o_i)} - y_t^{(o_i)}||$
    end for
    Generate O'_t using the procedure in Algorithm 5
    for $i = 1$ **to** m **do**
      Calculate the prediction by $\hat{y}_t^{(o'_i)} = {f'}_t^{(i)}(X_t^{(o'_i)})$
      Calculate the loss by $\mathcal{L}({f'}_t^{(i)}) = ||\hat{y}_t^{(o'_i)} - y_t^{(o'_i)}||$
    end for
    Update the order using Equation (5.2)
    for $i = 1$ **to** m **do**
      Update the parameter of $f_t^{(i)}$ using Equation (5.5)
    end for
    Update the Q table using Equation (5.3) and (5.4)
  end for
end loop

Algorithm 7 The procedure of diversity pruning

Require: n Evolutionary Regressor Chains $\{C_i\}_{i=1}^{n}$, data $(X_t^{(1)}, y_t^{(1)}), \ldots, (X_t^{(m)}, y_t^{(m)})$, n'

Ensure: $C_{n_1}, C_{n_2}, \ldots, C_{n'_n}$

Initialize an array p of size m with 1s
for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    Calculate $\text{Diversity}_t(i, j)$ by Equation (5.6)
  end for
end for
Choose C_{n_1} and C_{n_2} by Equation (5.7)
for $i = 3$ **to** n' **do**
  Choose C_{n_i} by Equation (5.8)
end for

5.4 Theoretical Analysis

In this section we give some theoretical analysis of our heuristic order searching strategy.

5.4.1 Metrics

Our heuristic order searching strategy is performed in an online fashion. The most commonly used metric in the area of online learning is regret Hazan (2016). Given a machine learning model f_t at time t, the loss of f_t is $\mathcal{L}_t(f_t)$. Regret is defined as

$$\text{Regret}_T = \sum_{t=1}^{T} \mathcal{L}_t(f_t) - \sum_{t=1}^{T} \mathcal{L}_t(f^*),$$

where $f^* = \text{argmin}_{f \in \mathcal{F}} \sum_{t=1}^{T} \mathcal{L}_t(f)$, and $\mathcal{F}$ is the set of all feasible models f. As described in previous sections, the challenge of data stream mining is its dynamicity. This means the optimal f^* change over time. The fixed optimal f^* is not suitable in this situation. A new metric, dynamic regret, is introduced in Zinkevich (2003) to solve this problem. Dynamic regret is defined as

$$\text{Regret}_T^d = \sum_{t=1}^{T} \mathcal{L}_t(f_t) - \sum_{t=1}^{T} \mathcal{L}_t(f'_t),$$

where f'_t is any feasible model. Most research focuses on the worst-case dynamic regret

$$\text{Regret}_T^d = \sum_{t=1}^{T} \mathcal{L}_t(f_t) - \sum_{t=1}^{T} \mathcal{L}_t(f_t^*),$$

where $f_t^* = \text{argmin}_{f_t \in \mathcal{F}_t} \mathcal{L}_t(f_t)$, and $\mathcal{F}_t$ is the set of all feasible models f_t at time t. It is well-known that it is impossible to obtain a bound, unless in terms of some regularities, such as path-length Zhang (2020). The path-length introduced in Mokhtari et al. (2016), is defined as the cumulative variance of the parameters ω_t^* of the optimal model f_t^*,

$$P_T^* = \sum_{t=2}^{T} ||\omega_t^* - \omega_{t-1}^*||^2.$$

5.4.2 Dynamic Regret Analysis

Let $C_t = \{F_t, O_t\}$ be an Evolutionary Regressor Chain at time t, where $F_t = \{f_t^{(i)}\}_{i=1}^m$ and $f_t^{(i)}$ is the ith model in the chain. Since order O_t is optimized heuristically and generated probabilistically, we consider the expected dynamic regret of our model.

$$\text{Regret}_T^d = \mathbf{E}\left(\sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{(i)})\right) - \sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{**(i)}).$$

where $f_t^{*(i)}$ is the optimal ith model in the chain at time t. We would like to distinguish two types of optimal model. Let $C_t^* = \{F_t^*, O_t^*\}$ be the Evolutionary Regressor Chain with optimal order O_t^* at time t, However $f_t^{*(i)} \in F_t^*$ is updated online, so it would not be optimal for $f_t^{*(i)}$. Hence we denote the optimal model as $f_t^{**(i)}$.

The following Lemma by Zhou Zhou (2009) give the upper bound of the expected runtime of the heuristic order searching strategy.

Lemma 1. *The expected runtime of the heuristic order searching strategy satisfy,*

$$\sum_{\tau=1}^{T}\tau P(t_c = \tau) \leq m^6 + \frac{1}{\rho}m\ln m.$$

The following Lemma is by Mokhtari et al. Mokhtari et al. (2016).

Lemma 2. *Let the domain set of parameters $w_t^{(i)}$ be convex. The update rule in Equation (5.5) is correspondingly adjusted to*

$$\omega_{t+1}^{(i)} = \Pi(\omega_t^{(i)} - \eta\nabla\mathcal{L}_t),$$

where Π is the projection onto the nearest point in the convex set. If the loss function $\mathcal{L}_t$ is α-strongly convex, i.e.

$$\mathcal{L}_t(x) \geq \mathcal{L}_t(y) + \nabla\mathcal{L}_t(y)(x-y) + \frac{\alpha}{2}||x-y||^2,$$

for all x, y, the gradient $\nabla\mathcal{L}_t$ is L-Lipschitz continuous, i.e.

$$||\nabla\mathcal{L}_t(x) - \nabla\mathcal{L}_t(y)|| \leq L||x - y||,$$

for all x, y, and $\eta < 1/L$, there exist constants K_1 and K_2, such that

$$\left(\sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{*(i)}) - \sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{**(i)})\right)$$

$$\leq K_1\sum_{i=1}^{m}P_T^{*(i)} + K_2,$$

where $P_T^{*(i)} = \sum_{t=2}^{T}||\omega_t^{**(i)} - \omega_{t-1}^{**(i)}||^2$ is the ith path-length.

The dynamic regret bound of proposed heuristic order searching strategy is given in the following theorem.

Theorem 1. *If all conditions in Lemma 2 are satisfied, and the variance of models on the same data stream are bounded by*

$$\mathcal{L}_t(g_{i_1,j}) - \mathcal{L}_t(g_{i_2,j}) \leq G,$$

for all $1 \leq i_1 < i_2 \leq m$, we have the upper bound of the dynamic regret,

$$Regret_T^d = O(\sum_{i=1}^{m}P_T^{*(i)}),$$

where $P_T^{*(i)} = \sum_{t=2}^{T}||\omega_t^{**(i)} - \omega_{t-1}^{**(i)}||^2$ is the ith path-length.

Proof. We first partition the expected dynamic regret into two parts,

$$\begin{aligned}\text{Regret}_T^d &= \mathbf{E}\left(\sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{(i)})\right) - \sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{**(i)}) \\ &= \left(\mathbf{E}\left(\sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{(i)})\right) - \sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{*(i)})\right) \qquad (5.9) \\ &\quad + \left(\sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{*(i)}) - \sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{**(i)})\right), \qquad (5.10)\end{aligned}$$

where $f_t^{*(i)}$ and $f_t^{**(i)}$ are two types of optimal model we have described above. Let t_c be the time when the heuristic order searching converges. Then term (9) can be expanded as,

$$\begin{aligned}
&\mathbf{E}\left(\sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{(i)})\right) - \sum_{t=1}^{T}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{*(i)}) \\
&= \sum_{\tau=1}^{T} P(t_c=\tau)\left(\sum_{t=1}^{\tau}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{(i)}) - \sum_{t=1}^{\tau}\sum_{i=1}^{m}\mathcal{L}_t(f_t^{*(i)})\right) \\
&= \sum_{\tau=1}^{T} P(t_c=\tau)\left(\sum_{t=1}^{\tau}\sum_{i=1}^{m}(\mathcal{L}_t(f_t^{(i)}) - \mathcal{L}_t(f_t^{*(i)}))\right) \\
&\leq mG\sum_{\tau=1}^{T}\tau P(t_c=\tau)
\end{aligned}$$

According to Lemma 1, we have

$$\text{term (9)} \leq m^2G(m^5 + \frac{1}{\rho}\ln m). \tag{5.11}$$

In the term (10), the order of the chain is consistent. Let the ith path-length $P_T^{*(i)} = \sum_{t=2}^{T}||\omega_t^{**(i)} - \omega_{t-1}^{**(i)}||^2$. According to Lemma 2, there exist constants K_1 and K_2, such that

$$\begin{aligned}
\text{term (10)} &= \sum_{i=1}^{m}\sum_{t=1}^{T}\left(\mathcal{L}_t(f_t^{*(i)}) - \mathcal{L}_t(f_t^{**(i)})\right), \\
&\leq K_1\sum_{i=1}^{m}P_T^{*(i)} + K_2
\end{aligned} \tag{5.12}$$

Combining Equation (5.11) and Equation (5.12), we have

$$\text{Regret}_T^d \leq m^2G(m^5 + \frac{1}{\rho}\ln m) + K_1\sum_{i=1}^{m}P_T^{*(i)} + K_2.$$

The m, ρ, G, K_1 and K_2 are all constants, which implies that $\text{Regret}_T^d = O(\sum_{i=1}^{m}P_T^{*(i)})$. This completes the proof of Theorem 1. □

Remark 3. *Convexity is important in optimization theory. Here we assume the domain set of parameters is convex, while there is no restriction in the algorithm in Figure 6. The assumption still holds if we select a large enough hyper-sphere in the parameter space as the domain set.*

Remark 4. *The assumption that, the variance of models on the same data stream are bounded by G for all $1 \leq i_1 < i_2 \leq m$, is reasonable. The only difference between $g_{i_1,j}$ and $g_{i_1,j}$ is that, the former takes $y_t^{(i_1)}$ as an additional feature, while the latter takes $y_t^{(i_2)}$ as an additional feature. The variance between them should not be too large.*

Remark 5. *The analysis of term (11) is based on the runtime analysis of the Ant Colony System in Gutjahr and Sebastiani (2008); Zhou (2009). Correlation drift is not taken into account, because if correlation drift occurs the convergence of the order search procedure cannot be guaranteed. The analysis with correlation drift will be our future work.*

5.5 Experimental Study

As our point of departure, we conduct parameter studies experiment of the evaporation factor and the number of chains respectively. Next, we assessed the efficiency of the heuristic order searching strategy. Then we conduct an experiment to verify the efficiency of the diversity pruning technique and determine how many chains should be left after pruning. Finally, we evaluate our method on three real-world multi-stream datasets.

