8-2022

Solving the challenges of concept drift in data stream classification.

Hanqing Hu
University of Louisville

SOLVING THE CHALLENGES OF CONCEPT DRIFT IN DATA STREAM CLASSIFICATION

By

Hanqing Hu
B.S., Miami University, 2012
M.S., Miami University, 2012

A Dissertation
Submitted to the Faculty of the
J.B. School of Engineering of the University of Louisville
In Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
In Computer Science and Engineering

Department of Computer Science and Computer Engineering
University of Louisville
Louisville, Kentucky

August 2022

SOLVING THE CHALLENGES OF CONCEPT DRIFT IN DATA STREAM CLASSIFICATION

By

Hanqing Hu
B.S., Miami University, 2012
M.S., Miami University, 2012

A Dissertation Approved On

6/6/2022

By the following Dissertation Committee:

_____
Mehmed Kantardzic

_____
Adel Elmaghraby

_____
Adrian P. Lauf

_____
James E. Lewis

_____
Nihat Altiparmak

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Mehmed Kantardzic, for his guidance and patience. I would also like to thank the other committee members, Dr. Adel Elmaghraby, Dr. Adrian P. Lauf, Dr. James Lewis and Dr. Nihat Altiparmak, for their comments and assistance over the years. I would also like to express my thanks to my parents and my friends for their support and encouragement.

ABSTRACT

SOLVING THE CHALLENGES OF CONCEPT DRIFT IN DATA STREAM CLASSIFICATION

Hanqing Hu

6/6/2022

The rise of network connected devices and applications leads to a significant increase in the volume of data that are continuously generated overtime time, called data streams. In real world applications, storing the entirety of a data stream for analyzing later is often not practical, due to the data stream's potentially infinite volume. Data stream mining techniques and frameworks are therefore created to analyze streaming data as they arrive. However, compared to traditional data mining techniques, challenges unique to data stream mining also emerge, due to the high arrival rate of data streams and their dynamic nature. In this dissertation, an array of techniques and frameworks are presented to improve the solutions on some of the challenges.

First, this dissertation acknowledges that a "no free lunch" theorem exists for data stream mining, where no silver bullet solution can solve all problems of data stream mining. The dissertation focuses on detection of changes of data distribution in data stream mining. These changes are called concept drift. Concept drift can be categorized into many types. A detection algorithm often works only on some types of drift, but not

all of them. Because of this, the dissertation finds specific techniques to solve specific challenges, instead of looking for a general solution.

Then, this dissertation considers improving solutions for the challenges of high arrival rate of data streams. Data stream mining frameworks often need to process vast among of data samples in limited time. Some data mining activities, notably data sample labeling for classification, are too costly or too slow in such large scale. This dissertation presents two techniques that reduce the amount of labeling needed for data stream classification. The first technique presents a grid-based label selection process that apply to highly imbalanced data streams. Such data streams have one class of data samples vastly outnumber another class. Many majority class samples need to be labeled before a minority class sample can be found due to the imbalance. The presented technique divides the data samples into groups, called grids, and actively search for minority class samples that are close by within a grid. Experiment results show the technique can reduce the total number of data samples needed to be labeled. The second technique presents a smart preprocessing technique that reduce the number of times a new learning model needs to be trained due to concept drift. Less model training means less data labels required, and thus costs less. Experiment results show that in some cases the reduced performance of learning models is the result of improper preprocessing of the data, not due to concept drift. By adapting preprocessing to the changes in data streams, models can retain high performance without retraining.

Acknowledging the high cost of labeling, the dissertation then considers the scenario where labels are unavailable when needed. The framework Sliding Reservoir Approach for Delayed Labeling (SRADL) is presented to explore solutions to such

problem. SRADL tries to solve the delayed labeling problem where concept drift occurs, and no labels are immediately available. SRADL uses semi-supervised learning by employing a sliding windowed approach to store historical data, which is combined with newly unlabeled data to train new models. Experiments show that SRADL perform well in some cases of delayed labeling.

Next, the dissertation considers improving solutions for the challenge of dynamism within data streams, most notably concept drift. The complex nature of concept drift means that most existing detection algorithms can only detect limited types of concept drift. To detect more types of concept drift, an ensemble approach that employs various algorithms, called Heuristic Ensemble Framework for Concept Drift Detection (HEFDD), is presented. The occurrence of each type of concept drift is voted on by the detection results of each algorithm in the ensemble. Types of concept drift with votes past majority are then declared detected. Experiment results show that HEFDD is able to improve detection accuracy significantly while reducing false positives.

With the ability to detect various types of concept drift provided by HEFDD, the dissertation tries to improve the delayed labeling framework SRADL. A new combined framework, SRADL-HEFDD is presented, which produces synthetic labels to handle the unavailability of labels by human expert. SRADL-HEFDD employs different synthetic labeling techniques based on different types of drift detected by HEFDD. Experimental results show that comparing to the default SRADL, the combined framework improves prediction performance when small amount of labeled samples is available.

Finally, as machine learning applications are increasingly used in critical domains such as medical diagnostics, accountability, explainability and interpretability of machine

learning algorithms needs to be considered. Explainable machine learning aims to use a white box approach for data analytics, which enables learning models to be explained and interpreted by human users. However, few studies have been done on explaining what has changed in a dynamic data stream environment. This dissertation thus presents Data Stream Explainability (DSE) framework. DSE visualizes changes in data distribution and model classification boundaries between chunks of streaming data. The visualizations can then be used by a data mining researcher to generate explanations of what has changed within the data stream. To show that DSE can help average users understand data stream mining better, a survey was conducted with an expert group and a non-expert group of users. Results show DSE can reduce the gap of understanding what changed in data stream mining between the two groups.

TABLE OF CONTENT

PAGE

LIST OF FIGURES

FIGURE PAGE

LIST OF TABLES

TABLE PAGE

CHAPTER 1. INTRODUCTION

The increased use of network connected Internet-of-Things, social networks and other digitally connected technologies means that the data from these sources have also increased significantly. Data from these sources are often generated continuously, and are called data streams (Gao et al, 2007). Mining these data stream can produce valuable knowledges, but traditional data mining techniques might not be equipped to handle the unique characteristics of data stream mining. Three major differences of characteristic exist:

1. *Fixed Size vs Unknown Size of Data.* Traditional data classification tasks are performed on fixed- size dataset. The data set contains all available data instances and is usually divided into training data and testing data. Data streams usually does not have a known size (Zliobaite et al., 2014). There is no access to all available data instances. Only a small portion of data samples are gathered from the stream to train a new learning model.
2. *Static vs Dynamic Data.* Because data set used in traditional classification has definitive size and is fully available, all characteristics of the data is known, such as data distribution and class imbalance. Traditional dataset is therefore static. In data streams, the current data characteristics may not represent the entire data stream. The data distribution often changes unpredictably and therefore it is dynamic (Webb et al., 2016).

3. *One-time Processing vs Repeated Processing*. Data processing tasks include cleaning, annotating, and labeling the data. Since traditional dataset is fully available, such tasks can be done once. For data streams, processing tasks might be done repeatedly as new data instances arrives overtime. Many processing tasks also has high cost (Ramírez-Gallego et al, 2017), which makes repeat expensive.

Data stream mining frameworks and techniques are thus created to meet the requirement for these new characteristics of streaming data. In practice, however, issues of data stream mining emerge due to the size and dynamism of data streams, which hinders the performance and practicality of the frameworks (Krempl et al, 2014). This dissertation identifies three major challenges that need to be addressed for data stream mining:

1.1 Challenge One: Dynamisms in Data Streams – Concept Drift

Because data streams often have unknown size, it is impractical to store all data in a data stream and analyze together. Each portion of data used in analysis therefore only shows the snapshot of the current data stream environment. In an evolving data stream, the data environment, which include range of data and distribution of data, often changes. Learning models trained on previous portions of data stream might have lower performance in the new data environment.

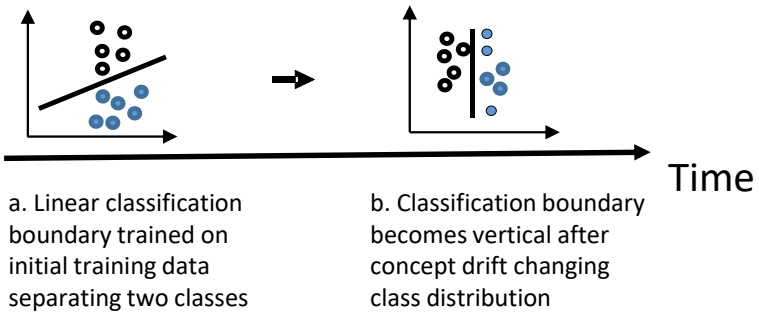


Figure 1.1. Demonstrating concept drift

Formally, in the domain of data stream classification, the underlying data distribution of the dataset is called a concept (Webb et al., 2016)(Lu et al, 2018). This definition of concept considers data samples as n-dimensional vector of features. One can then plot the data samples onto a n-dimensional data space. The space where the data sample occupies is the distribution of the data, or the concept. A concept drift describes the change in such data distribution. Another definition considers concept as the true classification prediction target (Harel et al, 2014). A concept drift describes the change in the prediction results for similar data samples. The two definitions thus define concept both from the angle of prediction input and prediction outcome. An illustrative example is shown in Figure 1.1. A linear classification model was trained in Figure 1.1.a, using existing two-dimensional training data at the beginning of the time axis. The linear model separates two classes of samples denoted by empty and filled circles. After some time has passed, in Figure 1.1.b the underlying data distribution changed, creating concept drift. This concept drift forced the linear classification model to change to a new vertical one.