5.5.1 Data Sets

We use three real-world, multi-stream datasets to evaluate the performance of our proposed method Evolutionary Regressor Chains. Current research only focus on correlations between two data streams. There are no datasets include more than two data streams. To evaluate our method, we collect raw multi-stream data from different applications and compose four datasets. The descriptions of the three multi-stream datasets are as follows:

Table 5.1 : Real-world datasets

Dataset	#sample	#feature	#streams
Train	62682	12	8
Weather	49728	8	10
Sensor	20844	3	6

Train The Train dataset Yu et al. (2020) records train operating data from Transport for NSW. All data are divided into eight streams according to the carriage number. The task is to predict the change of load of each carriage during the dwell time.

Weather The Weather dataset Wang et al. (2019) records weather data from ten weather stations around Beijing. Data from each station constitute a data stream. The task is to predict the wind at 10 meters.

Sensor Sensor dataset* records data from 54 sensors deployed in the Intel Berkeley Research Lab from February 28th to April 5th, 2004. We selected 6 sensors with few missing data as 6 data streams. The feature include the temperature, humidity and light. The task is to predict the voltage of the sensor.

The detail of the real-world datasets is summarized in Table 5.1.

5.5.2 Experiment Setting

In all our experiments, a linear regression model is chosen as the base learner because of its simple structure and the ease of training. For datasets with category

*http://db.csail.mit.edu/labdata/labdata.html

features, such as station names in the Train dataset, an embedding layer is utilized to transfer the category feature to a continuous feature in an embedding space.

An $L2$ norm of parameters is added to the loss as a regularity, to prevent overfitting while conducting gradient descent. That is the regularized loss,

$$\mathcal{RL} = \mathcal{L} + \lambda||\omega||^2,$$

where λ is the factor to control the effect of regularity. It is important to be careful to select a proper learning rate η and a proper factor λ. If the learning rate η is large, or the factor λ is small, the trainin pgrocess is unstable and leads to exploding gradients. On the other hand, if η is small, or λ is large, it can lead to low converge rate and influence the performance of the machine learning models. In all our experiments, the choices of η and λ are based on the experiment results.

All our experiments were conducted on an RHEL server with a 24-core Intel Xeon Gold 6126 CPU, 187GB of memory, and two NVIDIA Tesla V100 GPUs. Our method is implemented in PyTorch. The NVIDIA driver version is 440.59, the CUDA version is 10.2 and the PyTorch version is 1.8.1.

5.5.3 Parameter Study of the Number of Chains

In this section, we give a parameter study of the number of chains. To find a proper chain order as soon as possible, it is reasonable to generate multiple chains to search together. It is also the custom in heuristic optimization scenario.

The problem now is how many chains should be generated. We generated different numbers of Evolutionary Regressor Chains. The results are shown in Figure 5.2.

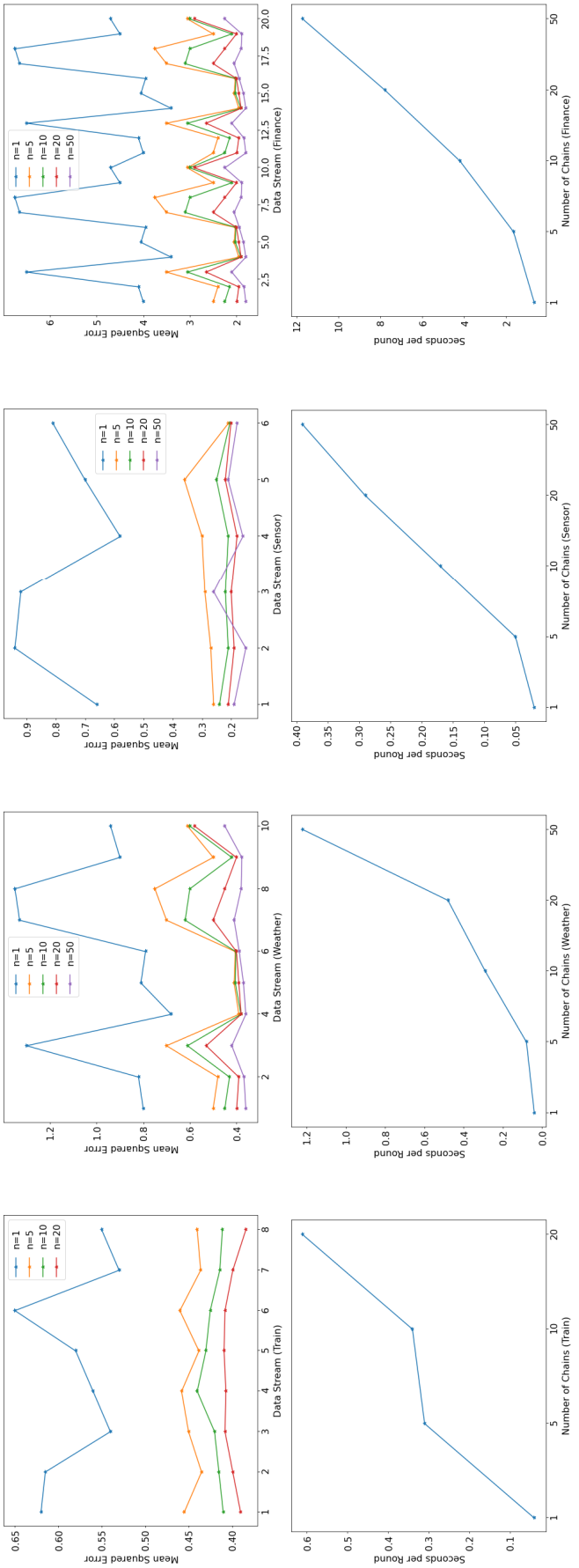


Figure 5.2 : Performance of models with different numbers of chains on five real-world datasets. The first subfigure show the mean squared error with different number of chains. The second subfigure show the average running time to process one sample.

As shown in the figure, the more chains are generated, the less error they yield. However, as the number of chains increases, the time complexity increases as well. It is a trade-off between accuracy and running time. According to Figure 5.2, for more than 10 chains, the improvement of performance is little, while the extra running time is large. The gains do not make up for the losses. In the following experiments, we only generate 10 chains.

5.5.4 Evaluation of the Diversity Pruning

In this section, we evaluate the efficiency of the diversity pruning technique. The aim is to inspect whether the diversity pruning technique can really improve the performance via increasing diversity. Another problem is that how many chains should be left after pruning.

We compare the performance of Evolutionary Regressor Chains with 3, 5 and 7 chains left after pruning and without pruning. on three multi-stream datasets. The results are shown in Figure 5.3.

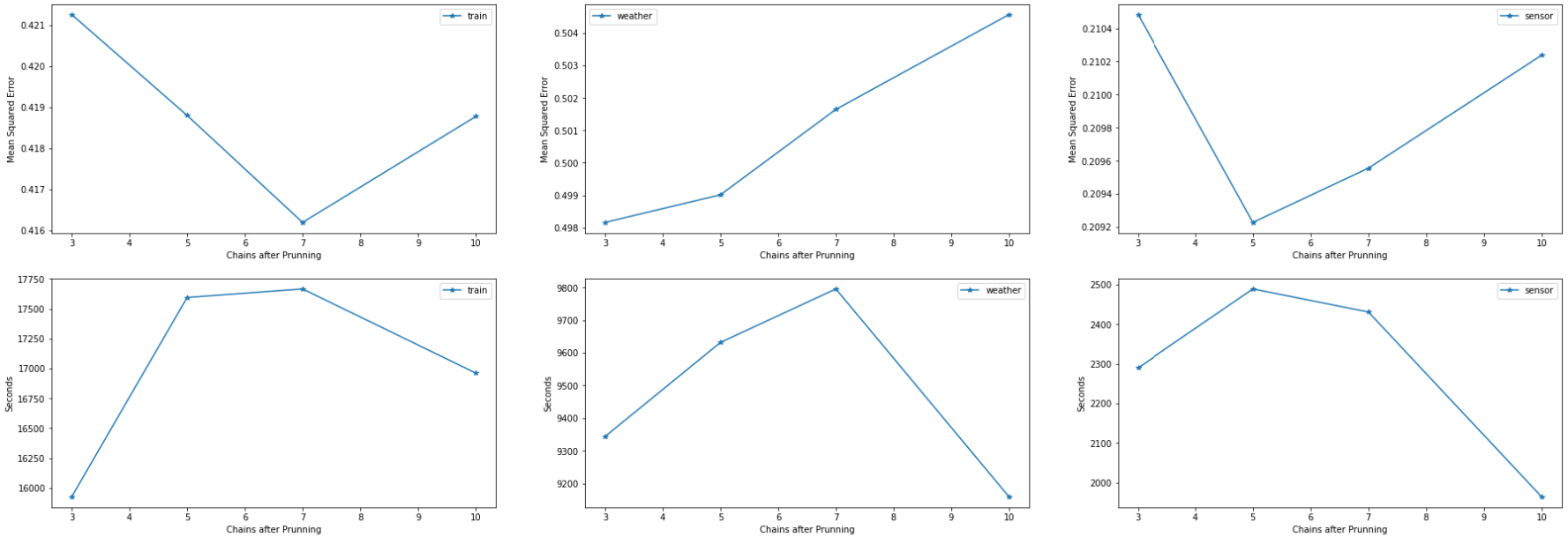


Figure 5.3 : Performance of models with 3/5/7 chains left after pruning and without pruning. The first row show the mean squared error of models with different number of chains left after pruning of three datasets. The second row show the average running time of models with different number of chains left after pruning of three datasets. 10 chains left means no pruning procedure is conducted.

Just like the choice of parameter ρ, it is hard to choose a uniform optimal number of chains left after pruning for all situations. The choice depends on the data. For the Train dataset, we set 7 chains left after pruning; for the Weather dataset, we set 3 chains left after pruning; for the Sensor dataset, we set 5 chains left after pruning.

5.5.5 Evaluation of the Evolutionary Regressor Chains

We evaluate the efficiency of our proposed Evolutionary Regressor Chains on the three multi-stream datasets. The aim is to evaluate whether treating the predictions from other data streams as additional feature can really help improve the performance of models.

Previous research only focus on the correlation between two data stream, which is not suitable in this setting. We extend some multi-target regression method, including Multi-target Regression Stacking (MTRS) Spyromitros-Xioufis et al. (2020), Regressor Chains (RC) and Ensemble Regressor Chains (ERC) Spyromitros-Xioufis et al. (2016), into the data stream setting to track the correlation between data streams. We also use the trivial idea that building a single model for each data stream as baseline (Baseline), in which all models are independent and non-connective. Each model is online updated when new instance arrived. The results are summarized in Table 5.2. Seconds means the average running time to process one data. It measures the time complexity of the method.

As seen in the tables, with information from other correlational data streams, which is learnt via the chain structure, the performance of the models are improved significantly on all three datasets. Hence, our proposed Evolutionary Regressor Chains which can learn the correlation between data streams, provides significant improvement to the performance of machine learning models.

Table 5.2 : Evaluation of Evolutionary Regressor Chains on Three Multi-Stream dataset

Dataset	method	λ	mean squared error	seconds
Train	Baseline	1	502.713	**0.0154**
	MTRS	1	66.364	0.0243
	RC	1	66.274	0.0157
	ERC	1	9.217	0.0430
	Our method	1	**0.414**	0.3567
Weather	Baseline	1	100.126	**0.0070**
	MTRS	1	2.611	0.0174
	RC	1	3.517	0.0073
	ERC	1	1.366	0.0163
	Our method	1	**0.497**	0.1587
Sensor	Baseline	1	12.304	**0.0041**
	MTRS	1	2.181	0.0089
	RC	1	2.349	0.0048
	ERC	1	0.431	0.0055
	Our method	1	**0.051**	0.0525

5.6 Summary

In this chapter, we propose an ensemble chain-structured model, Evolutionary Regressor Chains, with a heuristic order searching strategy and a diversity pruning technique to track the correlation between multiple data streams. The chain structure can track correlation correctly, and information from other data streams can improve the performance of the models. The heuristic order searching strategy is used to search the optimal order of the chain and to update the chains to adapt to correlation drift. Finding the optimal chain order can ensure that the learnt correlation are correct, and ensure improvement in performance. The diversity pruning technique is used to prune some unnecessary chains, which reduces the computation complexity and increases the diversity of the ensemble. The experiment results show the efficiency of our method.

Chapter 6

Learning to Fast Adapt in the Evolving Environment

6.1 Introduction

We have witnessed the popularity of deep learning in a wide variety of areas, including computer vision, natural language processing and speech recognition Yuan et al. (2022). The stacked hierarchical structure of a deep neural network makes it more powerful and flexible in learning complicated representations of knowledge from data. Each coin has two sides. The complicated structure brings about large-scale parameters. Thus, training a deep neural network requires both huge amounts of data and high-performance computation resources.

New challenges are encountered in the scenario of data stream classification brought on by concept drift. Concept drift refers to the phenomenon that the distribution of data changes over time. Current research handling concept drift follow a fix paradigm Lu et al. (2019), which is shown in Figure 4.1. Classifiers in non-stationary environment are supposed to be adaptable to suit dynamic data distribution. In this setting, on the one hand, the parameters will be updated numerous times using only reduced data. It is very easy to be trapped into over-fitting. On the other hand, the adaptation to concept drift has to be quick enough to deal efficiently with newly arrived data. However, vast quantities of iterations before convergence spends plenty of time, which can hardly meet the restricted time requirement. It is necessary to design an adaptation strategy for neural network classifier in the evolving environment to adapt to the concept drift.

Meta learning Schmidhuber (1987) has drawn more attractions recently. The idea is to learn the learning procedure using a meta learner. Though machine learning models can acquire knowledge from historical data automatically, manual intervention is still necessary for hyper-parameter selection, network architecture search, etc. In meta learning, a meta-level model can learn how to train a valid model from an enormous amount of training procedure. For this reason, meta learning is also known as learning to learn. Usually there are two learning procedures in the meta learning setting. In the base learning, a base learner is used to solve a specific task. Then a meta learner learns to improve the performance of the base learner. In particular, meta learning show its advantage in the few-shot problems Hospedales et al. (2022). Usually the parameters of a deep neural network are updated by a gradient descent based method. The optimization using gradient descent based parameter updating rules show poor performance in the few-shot problems due to lack of training data. Several methods that learn a proper parameter updating rule from former training procedure are proposed. One of the most famous methods is Model-Agnostic Meta-Learning (MAML) Finn et al. (2017), which learns proper initialization parameters. The network could converge rapidly with small training set. In Andrychowicz et al. (2016), a meta learner is used to learn the proper update rules to deal with the few-shot problems.

It is obvious that the same problem is encountered in both concept drift adaptation and few-shot problem. The same solution can be applied that using a meta learner to learn how to adapt to concept drift. Therefore, we proposed a concept drift adaptation strategy, Learning to Fast Adapt in the Evolving Environment (LFAEE). Traditional gradient descent based parameter updating rules are in the form

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_\theta \mathcal{L},$$

where α_t is manually setting the learning rate. In our strategy, a LSTM recurrent

neural network m is used for parameter updating. Given the current loss $\mathcal{L}$ and its gradient $\nabla_\theta \mathcal{L}$, the output of m is used as the next updating step. That is, parameters are updated as following,

$$\theta_t = \theta_{t-1} - m(\mathcal{L}, \nabla_\theta \mathcal{L}; \phi).$$

The model can adapt to concept drift rapidly with this updating step.

The main contributions of this work can be summarized as follows:

- We design a concept drift adaptation strategy for neural network classifiers in the evolving environment. A parameter updating rule based on meta-level LSTM recurrent neural network replaces traditional gradient descent based rules while updating the classifier.
- Experiments on several real-world datasets have shown the effectiveness of our strategy. The classifier can react to concept drift rapidly.

The rest of this chapter is organized as follows. In Section 6.2, we give a brief review of previous related work. In Section 6.3, the details of the proposed strategy, Learning to Fast Adapt in the Evolving Environment is given. In Section 6.4, we conducted several experiments to evaluate our strategy. In Section 6.5, we summarized our work.

6.2 Proposed Method

We begin this section by describing the meta-level LSTM recurrent neural network used for generating parameter updating step. This led to the adoption of some advice from Andrychowicz et al. (2016) to simplify our model. Finally, the detail of training procedure of the meta-level neural network is given.

6.2.1 Model Description

Let $S = \{X_1, X_2, \dots, X_t, \dots\}$ be a data stream, where X_t is a batch of data at time point t. Let $f(X;\theta)$ be a neural network classifier for a data stream classification task with parameter θ. Let D be historical data collected from S. A initial neural network classifier, $f(X;\theta_0)$ can be trained from historical data D.

As new data batch X_t arrives, the neural network classifier f makes its prediction $\bar{y}_t = f(X_t;\theta_{t-1})$. We assume that the true label y_t is available after predicting. We choose the negative log likelihood loss to evaluate the prediction $\bar{y}_t$. Let m be a LSTM recurrent neural network with parameter ϕ, which takes the loss $\mathcal{L}_t$ of f on X_t, and its gradient $\nabla_\theta \mathcal{L}_t$ as input and generates a proper updating step $m(\mathcal{L}_t, \nabla_{\theta_t}\mathcal{L}_t;\phi)$. Then the parameter θ is updated as following,

$$\begin{aligned} \bar{y}_t^{(i)} &= f(X_t;\theta_{t-1}^{(i-1)}), \\ \mathcal{L}_t^{(i)} &= loss(\bar{y}_t^{(i)}, y_t), \\ \theta_{t-1}^{(i)} &= \theta_{t-1}^{(i-1)} - m(\mathcal{L}_t^{(i)}, \nabla_\theta \mathcal{L}_t^{(i)};\phi). \end{aligned} \tag{6.1}$$

After a T-time updating, we have the final updated parameter $\theta_t = \theta_{t-1}^{(T)}$. The updated model $f(X;\theta_t)$ is used for predicting on next batch of data X_{t+1}. Given a batch of data X_t, the T-time parameter updating procedure described above is called an episode of our meta learning framework, denoted as E_t. Our meta learning framework is illustrated in Figure 6.1.

6.2.2 Coordinate-wise Parameter Sharing

As we discussed in Section 6.1, the neural network classifier $f(X;\theta)$ usually consists of large-scale parameters. To generate updating steps for all coordinates of parameter θ, vast quantities of parameters are needed in the LSTM recurrent neural network m. To avoid this issue, Andrychowicz et al. (2016) suggested a coordinate-wise parameter sharing strategy. Each coordinate of θ shares the same

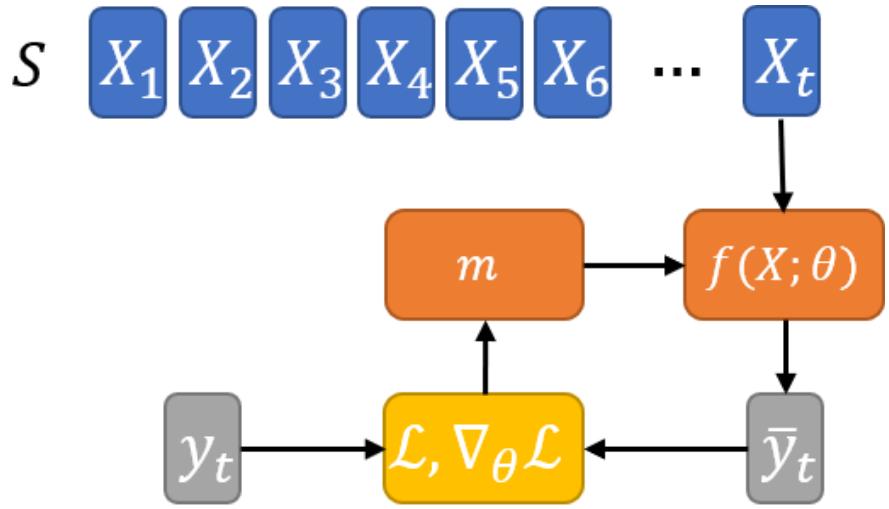


Figure 6.1 : The framework of our strategy

parameters in m, including weights and bias, but not the hidden states and cell states. As a consequence, we have a compact and easy-to-train meta-level LSTM recurrent neural network. Meanwhile, different hidden states and cell states mean that updating for each coordinate only depends on its own historical information respectively. Figure 6.2 demonstrates the mechanism of coordinate-wise parameter sharing.