Concept drift can take other forms and have other effect on learning models. To discuss concept drift in general, a concept of data distribution can be expressed through probabilities, as shown in equation 1.1 (Gao et al., 2007).

$$P(X, Y) = P(Y|X) * P(X) \quad (1.1)$$

P(X, Y) is the joint probability of data sample X with respect to class label Y. Equation 1.1 states that the probability for data sample X to be of Y label equals to probability of label Y given sample X (P(Y|X)) multiply by probability of sample X (P(X)). P(X) means how likely for a sample X to appear in the data stream. A higher P(Y|X) means

samples are more probable to be of class Y given the values of data feature X. A change in P(Y|X) in data stream thus means the classification criteria have changed, since the same X values now have a different probability of being class Y. A change in P(X) means that the likelihood of value X within the data stream changed. This implies that there are changes statistical distribution of feature values. Combined, equation 1.1 reflects both definitions of concept.

There are many more types of concept drift than shown in Figure 1.1 (Minku et al., 2010)(Gama et al., 2014)(Webb et al., 2016). Concept drifts can be categorized based on the characteristics of the change in underlying data environment. For example, one can divide concept drift based on how fast the change occurs into abrupt and gradual drift. Other categories also exist for how severe the change is and how many times it repeats.

One assumption of concept drift is that the timing and types of future concept drift are initially unknown to the learners, hence unpredictable. Most machine learning frameworks need first detecting a concept drift occurs, then react to it by trying to learn the new data distribution and new classification model (Ramakrishna & Rao, 2017) (Sethi et al., 2018). This approach is illustrated in Figure 1.2, which showed three main components in the data stream classification framework: Concept Drift Detection Unit, Classification

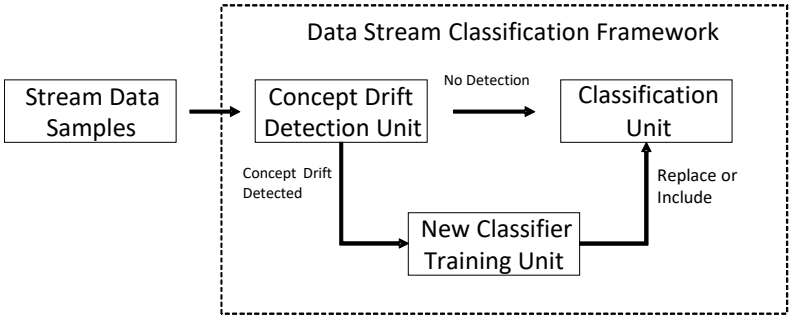


Figure 1.2. Flow chart for a common data stream classification framework

Unit and New Classifier Training Unit. In the beginning of a classification task, Concept Drift Detection Unit and Classification Unit are initialized using existing training data collected from the data stream. During initialization, parameters of the training process are determined, and necessary classifiers are trained. As new unclassified data samples from stream are ready for classification, they are first tested through Concept Drift Detection Unit. If there is no concept drift, the samples are then sent to the Classification Unit for prediction. If concept drift is detected, new classifiers will be trained using most recent data samples in the New Classifier Training Unit. The new classifiers are then used to replace or to combine with the existing classifiers. This puts concept drift detection in a critical spot for most data stream classification frameworks.

Designing high performance concept drift detection approach is not a trivial matter. There is often a trade-off between cost efficiency and performance among frameworks (Zliobaite et al., 2014)(Sethi & Kantardzic, 2017). On one hand, relative high performance can be achieved with labeled data, but labeling or even only partial labeling of an unlabeled, high-volume, indefinite-sized data stream using human experts may involve high cost (Zliobaite et al., 2014). This leads to the next challenge: high cost in data stream mining.

1.2 Challenge Two: High Cost in Data Stream Mining

There is cost associate with every aspect of data stream mining (De Francisci Morales, et al., 2016). However, labeling cost stands out among others because the unavoidable involvement of human labor (Žliobaite, 2010): if a machine exists that can label samples for model training, then one can use the same machine directly for classification instead. The scale of modern-day machine learning applications and the volume of data makes high label availability a luxury. To highlight the problem of labeling

cost, consider the task of detecting hate speech from live tweets (Burnap & Williams, 2016), using a classification system facing the twitter stream (estimated at 500M daily tweets). If only 0.5% of the tweets are requested to be labeled, using crowd sourcing websites such as Amazon's Mechanical Turk, this would imply a daily expenditure of $50K (each worker paid $1 for 50 tweets). It will also require a continuous availability of 350 crowd sourced workers (assuming each can label 10 tweets per minute, and work for 12 hours/day), every single day, for this task alone. Due to practical and economic limitations, data stream classification applications need to be able to operate from unlabeled, or at most sparsely labeled data, to be of any real use.

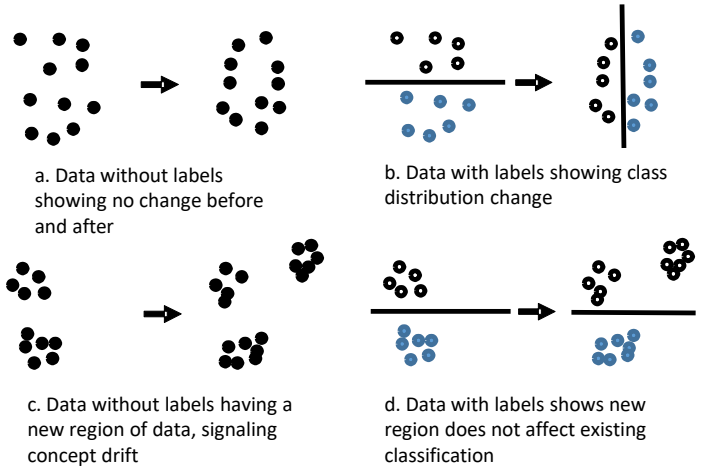


Figure 1.3. Detecting various types of concept drift is difficult without label

Although unlabeled data can be used for concept drift detection, eliminating the cost of human intervention (dos Reis et al., 2016)(Cabanes & Bennani, 2012), this often results in lowered detection performance, since concept drifts take on various forms and are unpredictable (Hu et al., 2018). To illustrate the difficulty involved in unlabeled concept drift detection, two examples are shown in Figure 1.3. In Figure 1.3.a, the data stream seemingly undergoes no change when labels information is absent. Once labels are obtained in Figure 3.b, concept drift is shown rotated initial classification model from

horizontal to vertical. In this case, a false negative of concept drift detection is likely to be produced. In Figure 3.c, a new group of data samples appeared outside of existing data distribution, thus signaling a possible concept drift. After obtaining labels in Figure 1.3.b, the new samples are of the same class as the empty circles. There is no real concept drift because the classification model is not affected. In this case, a false positive detection is likely produced. Excessive false alarms make data stream mining framework unreliable (Wares et al, 2019).

To reduce the cost associated with labeling, semi-supervised learning methods are often employed (Reddy et al, 2018). A semi-supervised learning methods only requires part of the training dataset to be labeled, thus reducing labeling cost. However, the same trade-off between performance and cost still exists (Li & Liang, 2019). Semi-supervised learning can only reduce labeling cost to a certain degree before the lack of labels impacting the performance. It is therefore important to inform users of data stream mining frameworks whether such unreliability is prevalent in their applications. By explaining how data stream has changed, a user can make educated decision on whether the stream mining application is working as intended. This brings the third challenge: explain changes in data stream.

1.3 Challenge Three: Explainable Machine Learning and Explanation of Change in Data Stream Mining

Current machine learning frameworks mostly operate like a black-box (Holzinger, 2018): uses cannot see how the learning model is trained and how a classification decision is made. The problem of such black-box approach is the lack of understanding and accountability (Veale et al., 2018). If accident occur, in some cases it is impossible to tell

if it is the user's fault or the data mining algorithm's fault. A good example is Neural Network deployed for medical image diagnostic (Singh et al., 2020). As machine learning application becomes more and more mainstream in every aspect of society, the accountability of the learning models has become an important topic due to the following reasons (Adadi & Berrada, 2018):

1. A need to justify prediction result. Current Artificial Intelligence/Machine Learning (AI/ML) applications are not perfect. There needs to have some mechanism to explain the decision-making process when unexpected results are obtained. Through the decision-making process we can arrive at reasons and justifications for the decisions being made. In addition, many AI/ML enabled systems yield results that are biased and discriminatory (Caruana et al, 2015)(Howard et al, 2017). Model explainability can show bias exists and provide important information on where the bias and discrimination are generated within the process. Explainability ensures a way to prove that algorithmic decisions are fair and ethical.

2. User's right to explanation. This is an EU regulation included in the General Data Protection Regulation being in effect since May 25th, 2018. The right to explanation states that "Such rights primarily refer to individual rights to be given an explanation for decisions that significantly affect an individual, particularly legally or financially". For critical applications involving AI/ML, such as medical decisions and loan approval, the end user receiving the decision from some algorithm has the right to be informed how the decision about them is made.