6.2.3 Preprocessing

Different coordinates of gradient $\nabla_\theta \mathcal{L}$ are of various scale. Data normalization is an important preprocessing technique in traditional feature engineering. However, widely used normalization methods, such as mean removal and variance scaling, can destroy the structure in the gradient. We adopted the robust and powerful preprocessing technique proposed in Andrychowicz et al. (2016). Each coordinate of gradient $\nabla_\theta \mathcal{L}$ will be transferred by the map g,

$$g(x) = \begin{cases} \left(\frac{\log(|x|)}{p}, sign(x)\right) & \text{if} \quad |x| \geq e^{-p} \\ (-1, e^p x) & \text{otherwise} \end{cases}$$

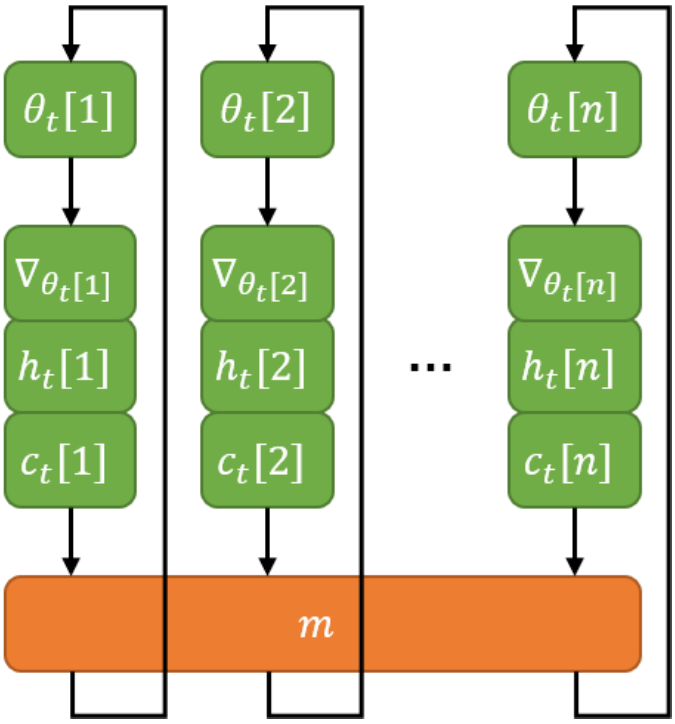


Figure 6.2 : The mechanism of coordinatewise parameter sharing

6.2.4 Training Procedure

Our first step was to collect several data streams, $S_1, S_2, \ldots, S_k$, and their historical data sets, $D_1, D_2, \ldots, D_k$. We hope to learn the proper parameter updating rule from training procedures on these data streams. An initial neural network classifier f_j is built on D_j. For each batch of data $X_{j,t}$ in S_j, we have an episode $E_{j,t}$ described in Section 6.2.1.

We then define the loss of the episode $E_{j,t}$. The most intuitive criterion in assessing the effect of the parameter updating is to evaluate the performance of the updated model on the next batch of data. That is,

$$\mathbf{L}(E_{j,t}) = loss(f_j(X_{t+1}; \theta_t), y_{t+1}).$$

Note that θ_t can be considered as a function of parameter ϕ. To make the expression

clearer, we write θ_t in the form of

$$\theta_t = h(\theta_{t-1}, \phi).$$

The loss of episode $E_{j,t}$ can be rewritten in the form of

$$\mathbf{L}(E_{j,t}) = loss(f_j(X_{t+1}; h(\theta_{t-1}, \phi)), y_{t+1}). \tag{6.2}$$

We noticed that $\theta_t = h(\theta_{t-1}, \phi)$ is of recursive form. For the sake of training simplification, we adopt the advice in Ravi and Larochelle (2017) and ignore the derivative of θ_{t-1} when calculating the gradient during training. Figure 6.3 show the computational graph for calculating the loss of an episode. The dashed arrow in Figure 6.3 means that the derivative will be ignored when calculating the gradient.

We have the total loss

$$\mathbf{L} = \sum_{j=1}^{k} \sum_{t=1}^{n_j} loss(f_j(X_{t+1}; h(\theta_{t-1}, \phi)), y_{t+1}),$$

and the parameter of m is acquired as

$$\phi = \arg\min_{\phi} \sum_{j=1}^{k} \sum_{t=1}^{n_j} loss(f_j(X_{t+1}; h(\theta_{t-1}, \phi)), y_{t+1}).$$

In practice, the parameter ϕ is updated episode by episode by Equation (2) using gradient descent based methods such as SGD.

The training procedure of Learning to Fine-Tuning in the Evolving Environment is summarized in Algorithm 8.

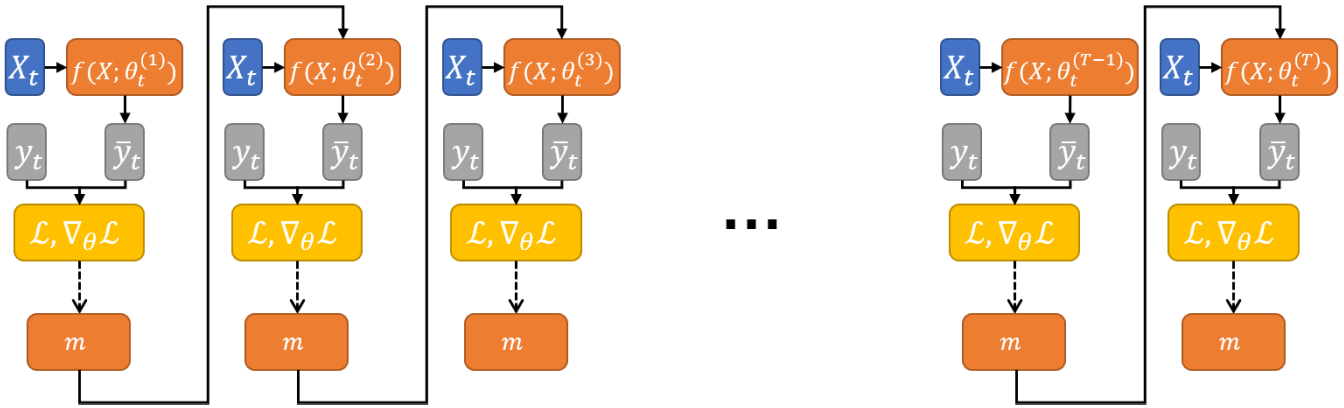


Figure 6.3 : The computational gragh for calculation the loss of a episode

Algorithm 8 Learning to Fine-Tuning in the Evolving Environment

Input: Data streams $S_1, S_2, \ldots, S_k$,

Historical data sets $D_1, D_2, \ldots, D_k$,

Parameter: training epoch n, updating step T

Output: ϕ

Initialize ϕ
for $e = 1$ to n **do**
  for $j = 1$ to k **do**
    Train $f(X; \theta_0)$ on D_j
    for all $X_{j,t}$ in S_j **do**
      for $i = 1$ to T **do**
        Update θ_i by Equation (1)
      end for
      Calculate the loss $\mathbf{L}$ by Equation (2)
      Update ϕ to minimize $\mathbf{L}$
    end for
  end for
end for
return ϕ

6.3 Experimental Study

We conducted several experiments to evaluate our proposed method. We first evaluate our method on four real-world datasets. Then we evaluate our method with meta learner trained on other data streams.

6.3.1 Datasets

We use four real-world datasets to evaluate our method. In this section, we give a brief review of the datasets *.

- **Airlines** The task is to predict whether the flight will delay.
- **Covtype** The dataset is collected from US Forest Service Region 2 Resource Information System. The task is to predict the type of forest cover.
- **Kddcup99** The task is to detect the intrusion and predict the type.
- **Pokerhand** The dataset record hands drawn from a standard deck of 52. The task is to predict the type of the poker hand.

The statistics of four datasets are shown in Table 6.1. The starting 10% data of the streams are used as training set and remain 90% data are used for evaluation.

6.3.2 Experiment Settings

In this work, a two-layer LSTM RNN is performed as our meta learner. The length of LSTM is set as 10. The parameters of the meta learner is estimated by a stochastic gradient descent with momentum. The learning rate is set as 0.001 and the momentum is set as 0.9. The epoch is set as 10.

*The datasets Airlines, Covtype and Pokerhand are available at https://moa.cms.waikato.ac.nz/datasets/. The dataset Kddcup99 is available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Table 6.1 : Real-world Data Stream to Evaluate LFAEE

Data Stream	#Example	#Feature	#Label
Airlines	539383	7	2
Covtype	581012	54	7
Kddcup99	494021	41	23
Pokerhand	1000000	10	10

The base learner is a two-layer full connected network. The base learner is updated following the output of the meta learner.

All experiments were conducted on a computing cluster with an INTEL Xeon Gold 6126 CPU @ 2.60GHz and 192G memory, and two NVIDIA Tesla V100 GPUs. and the operating system is Red Hat Enterprise Linux[†]. Our method is implemented in PyTorch. The NVIDIA driver version is 440.59, the CUDA version is 10.2 and the PyTorch version is 1.8.1.

6.3.3 Evaluation of LFAEE

In this section, we evaluate our method, LFAEE, on four real-world datsets. We choose stochastic gradient descent with momentum as baseline. First we use only one stream to train the meta learner. The starting 10% data of the stream are used to train both the base learner and the meta learner. The remaining 90% data are used for evaluation. The experimental results are listed in Table 6.2. As shown in the table, our method overperform than baseline on all four datasets. That is the updating step learned by the meta learner is better than the updating step calculated by gradient. Therefore the efficiency of LFAEE is validated.

[†]https://www.redhat.com

Table 6.2 : Evaluation of LFAEE on Four Real-world Datasets

Data Stream	Baseline	LFAEE
Airlines	0.6081	**0.6287**
Covtype	0.7841	**0.8221**
Kddcup99	0.6728	**0.7278**
Pokerhand	0.7226	**0.7410**

Table 6.3 : Evaluation of LFAEE with Cross-stream Meta Learners

Data Stream	t.o. Airlines	t.o. Covtype	t.o. Kddcup99	t.o. Pokerhand
Airlines	-	0.6284	0.6287	0.6293
Covtype	0.8186	-	0.8203	0.8201
Kddcup99	0.7296	0.7085	-	0.7282
Pokerhand	0.7393	0.7382	0.7431	-

6.3.4 Evaluation of LFAEE with Cross-stream Meta Learners

In this section, we evaluate our method where the meta learner is trained from other data streams. We would like to investigate whether the meta learner can learn cross-streams knowledge and hence the meta learner trained on one data stream can be used for other data streams. The meta learner trained on dataset Airlines is denoted as **t.o. Airlines**, for example. The experiment results are shown in Table 6.3. The results show than there are no significant differences if using meta learner trained on other data streams. The tasks of the four datasets are totally different. We can conclude that the meta learner can learn cross-streams knowledge and be used for other data streams.

6.4 Summary

In this work, we have proposed Learning to Fast Adapt in the Evolving Environment, a concept drift adaptation strategy for neural network classifiers in non-stationary environments. A meta-level LSTM recurrent neural network is used to generate proper updating step for parameters of the neural network classifier according to current loss and its gradient to react to the concept drift. The classifier can be updated in limited iteration times to fulfill the limited time requirement in the data stream classification scenario. In our experiments, our strategy was evaluated on both synthetic and real-world data sets with concept drift. It show that classifiers with our updating strategy are more robust in an environment with concept drift.

Chapter 7

Conclusions and Future Study

This chapter concludes the thesis. In addition, recommendation for future study is given.

7.1 Conclusions

Many methods have been developed to handle concept drift in data streams. However, these methods consider data streams separately, ignoring the correlation between data streams. This research concentrates on answering the following four questions: 1) How to model the correlation between two data streams; 2) How to model the correlation between multiple data streams; 3) How to track changes to the correlation between multiple data streams; 4) How to adapt a machine learning model according to the correlation between multiple data streams. This research conducted a comprehensive analysis of these questions and proposed several methods.

The main contributions of this research are as follows:

1. It proposes a novel concept drift adaptation method which can overcome the insufficient training problem caused by scarce newly arrived data. We train the classifier on a latent feature space using knowledge learnt from historical data to make predictions on the newly arrived data.

2. It proposes a novel multi-stream Concept drift Handling Framework, which considers the correlation between multiple data streams rather than handling

data streams separately. The advantage of the framework is that the parameters of the membership functions are estimated using data from other streams. These data which have different distributions help to reach higher concept drift detection accuracy. A new drift detection method, FMDD, is designed to detect when and in which streams concept drift occurs, dividing streams into drifting and non-drifting streams at each time. The parameters of the fuzzy membership functions are estimated using data from other data streams, which lead to a remarkably high true positive rate. A new drift adaptation method, FMDA, is proposed using the correlation of multiple data streams to train a new model after concept drift is detected. By increasing the volume of training data, the over-fitting issue due to a lack of data is alleviated.

3. It proposes a chain-structured model, Evolutionary Regressor Chains, to track the correlation among multiple data streams. We firstly take the correlation between more than two data streams into account to improve the performance of the models in data stream regression, which has not been solved by existing research. To overcome the drawback that the randomly generated chain order cannot track the correlation correctly, we design a heuristic order searching strategy. The method updates the order iteratively to find an optimal order. We also design an online updating strategy to update the models in the multi-stream regression scenario. This strategy can effectively adapt to both concept drift and correlation drift. We propose a diversity pruning method to decrease the complexity of our method while maximizing the diversity of the ensemble. It can also increase the robustness of our method. We analyze some theoretical properties of our method and give its dynamic regret bound.

4. It proposes a concept drift adaptation strategy for neural network classifiers in the evolving environment. A parameter updating rule based on meta-level

LSTM recurrent neural network replaces traditional gradient descent-based rules while rapidly adapting the classifier.

7.2 Future Study

Although several methods are proposed in this research, the questions on which this research focuses are still challenging and further investigation is needed. This thesis identifies the following as future study:

- In this research, we propose a novel concept drift adaptation method, Drift Adaptation via Joint Distribution Alignment (DAJDA). The main drawback of DAJDA is that a generalized eigendecomposition problem needs to be solved, so the time complexity might be costly for an online learning case. In addition, DAJDA is only able to perform linear transformation on the instances. Some non-linearity is needed. This is a problem remaining to be solved in the future.

- In this research, we propose a Concept Drift Handling Framework. However, in our framework, all data streams share the same feature space. Therefore, data from other streams can be directly added to the training set. In many real-world applications, the feature space of multiple streams is different. How to use data from other data streams when the feature space is different will be our next target.

- In this research, we propose an ensemble of Evolutionary Regressor Chains. Our method is limited by the chain structure. The chain structure is simple and easy to implement. However, the correlation in the real world are not always linear. Replacing the chain structure with a more complex structure, like trees or graphs, will be our next direction for future research.

Bibliography

Ahmadi, Z. & Kramer, S., 2018, 'Modeling recurring concepts in data streams: a graph-based framework', *Knowledge and Information Systems*, vol. 55, no. 1, pp. 15–44.

Alippi, C., Boracchi, G. & Roveri, M., 2017, 'Hierarchical Change-Detection Tests', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 2, pp. 246–258.

Alippi, C. & Roveri, M., 2008, 'Just-in-Time Adaptive Classifiers—Part I: Detecting Nonstationary Changes', *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1145–1153.

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B. & de Freitas, N., 2016, 'Learning to learn by gradient descent by gradient descent', Lee, D., Sugiyama, M., Luxburg, U., Guyon, I. & Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, , vol. 29Curran Associates, Inc., pp. 3988–3996.

Antoniou, A., Edwards, H. & Storkey, A., 2019, 'How to train your MAML', *International Conference on Learning Representations*, .

Ashfahani, A. & Pratama, M., 2019, 'Autonomous Deep Learning: Continual Learning Approach for Dynamic Environments', *Proceedings of the 2019 SIAM International Conference on Data Mining*, Society for Industrial and Applied Mathematics, pp. 666–674.

Ashfahani, A., Pratama, M., Lughofer, E. & Ong, Y.-S., 2020, 'DEVDAN: Deep evolving denoising autoencoder', *Neurocomputing*, vol. 390, pp. 297–314.

Bach, S. H. & Maloof, M. A., 2008, 'Paired Learners for Concept Drift', *2008 Eighth IEEE International Conference on Data Mining*, pp. 23–32.

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

Basseville, M. & Nikiforov, I., 1993, *Detection of Abrupt Change Theory and Application*, vol. 15, Prentice Hall.

Behbood, V., Lu, J. & Zhang, G., 2011, 'Long term bank failure prediction using Fuzzy Refinement-based Transductive Transfer learning', *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, pp. 2676–2683.

Behbood, V., Lu, J. & Zhang, G., 2013, 'Fuzzy Bridged Refinement Domain Adaptation: Long-term Bank Failure Prediction', *International Journal of Computational Intelligence and Applications*, vol. 12, no. 01, p. 1350003.

Bernardo, A. & Della Valle, E., 2021, 'VFC-SMOTE: very fast continuous synthetic minority oversampling for evolving data streams', *Data Mining and Knowledge Discovery*, vol. 35, no. 6, pp. 2679–2713.

Bezdek, J. C., 2013, *Pattern recognition with fuzzy objective function algorithms*, Springer Science & Business Media.

Bifet, A. & Gavaldà, R., 2007, 'Learning from Time-Changing Data with Adaptive Windowing', *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*, SIAM, pp. 443–448.

Bifet, A. & Gavaldà, R., 2009, 'Adaptive Learning from Evolving Data Streams', Adams, N. M., Robardet, C., Siebes, A. & Boulicaut, J.-F. (eds.) *Advances in Intelligent Data Analysis VIII*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 249–260.

Bifet, A., Holmes, G. & Pfahringer, B., 2010a, 'Leveraging Bagging for Evolving Data Streams', Balcázar, J. L., Bonchi, F., Gionis, A. & Sebag, M. (eds.) *Machine Learning and Knowledge Discovery in Databases*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 135–150.

Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T. & Seidl, T., 2010b, 'MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering', Diethe, T., Cristianini, N. & Shawe-Taylor, J. (eds.) *Proceedings of the First Workshop on Applications of Pattern Analysis*, , vol. 11 of *Proceedings of Machine Learning Research*PMLR, Cumberland Lodge, Windsor, UK, pp. 44–50.

Bifet, A., Pfahringer, B., Read, J. & Holmes, G., 2013, 'Efficient Data Stream Classification via Probabilistic Adaptive Windows', *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, Association for Computing Machinery, New York, NY, USA, pp. 801–806, event-place: Coimbra, Portugal.

Blackard, J. A., 1998, *Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types*, PhD Thesis, Colorado State University, USA, iSBN: 059921242X.

Bonilla, E. V., Chai, K. & Williams, C., 2007, 'Multi-task Gaussian Process Prediction', Platt, J., Koller, D., Singer, Y. & Roweis, S. (eds.) *Advances in Neural Information Processing Systems*, , vol. 20Curran Associates, Inc., pp. 153–160.

Brzezinski, D. & Stefanowski, J., 2014, 'Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94.

Bu, L., Alippi, C. & Zhao, D., 2018, 'A pdf-Free Change Detection Test Based on Density Difference Estimation', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 2, pp. 324–334.

Budiman, A., Fanany, M. I. & Basaruddin, C., 2016, 'Adaptive Convolutional ELM For Concept Drift Handling in Online Stream Data', <https://arxiv.org/abs/1610.02348>.

Cano, A. & Krawczyk, B., 2019, 'Evolving rule-based classifiers with genetic programming on GPUs for drifting data streams', *Pattern Recognition*, vol. 87, pp. 248–268.

Cano, A. & Krawczyk, B., 2020, 'Kappa Updated Ensemble for drifting data stream mining', *Machine Learning*, vol. 109, no. 1, pp. 175–218.

Cano, A. & Krawczyk, B., 2022, 'ROSE: robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams', *Machine Learning*, vol. 111, no. 7, pp. 2561–2599.

Cerqueira, V., Gomes, H. M., Bifet, A. & Torgo, L., 2022, 'STUDD: a student–teacher method for unsupervised concept drift detection', *Machine Learning*.

Chandra, S., Haque, A., Khan, L. & Aggarwal, C., 2016, 'An Adaptive Framework for Multistream Classification', *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, Association for Computing Machinery, New York, NY, USA, pp. 1181–1190, event-place: Indianapolis, Indiana, USA.