3. Accountability and responsibility. In the case where AI/ML applications has made major errors that result in significant loss, it is important to identify all responsible parties involved. Explainability will be able to answer important questions, including but not limited to: Was the algorithm implemented correctly? Was the learning model trained correctly and sufficiently without bias? Was this application outside the capability of the AI algorithm? Was the application deployed correctly given the training data? By answering these questions, one can see whether the application developers made mistakes, or the application users operate it wrong, or the application is simply outside the capability of existing algorithms.

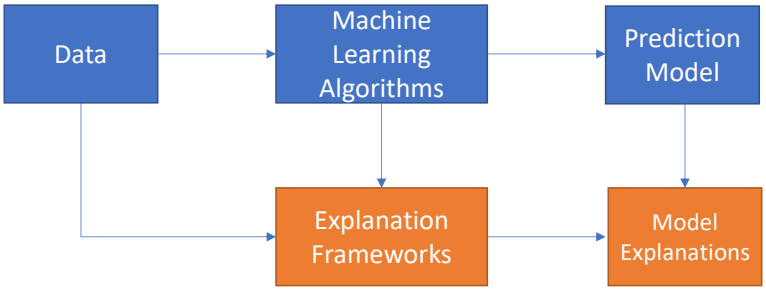
4. Control of errors. Explainability not only can justify decisions and assign responsibility when something goes wrong, it also can prevent errors being made in the first place. Explainability provide visibility to inner workings of the AI algorithm, which potentially exposes limitations, vulnerabilities and flaws. Understanding the limitations of the algorithm prevent applications to be used outside its own capability. Knowledge of the vulnerabilities prevent the application to be taken advantage of by adversaries. Awareness of the flaws reduces the damage and loss when an erroneous decision is made.

5. Improving algorithm. Many AI/ML applications need constant improvement to meet ever increasing demand. It is easier to improve an AI/ML model when it can be explained and understood compared to a black-box model. In explainable models, experts can find out the process behind an erroneous AI decision and find a way to fix the error.

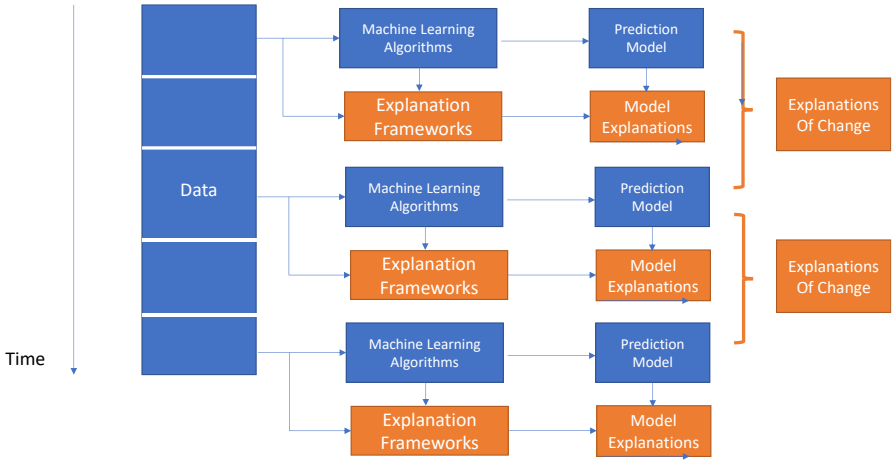
6. Gaining knowledge. An explainable AI/ML application can increase human understanding of the problem at hand. A good performing AI/ML model may find hidden patterns and relationships that human expert previously overlooked. For instance, an AI model exceling in early detection of disease may help doctors better understanding the cause and progression of the disease. Many mathematic and scientific discoveries come from simulations of theories at hand and then apply real data to confirm it. It is reasonable to speculate that machine learning can also help in similar ways by making sense of the data at hand first and then formulate theories to understand them. To achieve such goal, the explainability of the AI/ML model is required.

Model Explainability is the ability to explain to machine learning application users how a prediction or recommendation decision is made (Holzinger, 2018). It aims to be a white-box approach to machine learning application, as opposed to traditional black-box approaches. Existing state-of-the-art studies on Explainable Machine Learning focus on providing explanation to static learning models (Roscher et al. 2020)(Linardatos et al., 2020). As mentioned above, data stream classification frameworks employ different strategies than traditional frameworks due to dynamisms in streaming data. Because of the differences, static explainable machine learning approach, though adequate for traditional data mining applications, may not be sufficient for data stream mining applications (Hoens et al, 2012). Figure 1.4 demonstrates the different between static explainable machine learning versus explainability in a data stream.

Figure 1.4.a shows the structure of a typical static model explanation framework (Lundberg and Lee, 2017). A prediction model is trained using the static data. At the same time, the model explanations are generated uses the same data set and information from the



a. Traditional model explanation framework construct model explanation with static model



b. Explanation of changes in model in a dynamic data stream mining application

Fig 1.4 Difference between static model explanation framework structure vs dynamic explanation of change framework structure

trained model. Figure 1.4.b shows in a stream mining framework, multiple learning models might be created, and each will require a model explanation. Besides the model explanations, there should be extra explanations of change needed for data stream mining applications. These explanations should show what has changed in the data stream that caused the changes in the learning model. As discussed above, the unpredictability of concept drift and the limited label availability can result in many false alarms of concept

drift detection. In the case of concept drift being detected, it is important to understand why it is detected and how the drift affects previous learning models. Through its explanation, one can decide whether the detection is legitimate or a false positive. Equally important is that in the case of concept drift not being detected, one should know if it is a false negative. In either case, the explanation of changes brings interpretability and accountability of concept drift detection to data stream mining frameworks,

1.4 Solving Data Stream Mining Challenges

This dissertation aims to find solutions that can help to solve the three aforementioned challenges for data stream mining. The complexity of data stream mining means that finding a best solution might be difficult, or impossible. Chapter 2 of the dissertation shows that improving data stream mining follows the "No Free Lunch" theorem of optimization. The theorem states that no general-purpose optimization strategy exists, only specific solutions for specific problems. In the case of data stream mining, different types of concept drift require different assumptions to be made about the nature of the data stream. Without prior assumptions, a general optimized solution doesn't exist. Therefore, the dissertation presents specific solutions to solve specific problems within the domain of data stream mining.

The dissertation first presents two techniques that try to reduce the impact of labeling cost challenge associated with data stream mining. Chapter 3 formulates a sample selection algorithm that reduces the number of labels required for model training in highly imbalanced data stream classification. An imbalanced data stream has one class of data samples outnumbers another class. To train a learning model separating the two classes, labeled data samples from both classes are needed. In the case of a highly imbalanced data

stream, many data samples might need to be labeled before a single minority class samples can be found. To avoid unnecessary labeling of non-minority class samples, the presented algorithm search for minority data close to existing known minority class samples within a data grid. Result shows that this approach can reduce the number of samples needed for labeling. Chapter 4 shows adaptive preprocessing can reduce the frequency of retraining new models due to changes in the data stream. The presented preprocessing framework adjusts preprocessing parameters based on the new data range after concept drift is detected. Results show that in some cases, detected concept drifts were false positives and the real reason was improper preprocessing parameters. Thus, the framework can reduce the number of unnecessary model retraining, and reduce labeling costed associated with retraining.

Acknowledging limited label availability due to cost, Chapter 5 considers solutions for when label is not available when needed. The delayed labeling problem is presented in this Chapter. The problem occurs when labels are required for new model training after concept drift, but not immediate available. The framework Sliding Reservoir Approach for Delayed Labeling (SRADL) is presented. SRADL uses semi-supervised learning by employing a sliding windowed approach to store historical labeled data, which is combined with newly unlabeled data to train new models. The Chapter defined two scenarios of delayed labeling. The first scenario has all labels available only after a fixed delayed time. The second has small amounts of labeled samples trickling in, until all labels becoming available after the same fixed delayed time. Experiment results show that SRADL improve performance in the second scenario due to semi-supervised learning taking advantages of small numbers of labeled data.

Next, the dissertation considers improvements in solutions to the concept drift detection challenge of data stream mining. Concept drift can be divided into many different types based on different characteristics of the change. Existing detection algorithms often makes assumptions about some characteristics of the concept drift being detected. The "No Free Lunch" theorem in Chapter 2 shows that these algorithms are not the general solution to detect all concept drift due to their assumptions. To increase the types of concept drift detected, an ensemble approach that employs various algorithms, called Heuristic Ensemble Framework for Concept Drift Detection (HEFDD), is presented in Chapter 6. Since multiple algorithms are employed in the ensemble, the detection range of concept drift types can be increased. The occurrence of each type of concept drift is voted on by the detection results of each algorithm in the ensemble. Types of concept drift with votes past majority are then declared detected. Experiment results show that HEFDD can improve detection accuracy significantly while reducing false positives.

Since HEFDD provides the ability to detect different types of concept drift, such information can help producing synthetic labels in semi-supervised learning. A new combined framework, SRADL-HEFDD is presented in Chapter 7. SRADL-HEFDD employs different synthetic labeling techniques based on different types of drift detected. The synthetic labels are then used in semi-supervised model training as temporary replacement for true labels generated by human experts. The correctness of synthetic labels is of major concern, especially in high dimensional data streams. Experimental results show that comparing to the default SRADL, the combined framework improves prediction performance in low dimensional data stream.

Chapter 8 of the dissertation presents a framework that aims to bring explainbility of change to data stream mining. The Data Stream Explainability (DSE) framework divides data stream into chunks of data and visualizes data distribution and model classification boundaries for each chunk. Visualizations from multiple chunks are compared and changes of the data stream can be explored. Visualizations from both concept drifting and non-drifting chunks are shown such that false positive and false negative can be identified. The visualizations can then be used by a data mining researcher to generate explanations of what has changed or has not changed within the data stream. To show that DSE can help average users understand data stream mining better, a survey was conducted with an expert group and a non-expert group of users. Results show DSE can reduce the gap of understanding what changed in data stream mining between the two groups.

CHAPTER 2. NO FREE LUNCH THEOREM FOR CONCEPT DRIFT DETECTION[1]

As mentioned in chapter 1, data stream classification has become a very important topic of study as the amount of digital data increases rapidly Traditional classification problems consist of learning an underlying data distribution, called concept, among several data classes from a static dataset. The dataset is assumed to contain all information needed for training classifiers. This model is insufficient when applying to real-world data stream classification applications because many of those applications, such as online sentiment analysis, intrusion detection, and fraud detection, have dynamic data stream. For example, a new type of credit card fraud can appear that tries to circumvent existing fraud detection set in place. These changes in data streams, named concept drift (Pinage et al., 2016)(Sethi et al., 2018), often affect the underlying data distribution and reduce the performance of existing classifiers.

Concept drift is a complex phenomenon. There can be many different types of concept drifts, categorized by speed, severity and distribution of change. It is important to identify the best detection approaches to be used under each specific stream classification application environment, which include labeled/unlabeled data, fast/slow drift,

[1] This chapter has been published at Hu et al, 2020.

repeated/non-repeated drift, etc. It is difficulty to have a general-purpose universal strategy for concept drift. This is why concept drift detection follows the No Free Lunch Theorem (Ho & Pepyne, 2002), which means that there is no one-fit-all approach for concept drift detection. This chapter looks at ways to classify concept drift and the studies that try to handle these different types of algorithms.

2.1 Concept Drift with New Class or New Feature

In some real-word applications, it is reasonable to assume that the number of classes may not be constant. This situation happens when the new samples are considered a new class (Masud et al., 2009), demonstrated in Figure 2.1. Figure 2.1.a shows the stream initially having two class, circle and triangle, with a dashed line between the two classes as classification decision boundary. Later on in the stream, new samples (square) arrive,

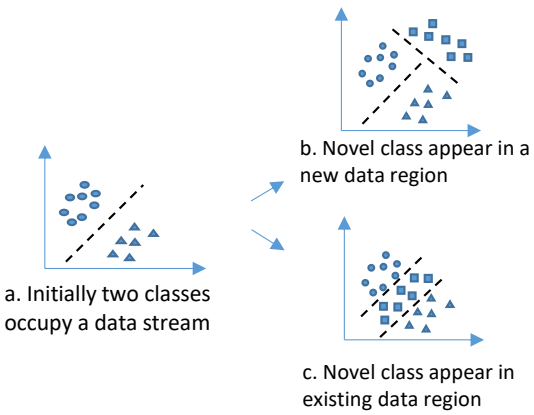


Figure 2.1. Novel class in streaming data

and they are considered as a novel class. There can be two cases of novel class. The first case is shown in Figure 2.1.b. where new class samples appear outside of existing circle and triangle class, and the decision boundary between circle and triangle remains the same. The second case is shown in Figure 2.1.c. Here, new square samples appear between the circle samples and triangle samples, which changes the original decision boundary. A

possible application that involves such drift is online topic classification, where each topic is a class, and a classification framework tries to decide which text message belongs to which topic. New topics can appear while old topics can disappear in online conversation. Thus, the classification framework needs to deal with new class labels and forgetting old class labels.

Masud et al. (Masud et al., 2009) proposed a non-parametric ensemble approach that detects emergence of novel class by separating test and training instances and measuring cohesion among unlabeled test instances. More recent studies also employ the ensemble strategy. Lughofer et al (Lughofer el al., 2015) employs evolving fuzzy classifiers (EFCs) to integrate new classes. EFCs is able to incrementally adapt structure and parameters on the fly when new classes appear. The approach is applied to visual inspection in the study but can be readily extended to stream classification. Haque and Baron (Haque et al., 2016) proposed SAND: Semi-Supervised Adaptive Novel Class Detection and Classification. The SAND framework keeps an ensemble of classifiers, each of which trained on a dynamically determined chunk of data. The ensemble detects novel class using similar cohesion measurement among unclassified outliers. Al-Khateeb et al. (Al-Khateeb et al., 2016) proposed a class-based ensemble that distinguish between novel class and recurring class. A recurring class is a class that disappear in the stream then reappear later on. The base learner of the ensemble is trained based on each class and "remembers" existing class information. Sun et al. (Sun et al., 2016) also proposed a class-based ensemble that is able to handle gradually evolved classes. Mustafa et al. (Mustafa et al., 2017) proposed an approach that utilize feature learning and denoising autoencoding to detect novel class. It is a neural-network-based approach that is non-parametric.

Mohamad et al. (Mohamad et al., 2018) proposed an active learning framework for novel class detection. A non-parametric Bayesian model is applied so that active learning implementation can cope with the lack of prior knowledge about the new class. Recently, Saha et al. (Saha et al., 2018) detects novel class using instance distribution in decision tree leaves.

For data stream, it is not necessary to have fixed feature space. New feature can appear in mid-stream. And this case is illustrated in Figure 2.2. Figure 2.2.a shows a data stream with two features and two classes (circle and triangle) initially. After some time, a third feature appear in the stream as presented in Figure 2.2.b. These new samples arrive

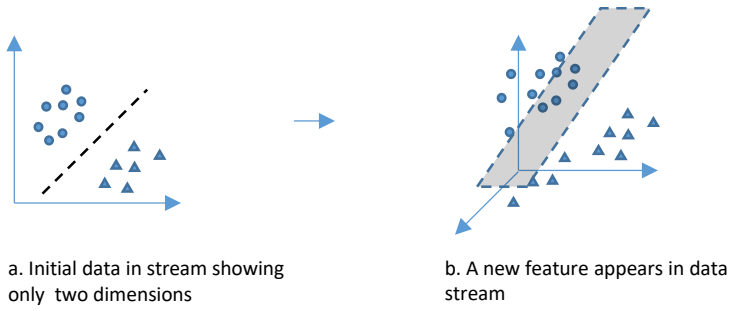


Figure 2.2. Novel features in streaming data

with three dimensions, but the number of classes is still two. The classification model transforms from a linear model in a two dimensional data space to a plane in a three dimensional data space. Masud et al. (Masud et al., 2013) proposed an ensemble framework that contains both novel class and feature drift detection element. Each base learner is equipped with novel class detection. A feature set homogenization technique is applied to deal with evolving features.

2.2 Concept Drift Types without New Class or Feature

This section assumes the data stream with no new feature appearing and no new class being designated, as opposed to the previous secton. Table 2.1 shows the overview of concept drift types. The types are compiled from studies done by Minku et al. (Minku et al., 2010), Gama et al. (Gama et al., 2014), Webb et al. (Webb et al., 2016) and Khamassi et al. (Khamassi et al., 2019). The left most column shows two different scopes of drift detection. Single Drift means that the detection process only looks for the current drift. It does not take previous drifts into consideration, or drift detection without memory. Drift Sequence means that detection process takes multiple historical drifts up to the current drifts into account, or drift detection with memory. The middle column in Table 2.1 shows the criteria of categorization. In Single Drift, the speed of change categorizes drifts based on how fast the classification boundary of underlying model changes. Sudden drift occurs immediately, while Gradual and Incremental drift occur slowly. The distribution of change categorizes drifts based on whether the drift is happening in place, called Fixed Space drift or happening at a new place in the feature space, called Non-fixed Space drift. For Drift Sequence approaches two criteria are considered: Recurrence and Time interval. Recurrence categorizes a series of drifts into Recurrent, where a single drift in the series

repeats itself, and Non-recurrent, where repeating drift is absent. Finally, time Interval

Table 2.1. Overview of concept drift types

Detection Scope	Criteria	Type
Single Drift (Drift Without Memory)	Speed	Sudden/Abrupt
		Gradual
		Incremental
	Distribution	Fixed Space
		Non-fixed Space
Drift Sequence (Drift With Memory)	Recurrence	Recurrent
		Non-recurrent
	Time Interval	Periodic
		Irregular

describes whether a series of drifts occurs in periodically or with an irregular interval.