Chiu, C. W. & Minku, L. L., 2022, 'A Diversity Framework for Dealing With Multiple Types of Concept Drift Based on Clustering in the Model Space', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 3, pp. 1299–1309.

Chu, F. & Zaniolo, C., 2004, 'Fast and Light Boosting for Adaptive Mining of Data Streams', Dai, H., Srikant, R. & Zhang, C. (eds.) *Advances in Knowledge Discovery and Data Mining*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 282–292.

Dai, W., Xue, G.-R., Yang, Q. & Yu, Y., 2007a, 'Transferring Naive Bayes Classifiers for Text Classification', *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, AAAI Press, pp. 540–545, event-place: Vancouver, British Columbia, Canada.

Dai, W., Yang, Q., Xue, G.-R. & Yu, Y., 2007b, 'Boosting for Transfer Learning', *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, Association for Computing Machinery, New York, NY, USA, pp. 193–200, event-place: Corvalis, Oregon, USA.

Dasu, T., Krishnan, S., Venkatasubramanian, S. & Yi, K., 2006, 'An Information-Theoretic Approach to Detecting Changes in MultiDimensional Data Streams', *Interfaces of Statistics, Computing Science and Applications*.

Davis, J. & Domingos, P., 2009, 'Deep Transfer via Second-Order Markov Logic', *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, Association for Computing Machinery, New York, NY, USA, pp. 217–224, event-place: Montreal, Quebec, Canada.

De Smedt, J., Deeva, G. & De Weerdt, J., 2020, 'Mining Behavioral Sequence

Constraints for Classification', *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1130–1142.

Dembczyński, K., Cheng, W. & Hüllermeier, E., 2010, 'Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains', *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, Omnipress, Madison, WI, USA, pp. 279–286, event-place: Haifa, Israel.

Deng, Y., Ren, Z., Kong, Y., Bao, F. & Dai, Q., 2017, 'A Hierarchical Fused Fuzzy Deep Neural Network for Data Classification', *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 4, pp. 1006–1012.

Disabato, S. & Roveri, M., 2019, 'Learning Convolutional Neural Networks in presence of Concept Drift', *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.

Ditzler, G. & Polikar, R., 2011, 'Hellinger distance based drift detection for nonstationary environments', *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, pp. 41–48.

Domingos, P. & Hulten, G., 2000, 'Mining High-Speed Data Streams', *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, Association for Computing Machinery, New York, NY, USA, pp. 71–80, event-place: Boston, Massachusetts, USA.

Dong, F., Lu, J., Song, Y., Liu, F. & Zhang, G., 2022, 'A Drift Region-Based Data Sample Filtering Method', *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9377–9390.

Dorigo, M., Maniezzo, V. & Colorni, A., 1996, 'Ant system: optimization by a

colony of cooperating agents', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41.

Dries, A. & Rückert, U., 2009, 'Adaptive concept drift detection', *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 2, no. 5-6, pp. 311–327, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sam.10054.

Elwell, R. & Polikar, R., 2011, 'Incremental Learning of Concept Drift in Nonstationary Environments', *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531.

Evgeniou, T. & Pontil, M., 2004, 'Regularized Multi–Task Learning', *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, Association for Computing Machinery, New York, NY, USA, pp. 109–117, event-place: Seattle, WA, USA.

Fekri, M. N., Patel, H., Grolinger, K. & Sharma, V., 2021, 'Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network', *Applied Energy*, vol. 282, p. 116177.

Finn, C., Abbeel, P. & Levine, S., 2017, 'Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks', Precup, D. & Teh, Y. W. (eds.) *Proceedings of the 34th International Conference on Machine Learning*, , vol. 70 of *Proceedings of Machine Learning Research*PMLR, pp. 1126–1135.

Finn, C. & Levine, S., 2018, 'Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm', *International Conference on Learning Representations*, .

Frías-Blanco, I., Campo-Ávila, J. d., Ramos-Jiménez, G., Morales-Bueno, R., Ortiz-Díaz, A. & Caballero-Mota, Y., 2015, 'Online and Non-Parametric Drift

Detection Methods Based on Hoeffding's Bounds', *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823.

Gama, J. & Castillo, G., 2006, 'Learning with Local Drift Detection', Li, X., Zaïane, O. R. & Li, Z. (eds.) *Advanced Data Mining and Applications*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 42–55.

Gama, J. & Kosina, P., 2014, 'Recurrent concepts in data streams classification', *Knowledge and Information Systems*, vol. 40, no. 3, pp. 489–507.

Gama, J., Medas, P., Castillo, G. & Rodrigues, P., 2004, 'Learning with Drift Detection', Bazzan, A. L. C. & Labidi, S. (eds.) *Advances in Artificial Intelligence – SBIA 2004*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 286–295.

Gama, J., Rocha, R. & Medas, P., 2003, 'Accurate Decision Trees for Mining High-Speed Data Streams', *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, Association for Computing Machinery, New York, NY, USA, pp. 523–528, event-place: Washington, D.C.

Gama, J., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A., 2014, 'A Survey on Concept Drift Adaptation', *ACM Comput. Surv.*, vol. 46, no. 4, place: New York, NY, USA Publisher: Association for Computing Machinery.

Gao, J., Fan, W., Jiang, J. & Han, J., 2008, 'Knowledge Transfer via Multiple Model Local Structure Mapping', *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, Association for Computing Machinery, New York, NY, USA, pp. 283–291, event-place: Las Vegas, Nevada, USA.

Ghomeshi, H., Gaber, M. M. & Kovalchuk, Y., 2019, 'EACD: evolutionary adaptation to concept drifts in data streams', *Data Mining and Knowledge Discovery*, vol. 33, no. 3, pp. 663–694.

Gomes, H. M., Barddal, J. P., Enembreck, F. & Bifet, A., 2017a, 'A Survey on Ensemble Learning for Data Stream Classification', *ACM Comput. Surv.*, vol. 50, no. 2, place: New York, NY, USA Publisher: Association for Computing Machinery.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G. & Abdessalem, T., 2017b, 'Adaptive random forests for evolving data stream classification', *Machine Learning*, vol. 106, no. 9, pp. 1469–1495.

Gomes, J. B., Gaber, M. M., Sousa, P. A. C. & Menasalvas, E., 2014, 'Mining Recurring Concepts in a Dynamic Feature Space', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 95–110.

Gong, B., Shi, Y., Sha, F. & Grauman, K., 2012, 'Geodesic flow kernel for unsupervised domain adaptation', *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2066–2073.

Gopalan, R., Li, R. & Chellappa, R., 2011, 'Domain adaptation for object recognition: An unsupervised approach', *2011 International Conference on Computer Vision*, pp. 999–1006.

Gözüaçık, O., Büyükçakır, A., Bonab, H. & Can, F., 2019, 'Unsupervised Concept Drift Detection with a Discriminative Classifier', *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, Association for Computing Machinery, New York, NY, USA, pp. 2365–2368, event-place: Beijing, China.

Graves, A., Wayne, G. & Danihelka, I., 2014, 'Neural Turing Machines', <https://arxiv.org/abs/1410.5401>.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B. & Smola, A., 2006, 'A Kernel Method for the Two-Sample-Problem', Schölkopf, B., Platt, J. & Hoffman, T. (eds.) *Advances in Neural Information Processing Systems*, , vol. 19MIT Press, pp. 513–520.

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

Gutjahr, W. J. & Sebastiani, G., 2008, 'Runtime Analysis of Ant Colony Optimization with Best-So-Far Reinforcement', *Methodology and Computing in Applied Probability*, vol. 10, no. 3, pp. 409–433.

Gâlmeanu, H. & Andonie, R., 2022, 'Weighted Incremental–Decremental Support Vector Machines for concept drift with shifting window', *Neural Networks*, vol. 152, pp. 528–541.

Halstead, B., Koh, Y. S., Riddle, P., Pears, R., Pechenizkiy, M., Bifet, A., Olivares, G. & Coulson, G., 2022, 'Analyzing and repairing concept drift adaptation in data stream classification', *Machine Learning*, vol. 111, no. 10, pp. 3489–3523.

Han, D., Giraud-Carrier, C. & Li, S., 2015, 'Efficient mining of high-speed uncertain data streams', *Applied Intelligence*, vol. 43, no. 4, pp. 773–785.

Harries, M., 1999, 'SPLICE-2 Comparative Evaluation: Electricity Pricing', Tech. rep., The University of South Wales.

Hazan, E., 2016, 'Introduction to Online Convex Optimization', *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325.

Hinder, F., Artelt, A. & Hammer, B., 2020, 'Towards Non-Parametric Drift Detection via Dynamic Adapting Window Independence Drift Detection (DAWIDD)', III, H. D. & Singh, A. (eds.) *Proceedings of the 37th International Conference on Machine Learning*, , vol. 119 of *Proceedings of Machine Learning Research*PMLR, pp. 4249–4259.

Hospedales, T., Antoniou, A., Micaelli, P. & Storkey, A., 2022, 'Meta-Learning in Neural Networks: A Survey', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5149–5169.

Hulten, G., Spencer, L. & Domingos, P., 2001, 'Mining Time-Changing Data Streams', *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, Association for Computing Machinery, New York, NY, USA, pp. 97–106, event-place: San Francisco, California.

Jiang, J. & Zhai, C., 2007, 'Instance Weighting for Domain Adaptation in NLP', *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Association for Computational Linguistics, Prague, Czech Republic, pp. 264–271.

Kauschke, S. & Fürnkranz, J., 2018, 'Batchwise Patching of Classifiers', *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1.

Kifer, D., Ben-David, S. & Gehrke, J., 2004, 'Detecting Change in Data Streams', Nascimento, M. A., Özsu, M. T., Kossmann, D., Miller, R. J., Blakeley, J. A. & Schiefer, K. B. (eds.) *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004*, Morgan Kaufmann, pp. 180–191.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu,

A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D. & Hadsell, R., 2017, 'Overcoming catastrophic forgetting in neural networks', *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, publisher: Proceedings of the National Academy of Sciences.

Koch, G., Zemel, R., Salakhutdinov, R. & others, 2015, 'Siamese neural networks for one-shot image recognition', *ICML deep learning workshop*, , vol. 2Lille, p. 0.

Kolter, J. Z. & Maloof, M. A., 2007, 'Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts', *Journal of Machine Learning Research*, vol. 8, no. 91, pp. 2755–2790.