2.2.1 Speed as Criteria in Categorizing Concept Drifts

The speed of change in data from one data distribution before concept drift to another after concept drift is measured by the number of time steps needed for the change to complete (Minku et al., 2010). A concept drift is complete when the data distribution settled at its final place. A time step can be the arrival of a single sample, a group of samples, or a fixed time interval. The less time steps it takes for the concept to complete, the faster

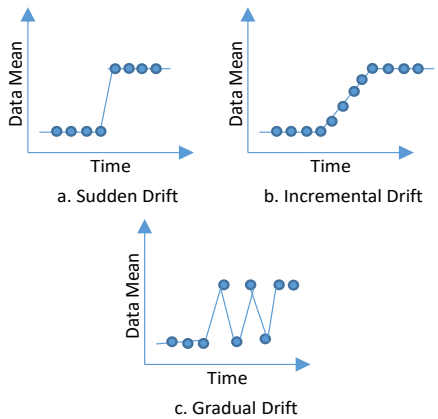


Figure 2.3. Sudden, Incremental and Gradual Drift classified by speed of change

the drift is. If the drift is completed in one time step, it is called a Sudden or an Abrupt Drift. An example of data stream with sudden drift is one-dimensional sensor data for monitoring machine conditions. A fault in a machine part is caused by sudden change in the sensor reading to a new value, illustrated in Figure 2.3.a. If the drift takes more than one time step, it is either an Incremental Drift or a Gradual Drift. The difference between Incremental and Gradual drift is whether there are intermediate samples between the initial and final stage of concept drift. Gradual drift can happen in technology adoption data stream when a new technology emerges. Some consumers will switch to the new technology immediately while others will continue use the old technology, but slowly adopting the new over time. Incremental drift can happen in city demographic data where population shift can happen over several years. Visually, all three types of drift categorized by speed of change is illustrated in Figure 2.3.

Each dot in Figure 2.3 is a time step of a one-dimensional data stream. The drift shown in Figure 2.3.a is the Sudden Drift, where the data mean of the samples abruptly shifts to a higher value. The entire process is complete in one time step. Incremental Drift, shown in Figure 2.3.b, has data mean slowly shifting upwards. After 5 time steps the final data mean settles, completing the drift. The data samples that arrive during the 5 time steps have data mean that are between the initial and final data means. These samples are intermediate samples. When these intermediate samples are absent and the drift is not sudden, then it is a Gradual Drift, shown in Figure 2.3.c. This drift also takes 5 timesteps to complete. However, the data mean changes back and forth between the initial and the final, with no intermediate values.

2.2.2 Data Distribution as Criteria in Categorizing Concept Drifts

This criterion divides concept drift based on whether there is change in the global data distribution after concept drift is completed (Gama et al., 2014). By definition, a concept drift means changes in classification model, resulted from individual class distribution change. However, it is possible that individual class distribution change does not affect the overall global distribution, as in the case of Fixed space concept drift (Hu & Kantardzic, 2022), as shown in Figure 2.4.a. The global data distribution remains the same before and after the drift. However, within the global data distribution, single class distributions indeed may change, which alters the classification model between the two classes. An example of Fixed space concept drift, Bio-reactor data stream can have bacteria growth condition shifts under different production stages, but the condition remains in a fixed range to keep the population alive. A non-fixed space drift alters the global data distribution during the drift process (Hu & Kantardzic, 2022). Online trending topic data stream is a good example of non-fixed space drift (Lee et al., 2011), where a completely new trend can emerge in time. In Figure 2.4.b, the global distribution along with single class distribution changes after the drift. A new group of samples form a dense region

outside the initial global distribution, and it causes changes in the model. This is a non-fixed space drift.

2.2.3 Combinations of Drift Types Using Two Criterias: Speed and Distribution

Since speed and distribution are two independent criteria, the combinations of concept drift types under the two criteria yields six different types of concept drift: Fixed space sudden, fixed space gradual, fixed space incremental, non-fixed space sudden, non-

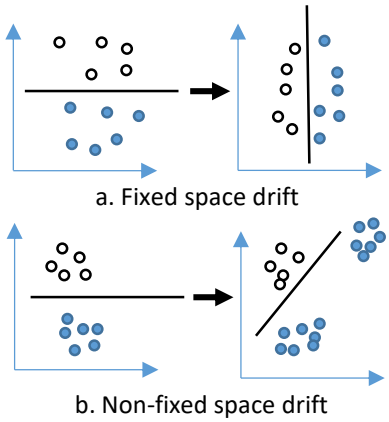


Figure 2.4. Illustration of Fixed space and Non-fixed space drifts

fixed space gradual and non-fixed space incremental (Hu et al., 2020). These types are summarized in Figure 2.5.

Fixed space sudden (FSS) drift is illustrated in Figure 2.5.a. Initially, the linear model between the circle and triangle classes is slanted upwards. Within one time step between time steps 2 and 3, the class boundary shifts to slating downwards. Although

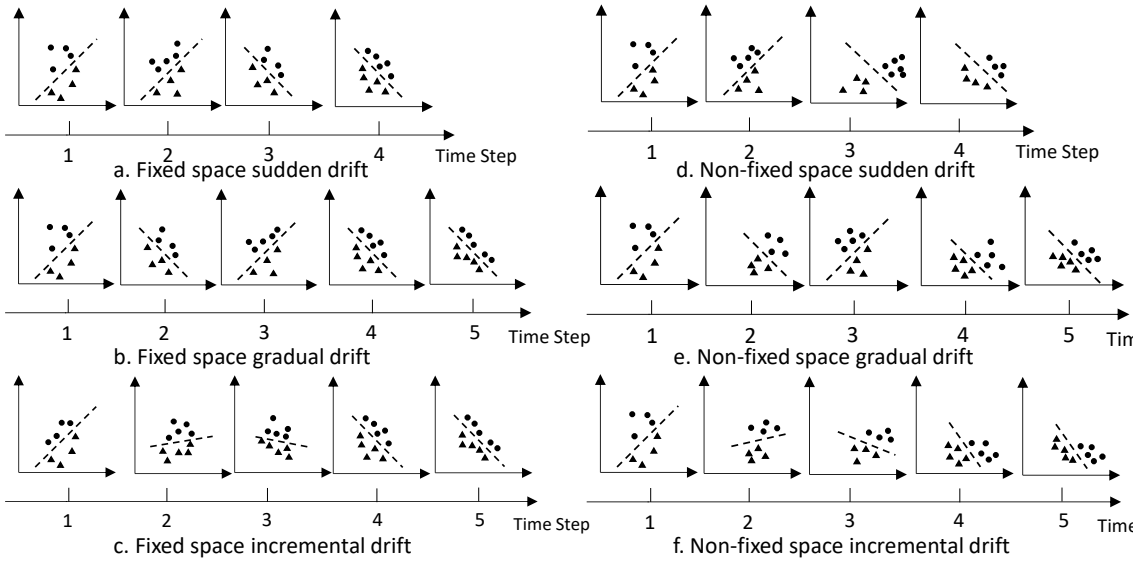


Figure 2.5. Illustration of combination between speed and distribution of change

individual class distributions have changed, the global distribution remains the same. Because the drift completes in one time step and the global distribution is fixed, this is a fixed space sudden drift.

Fixed space gradual (FSG) drift is illustrated in Figure 2.5.b. The stream start with an upward class boundary. After the first time step, new data samples display a downward class boundary, which is again reversed to the previous upward boundary in time step 3. Eventually starting from time step 4, the class boundary settles to be a downward sloped. Since this drift remains in the same global space and takes more than one time step to complete but do not contain intermediate samples, this is a fixed space gradual drift.

Figure 2.5.c shows the fixed space incremental drift (FSI). The drift slowly occurs between time step 1 and 4. In time step 2 and 3, intermediate samples rotates the decision boundary slowly. Eventually the class boundary settles at time step 4. The presence of intermediate samples differentiates this drift from the gradual drift example above.

Figure 2.5.d shows the non-fixed space sudden drift (NSS). During time step 1 and 2, both classes start off with an upward slanted class boundary. Suddenly in time step 3, a sudden shift in distribution of the circle class changed the decision boundary to be downward. In addition to the class boundary change, the circle class distribution changes also changed the shape of the global data space. Hence this drift is called non-fixed space sudden drift.

In Figure 2.5.e the data stream displays non-fixed space gradual drift (NSG). Similar to fixed space gradual drift, the data first drifted to a new distribution and class boundary in time step 2, then reverting the drift in time step 3. After step 3, the data settles at the new boundary and distribution and drift is completed in 3 time steps. Again because there is no intermediate samples, the drift is a gradual drift.

For non-fixed incremental drift (NSI), the illustration is shown in Figure 2.5.f. Similar to illustration above, the drift takes 3 timesteps to complete. The class boundary changed from upward sloped to downward sloped. In addition to the boundary, the global distribution of data also changed. Different from above is the presence of intermediate data in time step 2 and 3. The circle class slowly shifts its distribution to the lower right corner, rotating the decision boundary slowly in the process. Eventually in step 4 the boundary and data distribution settle, completing the drift.

The reason for combining speed of change and distribution of change into six categories is because current concept drift detection methodologies are with different sensitivities to these six types of concept drift (Minku et al., 2010)(Gama et al., 2014)(Webb et al., 2016). Some focus on detecting sudden concept drift, while others focus detecting non-fixed concept drift. These different emphasises of detection methodologies

result in concept drift detection approaches being only suitable to a subset of the six concept drift types. This aspect of concept drift detection is discussed further in Section 2.5.

2.2.4 Recurrence of Drift

When analyzing a sequence of drifts, a concept drift that repeats itself is called a recurrent drift, or cyclic drift. An example of recurrent drift is energy consumption data, where the day/night consumption level repeats. Recognizing a recurrent drift can help stream mining framework save resources. After the first occurrence of the drift, the classification model trained after the drift can be stored for later use. Once the drift occurs again, the same model can be retrieve from storage without training a new model. Visually, this is illustrated in Figure 2.6.

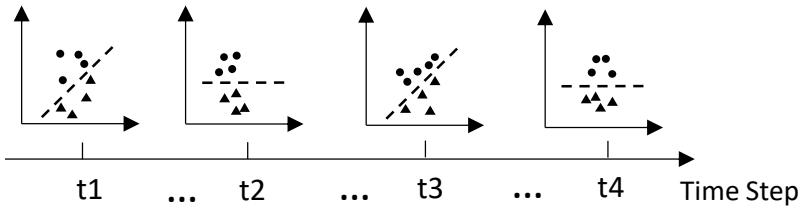


Figure 2.6. Illustration of recurrent drift

The time step in Figure 2.6 shows t1, t2, etc instead of 1, 2, etc because between each drift there are enough time passed such that the drift can be considered complete. Therefore, the figure shows a series of drifts instead of one single gradual drift. The focus is on the relationships and patterns between four concept drifts at time t1, t2, t3 and t4. At time step t2, a sudden drift changed the class boundary from upward sloped in t1 to horizontal. Then at t3, a sudden drift reverts the drift at t2. Again, at t4, the drift of t2 is repeated to bring class boundary to horizontal again. The drift at t4 is a recurrent drift from

t2, therefore this sequence of drifts is a recurrent sequence. Obviously, a non-recurrent drift sequence does not contain drifts that repeat.

2.2.5 Time Interval of Drift

In a sequence of drifts, the time interval between each drift can be the same or different. When each drift occurs exactly the same time apart, then the drift sequence is called a periodic drift. Otherwise the drifts are irregular if they are not appearing in fixed time interval. It is important to recognize that each concept drift in this sequence maybe completely different. An example of periodic drift is sun energy data, where the movement of the sun over seasons is highly periodic. Visually, Figure 2.7 demonstrates periodic drift sequence, showing that the time step between each drift is fixed n.

2.2.6 Combination of Recurrence and Time Interval in Concept Drift

Similar to combining speed and distribution types in a single drift, recurrence and time interval types of a drift sequence can be combined as well. The combinations have four types of drift sequences shown in Figure 2.8. Figure 2.8.a, shows periodic recurrent drift, where a sequence of two types of concept drifts repeat in a fixed time step interval.

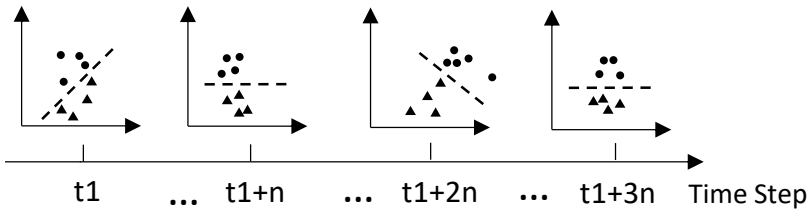


Figure 2.7. Illustration of periodic drift

Figure 2.8.b shows periodic non-recurrent drift, where each drift occurs in a fixed time interval but no drift repeats. An irregular recurrent drift is shown in Figure 2.8.c where

drifts repeat but the interval between each drift are not fixed. Finally Figure 2.8.d shows irregular non-recurrent drift, where each drift doesn't repeat and occurs irregularly.

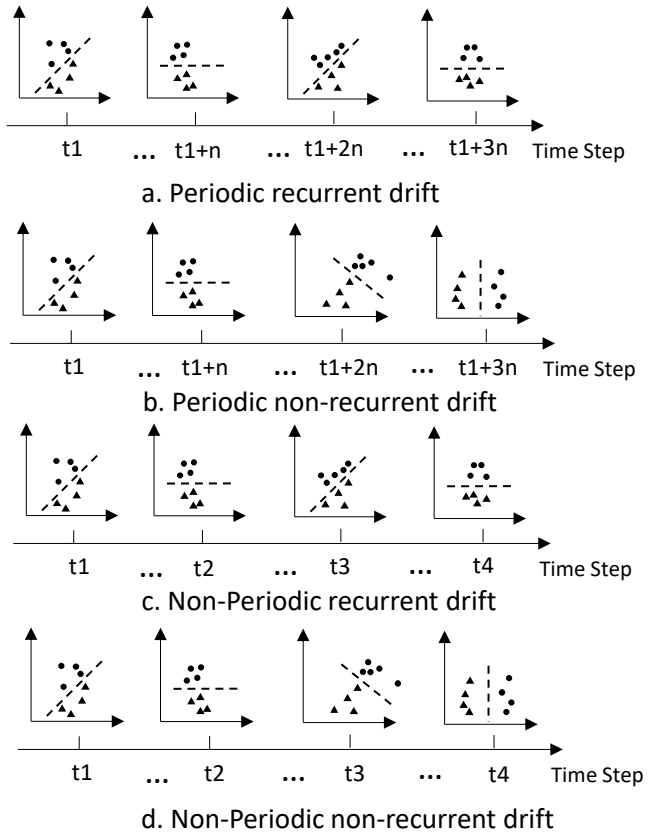


Figure 2.8. Illustration of combination between recurrent and periodic drift

2.3 Detection of Single Drift

The studies on concept drift detection can be roughly divided into two groups (Table 2.2): a) performance-based approach and b) data distribution-based approach. Performance based techniques continuously monitor the sequence of performance metrics, such as accuracy, F-measure, precision and recall; to signal a change, in the event of a significant drop in these values. These techniques require labeled data and therefore are often called supervised approaches (Gama et al., 2014). Data distribution-based approaches

monitor distribution change such as location, density and range. These techniques can use labeled or unlabeled data, and approaches using unlabeled data are called unsupervised approach (Gemaque et al., 2020). The two groups can be further divided based on different detection methods and metrics, each having their pros and cons.

Performance based approaches are able to detect all types of single concept drifts (Hu et al., 2020). Regardless of types, concept drift changes the underlying classification model, which manifests as a change in the performance of the current model. However, performance-based approach requires labeled data instances to obtain accurate measurement. Given the large quantity and speed of real-world data stream, this is not always feasible in real-world applications. In contrast, data distribution-based approaches may not be able to detect all types of concept drift, because they can do so with little to no labels. The two groups of approach are a trade-off between resources and effectiveness: performance-based approaches requires high amount of resources but also can be more effective. Whereas data distribution-based approaches require less resources but may also be less effective.

A compromise between the two groups exists where unlabeled data can be used to measure classification confidence or uncertainty. This measurement shows how confident the classifier is to predict a class label for a particular sample. A change in confidence or uncertainty can signal potential concept drift. Monitoring such measurement is similar to monitoring classification performance and therefore this detection approach will be briefly discussed together with performance-based approach.

Table 2.2. Overview of concept drift detection approaches

Detection Target	Detection Method	Example Studies
Performance Based (Require labelled data)	Statistical Test	DDM (Gama et al., 2004), STEPD (Nishida & Yamauchi, 2007), ECDD (Ross et al., 2012),FHDDM (Pesaranghader & Viktor, 2016), Fisher's Exact Test (de Lima Cabral & de Barros, 2018), …
	Ensemble	ASHT(Bifet et al., 2009), AUE(Brzeziński & Stefanowski, 2011), DDD(Minku & Yao, 2012), DDWM (Sidhu & Bhatia, 2018), KME(Ren et al., 2018), …
	Others	RBM (Jaworski et al., 2017), OHNBC (Astudillo et al., 2016), …
Data Distribution Based (Can use unlabeled data)	Statistical Test	Kolmogorov-Smirnov (KS) Test (Polonik, 1999)(Kifer et al., 2004)(dos Reis et al., 2016)(Spinosa et al., 2007), Multi-kernel (Siahroudi et al., 2018), Multi-component (Liu et al., 2018), …
	Density	OLINDDA (Spinosa et al., 2007), SAND (Haque et al., 2016), Clustering (Kantardzic et al., 2010), GC3(Sethi et al., 2016)(Sethi et al., 2014), …
	Classification Margin	1-norm SVM (Dries & Rückert, 2009), MD3 (Sethi & Kantardzic, 2017), Model Explanation (Demšar & Bosnić, 2018), …
	Others	SOM (Cabanes & Bennani, 2012), One-class Classifier (Krawczyk & Woźniak, 2015), …

2.3.1 Performance Based Approach for Concept Drift Detection

Statistical test can be applied to detect changes in performances. The drift detection techniques based on Statistical Process Control monitor the online trace of error rates, and

detects deviations based on ideas taken from control charts. A significantly increased error rate violates the performance distribution model, and as such is assumed to be a result of concept drift. The basic principle of performance based statistical testing is illustrated in Figure 2.9. The performance of some classifier has a statistical distribution before the drift occurs. After the concept drift, the performance distribution shifted to a lower mean. If the shift is significant enough according to statistical test, then a concept drift is detected.