Korycki, L. & Krawczyk, B., 2022a, 'Adversarial concept drift detection under poisoning attacks for robust data stream mining', *Machine Learning*.

Korycki, L. & Krawczyk, B., 2022b, 'Instance exploitation for learning temporary concepts from sparsely labeled drifting data streams', *Pattern Recognition*, vol. 129, p. 108749.

Krawczyk, B., 2021, 'Tensor decision trees for continual learning from drifting data streams', *Machine Learning*, vol. 110, no. 11, pp. 3015–3035.

Krawczyk, B. & Cano, A., 2019, 'Adaptive Ensemble Active Learning for Drifting Data Stream Mining', *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, pp. 2763–2771.

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

Lake, B. M., Ullman, T. D., Tenenbaum, J. B. & Gershman, S. J., 2016, 'Building Machines That Learn and Think Like People', <https://arxiv.org/abs/1604.00289>.

Lei, T., Jia, X., Zhang, Y., He, L., Meng, H. & Nandi, A. K., 2018, 'Significantly Fast and Robust Fuzzy C-Means Clustering Algorithm Based on Morphological Reconstruction and Membership Filtering', *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 3027–3041.

Lei, T., Jia, X., Zhang, Y., Liu, S., Meng, H. & Nandi, A. K., 2019, 'Superpixel-Based Fast Fuzzy C-Means Clustering for Color Image Segmentation', *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 9, pp. 1753–1766.

Lei, T., Liu, P., Jia, X., Zhang, X., Meng, H. & Nandi, A. K., 2020, 'Automatic Fuzzy Clustering Framework for Image Segmentation', *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 9, pp. 2078–2092.

Li, K. & Malik, J., 2017, 'Learning to Optimize', *International Conference on Learning Representations*, .

Li, P., Wu, X., Hu, X. & Wang, H., 2015, 'Learning concept-drifting data streams with random ensemble decision trees', *Neurocomputing*, vol. 166, pp. 68–83.

Li, Z., Zhou, F., Chen, F. & Li, H., 2017, 'Meta-SGD: Learning to Learn Quickly for Few-Shot Learning', <https://arxiv.org/abs/1707.09835>.

Liao, X., Xue, Y. & Carin, L., 2005, 'Logistic Regression with an Auxiliary Data Source', *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, Association for Computing Machinery, New York, NY, USA, pp. 505–512, event-place: Bonn, Germany.

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

Liu, A., Lu, J. & Zhang, G., 2021a, 'Concept Drift Detection via Equal Intensity k-Means Space Partitioning', *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3198–3211.

Liu, A., Lu, J. & Zhang, G., 2021b, 'Diverse Instance-Weighting Ensemble Based on Region Drift Disagreement for Concept Drift Adaptation', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 293–307.

Liu, A., Zhang, G. & Lu, J., 2017, 'Fuzzy time windowing for gradual concept drift adaptation', *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–6.

Lobo, J. L., Del Ser, J., Osaba, E., Bifet, A. & Herrera, F., 2021, 'CURIE: a cellular automaton for concept drift detection', *Data Mining and Knowledge Discovery*, vol. 35, no. 6, pp. 2655–2678.

Lobo, J. L., Laña, I., Ser, J. D., Bilbao, M. N. & Kasabov, N., 2018, 'Evolving Spiking Neural Networks for online learning over drifting data streams', *Neural Networks*, vol. 108, pp. 1–19.

Lobo, J. L., Ser, J. D., Bifet, A. & Kasabov, N., 2020, 'Spiking Neural Networks and online learning: An overview and perspectives', *Neural Networks*, vol. 121, pp. 88–100.

Long, M., Wang, J., Ding, G., Sun, J. & Yu, P. S., 2013, 'Transfer Feature Learning with Joint Distribution Adaptation', *2013 IEEE International Conference on Computer Vision*, pp. 2200–2207.

Losing, V., Hammer, B. & Wersing, H., 2016, 'KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift', *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 291–300.

Losing, V., Wersing, H. & Hammer, B., 2018, 'Enhancing Very Fast Decision Trees with Local Split-Time Predictions', *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 287–296.

Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S. & Zhang, G., 2015, 'Transfer learning using computational intelligence: A survey', *Knowledge-Based Systems*, vol. 80, pp. 14–23.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. & Zhang, G., 2019, 'Learning under Concept Drift: A Review', *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363.

Lu, N., Lu, J., Zhang, G. & Mantaras, R. L. d., 2016, 'A concept drift-tolerant case-base editing technique', *Artificial Intelligence*, vol. 230, pp. 108–133.

Lu, N., Zhang, G. & Lu, J., 2014, 'Concept drift detection via competence models', *Artificial Intelligence*, vol. 209, pp. 11–28.

Lu, Y., Cheung, Y.-M. & Yan Tang, Y., 2020, 'Adaptive Chunk-Based Dynamic Weighted Majority for Imbalanced Data Streams With Concept Drift', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 2764–2778.

Malialis, K., Panayiotou, C. G. & Polycarpou, M. M., 2021, 'Online Learning With Adaptive Rebalancing in Nonstationary Environments', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4445–4459.

Manapragada, C., Webb, G. I. & Salehi, M., 2018, 'Extremely Fast Decision Tree', *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge*

Discovery & Data Mining, KDD '18, Association for Computing Machinery, New York, NY, USA, pp. 1953–1962, event-place: London, United Kingdom.

Mann, H. B. & Whitney, D. R., 1947, 'On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other', *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50 – 60, publisher: Institute of Mathematical Statistics.

Mihalkova, L., Huynh, T. & Mooney, R. J., 2007, 'Mapping and Revising Markov Logic Networks for Transfer Learning', *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, AAAI Press, pp. 608–614, event-place: Vancouver, British Columbia, Canada.

Mihalkova, L. & Mooney, R. J., 2009, 'Transfer Learning from Minimal Target Data by Mapping across Relational Domains', *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1163–1168, event-place: Pasadena, California, USA.

Mokhtari, A., Shahrampour, S., Jadbabaie, A. & Ribeiro, A., 2016, 'Online optimization in dynamic environments: Improved regret rates for strongly convex problems', *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 7195–7201.

Ng, W. W. Y., Tian, X., Lv, Y., Yeung, D. S. & Pedrycz, W., 2017, 'Incremental Hashing for Semantic Image Retrieval in Nonstationary Environments', *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3814–3826.

Nguyen, T. T. T., Nguyen, T. T., Liew, A. W.-C. & Wang, S.-L., 2018, 'Variational inference based bayes online classifiers with concept drift adaptation', *Pattern Recognition*, vol. 81, pp. 280–293.

Nie, F., Liu, C., Wang, R., Wang, Z. & Li, X., 2022, 'Fast Fuzzy Clustering Based on Anchor Graph', *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 7, pp. 2375–2387.

Nishida, K. & Yamauchi, K., 2007, 'Detecting Concept Drift Using Statistical Testing', Corruble, V., Takeda, M. & Suzuki, E. (eds.) *Discovery Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 264–269.

Oza, N. C. & Russell, S., 2001, 'Experimental Comparisons of Online and Batch Versions of Bagging and Boosting', *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, Association for Computing Machinery, New York, NY, USA, pp. 359–364, event-place: San Francisco, California.

Pan, S. J., Tsang, I. W., Kwok, J. T. & Yang, Q., 2011, 'Domain Adaptation via Transfer Component Analysis', *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210.

Pan, S. J. & Yang, Q., 2010, 'A Survey on Transfer Learning', *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359.

Pesaranghader, A., Viktor, H. & Paquet, E., 2018, 'Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams', *Machine Learning*, vol. 107, no. 11, pp. 1711–1743.

Pinagé, F., dos Santos, E. M. & Gama, J., 2020, 'A drift detection method based on dynamic classifier selection', *Data Mining and Knowledge Discovery*, vol. 34, no. 1, pp. 50–74.

Pratama, M., Ashfahani, A. & Hady, A., 2019a, 'Weakly Supervised Deep Learning Approach in Streaming Environments', *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1195–1202.

Pratama, M., de Carvalho, M., Xie, R., Lughofer, E. & Lu, J., 2019b, 'ATL: Autonomous Knowledge Transfer from Many Streaming Processes', *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, Association for Computing Machinery, New York, NY, USA, pp. 269–278, event-place: Beijing, China.

Pratama, M., Dimla, E., Tjahjowidodo, T., Pedrycz, W. & Lughofer, E., 2020a, 'Online Tool Condition Monitoring Based on Parsimonious Ensemble+', *IEEE Transactions on Cybernetics*, vol. 50, no. 2, pp. 664–677.

Pratama, M., Pedrycz, W. & Webb, G. I., 2020b, 'An Incremental Construction of Deep Neuro Fuzzy System for Continual Learning of Nonstationary Data Streams', *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 7, pp. 1315–1328.

Rathore, P., Bezdek, J. C., Erfani, S. M., Rajasegarar, S. & Palaniswami, M., 2018, 'Ensemble Fuzzy Clustering Using Cumulative Aggregation on Random Projections', *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1510–1524.

Ravi, S. & Larochelle, H., 2017, 'Optimization as a Model for Few-Shot Learning', *International Conference on Learning Representations*, .

Raza, H., Prasad, G. & Li, Y., 2015, 'EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments', *Pattern Recognition*, vol. 48, no. 3, pp. 659–669.

Read, J. & Martino, L., 2020, 'Probabilistic regressor chains with Monte Carlo methods', *Neurocomputing*, vol. 413, pp. 471–486.

Read, J., Pfahringer, B., Holmes, G. & Frank, E., 2011, 'Classifier chains for multi-label classification', *Machine Learning*, vol. 85, no. 3, pp. 333–359.

Ross, G. J., Adams, N. M., Tasoulis, D. K. & Hand, D. J., 2012, 'Exponentially

weighted moving average charts for detecting concept drift', *Pattern Recognition Letters*, vol. 33, no. 2, pp. 191–198.

Roveri, M., 2019, 'Learning Discrete-Time Markov Chains Under Concept Drift', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2570–2582.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R. & Hadsell, R., 2016, 'Progressive Neural Networks', <https://arxiv.org/abs/1606.04671>.

Ryan, S., Corizzo, R., Kiringa, I. & 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.