Gama et al. (Gama et al., 2004) proposed Drift Detection Method (DDM) that detects significant changes in streaming data classifiers' performance. The DDM approach monitors the probability of error at time t as p_t and the standard deviation as (2)

$$s_t = \sqrt{p_t(1-p_t)/i} \tag{2}$$

When, $p_t + s_t$ reaches its minimum value, the corresponding values are stored in pmin and smin. A warning is signaled when $p_t + s_t > p_{min} + 2 \times s_{min}$, and a drift is signaled when $p_t + s_t > p_{min} + 3 \times s_{min}$. An improved version of DDM, called Early Drift Detection Method (EDDM) was proposed by Baena-Garcıa et al. (Baena-García et al., 2006). EDDM's advantage is to be able to detect gradual drift earlier than DDM. The

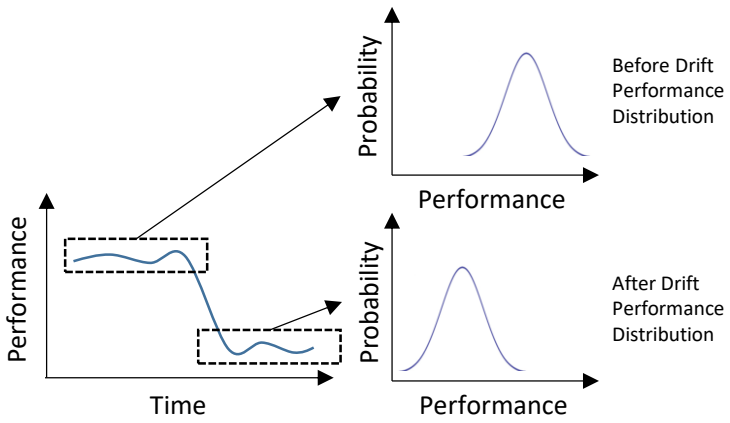


Figure 2.9. Basic principle of performance based statistical testing

EDDM was developed as an extension of DDM, and was made suitable for slow moving gradual drifts, where DDM previously failed. EDDM monitors the number of samples between two classification errors, as a metric to be tracked online for drift detection. It was assumed that, in stationary environments, the distance (in number of samples) between two subsequent errors would increase. A violation of these condition was seen to be indicative of drift. Nishida et al. (Nishida & Yamauchi, 2007) applied statistical test of equal proportions (STEPD) to concept drift detection problem. STEPD computes the accuracy of a chunk C of recent samples and compares it with the overall accuracy from the beginning of the stream, using a chi-squares test to check for deviation. Another approach, proposed by Ross et al. (Ross et al., 2012), utilizes exponentially weighted moving average charts for detecting concept drift (ECDD or EWMA). The moving average is created for the overall performance of the classifier so that any significant deviants from historical normal values can be detected as concept drift. The metric M at time t is updated as per (3)

$$M_0 = \mu_0,$$

$$M_t = \delta \times M_{t-1} + (1-\delta) \times \epsilon_t \quad (3)$$

Where, μ_0 and σ_0 are mean and standard deviation obtained from the training data, by random sampling. The error rate at time t is given by ϵ_t , θ is the acceptable deviation in terms of number of standard deviations from the mean and δ is the forgetting factor which controls the effect of previous data on the current sample. In (3), a new metric M at time t is updated by multiplying previous error rate with forgetting factor δ and adding current error rate of time t multiplying by $1-\delta$. Finally, the criterion for concept drift detection is given by (4)

$$if\ M_t - \mu_0 > \theta \times \sigma_0, Drift\ Detected \quad (4)$$

Frías-Blanco et al. (Frías-Blanco et al., 2015) proposed an incremental framework called Hoeffding Drift Detection Methods (HDDM) that uses Hoeffding Bounds as the statistical test for concept drift detection. The approach tracks the moving average of classifiers' performance. To detect concept drift, Hoeffdings' inequality was applied to detect significant changes in the moving average. Pesaranghader et al. (Pesaranghader & Viktor, 2016) proposed using Hoeffding's inequality as the statistical test for detection called Fast Hoeffding Drift Detection Method (FHDDM). The test compares the maximum overall probability of a correct prediction and the most recent probability of a correct prediction. Similar to STEPD, if the stream is non-drifting, then the two probably should be very similar. Drift is detected if there are significant difference between the two values. Yu and Abraham (Yu & Abraham, 2017) proposed a framework that utilize a hierarchical set of hypotheses testing to accurately detect concept drift. The first layer detects concept drift by tracking classifiers' previous and current performance and applying Linear Four Rates test to detect significant changes. The second layer uses output of the first layer and employs permutation test procedure to validate detection from first layer. Cabral et al. (de Lima Cabral & de Barros, 2018) proposed drift detection method using Fisher's Exact Test, another statistical hypothesis testing tool. An efficient implementation of Fisher's Exact Test enables the test to be applied to streaming data. Results show the test is superior to previous approaches such as STEPD (Nishida & Yamauchi, 2007), ECDD (Ross et al., 2012) and FHDDM (Pesaranghader & Viktor, 2016). Wang et al. (Wang et al., 2018) proposed Multiscale Drift Detection Test approach that employs resampling and paired

student t-test. The approach emphasizes on lowering computation cost of concept drift detection.

Another common strategy for performance-based drift detection uses ensemble learning. The basic principle for this strategy is illustrated in Figure 2.10. Each ensemble is composed of multiple diverse base classifiers. Concept drift affect each base classifier differently. Either the overall ensemble performance or individual base classifier performance can be monitored for concept drift detection. In Figure 2.10, the ensemble contains classifier A, B and C with varying accuracy and weight assigned. After the concept drift, all A, B and C see significant decline in accuracy, which triggers drift detection. If classifier C falls below performance threshold, a new classifier D is trained to replace C and all classifiers' weights are adjusted as well. Also, ensemble-based approach often tie concept drift detection with concept drift handling. Meaning by the time concept drift is detected, the ensemble approaches often have the mechanism to adjust the models to the change immediately.

Wang et al. (Wang et al., 2003) proposed a weighted ensemble that assign weights to each base classifier based on their current performance. Bifet et al. (Bifet et al., 2009) utilized two different bagging approach for their proposed windowed ensemble framework: ADWIN Bagging and Adaptive-Size Hoeffding Tree (ASHT) Bagging. A windowed approach is where a certain number of data samples from stream is grouped together, forming a window, or chunk, of grouped data samples. All activities of the framework, including concept drift detection, model learning, model evaluation, etc., are all performed within each window or chunk. Bifet et al.'s bagging approach can prune under-performing classifier and thus makes the ensemble more efficient in real-world application. In case

change is present, the window is shrunk and vice-versa. Whenever two large enough sub windows exhibit distinct averages of the performance metric, a drift is detected.

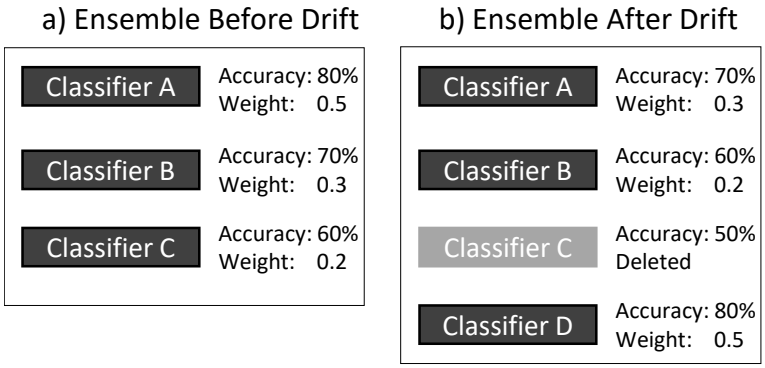


Figure 2.10. Basic principle of performance-based drift detection using ensemble learning

Brzeziński and Stefanowski (Brzeziński & Stefanowski, 2011) proposed an Accuracy Updated Ensemble (AUE) framework that uses Hoeffding Trees as base classifiers. Each base classifier incrementally adapts to concept drift while the entire ensemble also weighs each classifier based on their accuracy. Thus, AUE is able to react to both sudden and gradual change in classifier performance. Minku et al. (Minku & Yao, 2012) proposed Diversity for Dealing with Drifts (DDD). DDD acknowledges the diversity of different types of concept drift and tries to incorporate a variety of base classifiers in their ensemble. Each base classifier is assigned a level of diversity and the ensemble maintains multiple levels of diversities to detect various types of concept drift. An online bagging approach was used to prune the base classifiers. Dehghan et al. (Dehghan et al., 2016) proposed an approach that processes samples one by one and monitors the ensemble's error distribution. A measurement called Number and Distance of Errors (NDE) was created to describe the error of each sample's classification result. If the recent NDE is significantly different from overall NDE, then a concept drift is detected. Sidhu and

Bhatia (Sidhu & Bhatia, 2018) presented diversified dynamic weighted majority (DDWM). Similar to DDD, classifiers are grouped into two sets of different levels of diversity. When concept drift occurs, base classifier in either ensemble is removed if its accuracy falls too low. A new base classifier is added in either ensemble when its accuracy is comparable or better than the global prediction accuracy. Khamassi et al. (Khamassi et al., 2019) proposed an ensemble called EnsembleEDIST2 approach that uses Error Distance Approach for Drift Detection and Monitoring (EDIST2) to track detect concept drift by tracking ensemble's performance. EDIST2 is a concept drift detection based on the Khamassi et al.'s earlier work (Khamassi et al., 2015). The ensemble utilizes three diversity methods to benefit from their advantage and limit their disadvantages. Nikzad-Langerodi et al (Nikzad-Langerodi et al., 2018) proposed an ensemble of partial least square (PLS) models for applications in Melamine resin production. A committee disagreement measurement is calculated, and changes are detected using PageHinkley (PH) statistic on this metric. The study also explored supervised and unsupervised strategies using this framework.