Sahoo, D., Pham, Q., Lu, J. & Hoi, S. C. H., 2018, 'Online Deep Learning: Learning Deep Neural Networks on the Fly', *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, International Joint Conferences on Artificial Intelligence Organization, pp. 2660–2666.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D. & Lillicrap, T., 2016, 'Meta-Learning with Memory-Augmented Neural Networks', *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, JMLR.org, pp. 1842–1850, event-place: New York, NY, USA.

Schmidhuber, J., 1987, *Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook*, Diploma Thesis, Technische Universitat Munchen, Germany.

Schwaighofer, A., Tresp, V. & Yu, K., 2004, 'Learning Gaussian Process Kernels via Hierarchical Bayes', Saul, L., Weiss, Y. & Bottou, L. (eds.) *Advances in Neural Information Processing Systems*, , vol. 17MIT Press, pp. 1209–1216.

Shan, J., Zhang, H., Liu, W. & Liu, Q., 2019, 'Online Active Learning Ensemble Framework for Drifted Data Streams', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 486–498.

Shirkhorshidi, A. S., Wah, T. Y., Shirkhorshidi, S. M. R. & Aghabozorgi, S., 2021, 'Evolving Fuzzy Clustering Approach: An Epoch Clustering That Enables Heuristic Postpruning', *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 3, pp. 560–568.

Si, S., Tao, D. & Geng, B., 2010, 'Bregman Divergence-Based Regularization for Transfer Subspace Learning', *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 7, pp. 929–942.

Soleymani, F. & Paquet, E., 2020, 'Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath', *Expert Systems with Applications*, vol. 156, p. 113456.

Song, Y., Lu, J., Liu, A., Lu, H. & Zhang, G., 2022, 'A Segment-Based Drift Adaptation Method for Data Streams', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4876–4889.

Song, Y., Lu, J., Lu, H. & Zhang, G., 2020, 'Fuzzy Clustering-Based Adaptive Regression for Drifting Data Streams', *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 3, pp. 544–557.

Song, Y., Lu, J., Lu, H. & Zhang, G., 2021, 'Learning Data Streams With Changing Distributions and Temporal Dependency', *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14.

Spyromitros-Xioufis, E., Sechidis, K. & Vlahavas, I., 2020, 'Multi-target regression via output space quantization', *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W. & Vlahavas, I., 2016, 'Multi-target regression via input space expansion: treating targets as inputs', *Machine Learning*, vol. 104, no. 1, pp. 55–98.

Street, W. N. & Kim, Y., 2001, 'A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification', *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, Association for Computing Machinery, New York, NY, USA, pp. 377–382, event-place: San Francisco, California.

Stützle, T. & Hoos, H. H., 2000, 'MAX–MIN Ant System', *Future Generation Computer Systems*, vol. 16, no. 8, pp. 889–914.

Sun, Y., Tang, K., Minku, L. L., Wang, S. & Yao, X., 2016, 'Online Ensemble Learning of Data Streams with Gradually Evolved Classes', *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1532–1545.

Sun, Y., Tang, K., Zhu, Z. & Yao, X., 2018, 'Concept Drift Adaptation by Exploiting Historical Knowledge', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4822–4832.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. & Hospedales, T. M., 2018, 'Learning to compare: Relation network for few-shot learning', *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1199–1208.

Tahmasbi, A., Jothimurugesan, E., Tirthapura, S. & Gibbons, P. B., 2021, 'DriftSurf: Stable-State / Reactive-State Learning under Concept Drift', Meila, M. & Zhang, T. (eds.) *Proceedings of the 38th International Conference on*

Machine Learning, , vol. 139 of *Proceedings of Machine Learning Research*PMLR, pp. 10054–10064.

Thrun, S. & Pratt, L., 1998, 'Learning to Learn: Introduction and Overview', Thrun, S. & Pratt, L. (eds.) *Learning to Learn*, Springer US, Boston, MA, pp. 3–17.

Valiant, L. G., 1984, 'A Theory of the Learnable', *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, Association for Computing Machinery, New York, NY, USA, pp. 436–445.

Vayatis, N., Depecker, M. & Clémençcon, S., 2009, 'AUC optimization and the two-sample problem', Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. & Culotta, A. (eds.) *Advances in Neural Information Processing Systems*, , vol. 22Curran Associates, Inc., pp. 360–368.

Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k. & Wierstra, D., 2016, 'Matching Networks for One Shot Learning', Lee, D., Sugiyama, M., Luxburg, U., Guyon, I. & Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, , vol. 29Curran Associates, Inc., pp. 3630–3638.

Wang, B., Lu, J., Yan, Z., Luo, H., Li, T., Zheng, Y. & Zhang, G., 2019, 'Deep Uncertainty Quantification: A Machine Learning Approach for Weather Forecasting', *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, Association for Computing Machinery, New York, NY, USA, pp. 2087–2095, event-place: Anchorage, AK, USA.

Wang, H. & Abraham, Z., 2015, 'Concept drift detection for streaming data', *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9.

Wang, X., Kang, Q., Zhou, M., Pan, L. & Abusorrah, A., 2021, 'Multiscale Drift Detection Test to Enable Fast Learning in Nonstationary Environments', *IEEE Transactions on Cybernetics*, vol. 51, no. 7, pp. 3483–3495.

Webb, G. I., Lee, L. K., Goethals, B. & Petitjean, F., 2018, 'Analyzing concept drift and shift from sample data', *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1179–1199.

Widmer, G. & Kubat, M., 1996, 'Learning in the Presence of Concept Drift and Hidden Contexts', *Machine Learning*, vol. 23, no. 1, pp. 69–101.

Xu, S. & Wang, J., 2017, 'Dynamic extreme learning machine for data stream classification', *Neurocomputing*, vol. 238, pp. 433–449.

Yang, M.-S. & Nataliani, Y., 2018, 'A Feature-Reduction Fuzzy Clustering Algorithm Based on Feature-Weighted Entropy', *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 2, pp. 817–835.

Yang, Y., Zhou, D.-W., Zhan, D.-C., Xiong, H. & Jiang, Y., 2019, 'Adaptive Deep Models for Incremental Learning: Considering Capacity Scalability and Sustainability', *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, Association for Computing Machinery, New York, NY, USA, pp. 74–82, event-place: Anchorage, AK, USA.

Yang, Z., Al-Dahidi, S., Baraldi, P., Zio, E. & Montelatici, L., 2020, 'A Novel Concept Drift Detection Method for Incremental Learning in Nonstationary Environments', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 1, pp. 309–320.

Yen, S., Moh, M. & Moh, T.-S., 2019, 'CausalConvLSTM: Semi-Supervised Log

Anomaly Detection Through Sequence Modeling', *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1334–1341.

Yu, H., Liu, A., Wang, B., Li, R., Zhang, G. & Lu, J., 2020, 'Real-Time Decision Making for Train Carriage Load Prediction via Multi-stream Learning', Gallagher, M., Moustafa, N. & Lakshika, E. (eds.) *AI 2020: Advances in Artificial Intelligence*, Springer International Publishing, Cham, pp. 29–41.

Yu, H., Lu, J. & Zhang, G., 2022a, 'Continuous Support Vector Regression for Nonstationary Streaming Data', *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3592–3605.

Yu, H., Lu, J. & Zhang, G., 2022b, 'Topology Learning-Based Fuzzy Random Neural Networks for Streaming Data Regression', *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 2, pp. 412–425.

Yu, S., Abraham, Z., Wang, H., Shah, M., Wei, Y. & Príncipe, J. C., 2019, 'Concept drift detection and adaptation with hierarchical hypothesis testing', *Journal of the Franklin Institute*, vol. 356, no. 5, pp. 3187–3215.

Yu, S., Wang, X. & Príncipe, J. C., 2018, 'Request-and-Reverify: Hierarchical Hypothesis Testing for Concept Drift Detection with Expensive Labels', *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, International Joint Conferences on Artificial Intelligence Organization, pp. 3033–3039.

Yuan, L., Li, H., Xia, B., Gao, C., Liu, M., Yuan, W. & You, X., 2022, 'Recent Advances in Concept Drift Adaptation Methods for Deep Learning', Raedt, L. D. (ed.) *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, International Joint Conferences on Artificial Intelligence Organization, pp. 5654–5661.

Zaragoza, J. H., Sucar, L. E., Morales, E. F., Bielza, C. & Larrañaga, P., 2011, 'Bayesian Chain Classifiers for Multidimensional Classification', *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, AAAI Press, pp. 2192–2197, event-place: Barcelona, Catalonia, Spain.

Zhang, H., Liu, W. & Liu, Q., 2022, 'Reinforcement Online Active Learning Ensemble for Drifting Imbalanced Data Streams', *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3971–3983.

Zhang, L., 2020, 'Online Learning in Changing Environments', Bessiere, C. (ed.) *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, International Joint Conferences on Artificial Intelligence Organization, pp. 5178–5182.

Zhang, M.-L. & Zhou, Z.-H., 2014, 'A Review on Multi-Label Learning Algorithms', *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837.

Zhang, P., Zhu, X. & Shi, Y., 2008, 'Categorizing and Mining Concept Drifting Data Streams', *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, Association for Computing Machinery, New York, NY, USA, pp. 812–820, event-place: Las Vegas, Nevada, USA.

Zhang, Y., Chu, G., Li, P., Hu, X. & Wu, X., 2017, 'Three-layer concept drifting detection in text data streams', *Neurocomputing*, vol. 260, pp. 393–403.

Zhao, P., Cai, L.-W. & Zhou, Z.-H., 2020, 'Handling concept drift via model reuse', *Machine Learning*, vol. 109, no. 3, pp. 533–568.

Zhou, S., Li, D., Zhang, Z. & Ping, R., 2021, 'A New Membership Scaling Fuzzy C-Means Clustering Algorithm', *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 9, pp. 2810–2818.

Zhou, Y., 2009, 'Runtime Analysis of an Ant Colony Optimization Algorithm for TSP Instances', *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1083–1092.

Zinkevich, M., 2003, 'Online Convex Programming and Generalized Infinitesimal Gradient Ascent', *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, AAAI Press, pp. 928–935, event-place: Washington, DC, USA.

Álvarez, V., Mazuelas, S. & Lozano, J. A., 2022, 'Minimax Classification under Concept Drift with Multidimensional Adaptation and Performance Guarantees', Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G. & Sabato, S. (eds.) *Proceedings of the 39th International Conference on Machine Learning*, , vol. 162 of *Proceedings of Machine Learning Research*PMLR, pp. 486–499.