Mahdi et al. (Mahdi et al., 2018) integrates the entropy drift detection with ensemble classifier. Using information entropy as concept drift detector is first proposed by Vorburger et al. (Vorburger & Bernstein, 2006), then improved using a dynamic sliding window by Du et al. (Du et al., 2014). Mahdi et al. calculates entropy for each base classifier from a fixed-sized block of data. The classifiers are added or removed based on whether their entropy is of a desirable level. Ren et al. (Ren et al., 2018) combined both performance-based approach and distribution-based approach for concept drift detection. Their framework Knowledge-maximized ensemble (KME), uses both labeled and unlabeled data to maximize information on knowledge of the current concept, including

both classifier performance and sample distribution. This approach is included in the performance-based approach section because weighting the KME ensemble using classifier performance is a major part of the framework. The approach still heavily relies on labeled data. Krawczyk and Cano (Krawczyk & Cano, 2018) modified existing ensemble voting mechanism by allowing base classifiers to abstain from contributing to final decision. Each classifier's confidence level is monitored sample by sample and only classifiers with confidence over certain threshold is selected. The threshold is dynamic chosen based on current data stream environment. The abstaining option enhance the ensemble's ability to deal with noisy data stream.

Some ensemble approaches also try to address the cost of labeling by using partial labeled data. Ditzler et al. (Ditzler & Polikar, 2011) applied semi-supervised support vector machine to stream data mining problems. Their ensemble is trained, tested and updated using both labeled and unlabeled data. Ahmadi & Beigy (Ahmadi & Beigy, 2012) applied majority voting, previously used for fully labeled classification problems, to the ensemble of partially labeled semi-supervised classifiers. Hosseini et al.. (Hosseini et al., 2016) proposed an ensemble semi-supervised classification framework that is able to handle concept drift and partial labeling. Each of their classifier represents a single concept. The classifiers are updated using the latest partially labeled data.

Several performance-based approaches use neither statistical test nor ensemble learning. Rutkowski et al. (Rutkowski et al., 2014) proposed a new decision tree construction method for stream data mining. The study derived a new splitting criterion based on misclassification errors. When combined with the Gini index, their decision tree was able to achieve high prediction accuracy in a concept drifting stream. Du et al. (Du et

al., 2014) calculated information entropy from an adaptive sliding window, which is then used for concept drift detection. The sliding window is dynamically determined, and the approach is able to detect the exact moment for retraining classification model. Jaworski et al. (Jaworski et al., 2017) applies Restricted Boltzmann Machine (RBM), a type of neural network, to detect concept drift by evaluating free energy and reconstruction error from RBM. Another neural network-based concept drift detection is proposed by Lobo et al. (Lobo et al., 2018), which uses Evolving Spiking Neural Networks. Astudillo (Astudillo et al., 2016) proposed using Online Histogram-based Naïve Bayes Classifier (OHNBC) and the change in classification performance to detect concept drift. OHNBC has the advantage of dealing with data stream that have label and unlabeled data instances mixed, with training, testing and deployment of classification model interleaved.

When labels are not available, it is possible to estimate classifier performance by measuring classification confidence or classification uncertainty. Lughofer et al (Lugofer et al., 2016) explained such concept drift detection scheme in their study. Classifier behavior is monitored and a modified version of Page-Hinkley test is employed to detect statistically meaningful changes in classification uncertainty. The study proposed two variants of frameworks to deal with both semi-supervised and unsupervised classification. Kim & Park (Kim & Park, 2017) used probabilistic estimation on classification result to detect concept drift in data streams with limited or no access to labeled data. Random samples are selected, and a classification confidence vector is calculated. A windowed monitoring approach is then used to detect significant changes in confidence for potential concept drift.

2.3.2 Comparisons among Performance Based Approaches for Concept Drift Detection

Pesaranghader et al. (Pesaranghader & Viktor, 2016) summarized comparison of experimental results between several approaches: FHDDM, DDM (Gama et al., 2004),

Table 2.3. Comparison of Experimental result compiled by (Pesaranghader & Viktor, 2016)

Algorithms / Data Sets / Detection Approach		Electric Market (Zliobaite, 2013)		Airline (Ikonomovska, 2011)		Poker Hand (Cattral & Oppacher, 2017)	
		# of Drifts	Accuracy	# of Drifts	Accuracy	# of Drifts	Accuracy
HT	**FHDDM (Pesaranghader & Viktor, 2016)**	**77**	**84.38%**	**339**	**65.66%**	**1557**	**76.45%**
	DDM (Gama et al., 2004)	**169**	**84.41%**	**14**	**65.29%**	**1046**	**72.74%**
	EDDM (Baena-García et al., 2006)	**191**	**84.91%**	**54**	**65.06%**	**4806**	**77.30%**
	ADWIN (Bifet et al., 2009)	**110**	**83.40%**	**341**	**65.25%**	**2373**	**74.56%**
	HDDMA-test (Rutkowski et al., 2014)	**210**	**85.71%**	**88**	**64.99%**	**2565**	**76.40%**
	HDDMW-test (Rutkowski et al., 2014)	**117**	**85.06%**	**652**	**65.02%**	**2211**	**77.11%**
NB	**FHDDM**	**96**	**82.69%**	**297**	**66.44%**	**1660**	**76.30%**
	DDM	**143**	**81.18%**	**13**	**65.33%**	**433**	**61.97%**
	EDDM	**203**	**84.83%**	**23**	**65.18%**	**4863**	**77.48%**
	ADWIN	**128**	**81.63%**	**300**	**66.79%**	**2453**	**74.60%**
	HDDMA-test	**211**	**84.92%**	**72**	**67.22%**	**2615**	**76.48%**
	HDDMW-test	**132**	**84.09%**	**620**	**65.34%**	**2312**	**77.11%**

EDDM(Baena-García et al., 2006), ADWIN(Bifet et al., 2009) and HDDM(Frías-Blanco et al., 2015). The comparison contains two version of HDDM with two different statistical test metric: A-test and W-test. The classifiers used are Hoeffding Tree (HT) and Naïve

Bayes (NB). The dataset used are Electric Market (EM) (Zliobaite, 2013), Airline (Ikonomovska, 2011) and Poker Hands (Cattral & Oppacher, 2017). Table 2.3 shows the number of drifts detected and average accuracy of underlying classifiers under three datasets.

Several observations can be made from Table 2.3. First, no one detection algorithm has the best result for all three datasets. This means that even performance-based approach can detect all types of drift, their performance is still depended on nature of the data stream itself. Second, high number of concept drift detected does not necessarily mean high accuracy. For instance, for the airline dataset using HT as classifier, the highest number of drifts detected is 652 by HDDMW-test and the lowest number of drift detected is 14 by DDM. However, DDM has the highest average accuracy in this experiment whereas HDDMW-test scores in the middle. It means that a lot of these detections are false positive. This may be due to the parameter set by the experimenter being very sensitive. Third, the accuracy gain within each dataset is limited. The majority of the results shows that performance difference between lowest accuracy and highest accuracy is less than 3%. Exceptions are Poker Hand with HT and Poker Hand with NB, with differences being 4.56% and 15.51% respectively. This further show that performance-based approach is still heavily affected by the characteristic of the stream data set.

2.3.3 Data Distribution Based Approach for Concept Drift Detection

Data distribution-based approach has the advantage of being able to process both labeled and unlabeled data. When this group of approaches work with labeled data, their detection performance can be on par with that of performance based approach (Dries &

Rückert, 2009)(Sethi & Kantardzic, 2017). When working with unlabeled data, however, each approach is best suitable for a subset of concept drift types.

Statistical test is used in data distribution-based approach. Instead of tracking classifier performances, here the test is applied to track significant data distribution change. This is illustrated in Figure 2.11 using two-dimensional data (X, Y). The actual underlying class boundary of the demonstrated two-dimensional data would be unknown if class labels are not available. However, there are changes in the probability distribution of the two features before and after the drift, as shown in Figure 2.11.a and 2.9.b. If such features pass some statistically test to be significant different, then a concept drift can be detected without label. Polonik (Polonik, 1999) generalized Kolmogorov-Smirnov (KS) test to use beyond simple one-dimensional data. KS test determines whether two distributions from two sets of samples are equal. When applied in streaming data, data samples from two different time spans have their respective probability distributions D and F. If D does not equal to F according to KS test, then it is possible that concept drift has occurred. Kifer et al. (Kifer et al., 2004) applied KS test for concept drift detection. Glazer et al. (Glazer et al., 2012) applied KS test for detection of change of high-density area in high dimensional data. The study modified classic minimum-volume set (MV-set) estimators for density estimation and enables KS test to be applied to high dimensional data. The author also noted that change in high density area is directly related to concept drift detection in streaming data. Sobolewski and Wozniak (Sobolewski & Wozniak, 2013) proposed a KS test concept drift detection framework with that can work on unlabeled data stream. The approach acknowledges that concept drift that does not change the global distribution cannot be detected. This study thus demonstrated that KS test drift detector, when applied

to unlabeled data, is not suitable for detecting fixed space drift. dos Reis et al. (dos Reis et al., 2016) modified KS test to be able to perform incrementally. Traditional non-incremental KS test requires O(NlogN) whereas the new proposed test requires O(logN). The fast test is suitable for stream data with big volume and still produce the same result as traditional KS test. However, because the test is unsupervised, traditional KS test limitations on unlabeled data still applies.

Besides using KS test, Song et al. (Song et al., 2007) applied kernel density to detect concept drift. This study identifies suitable kernel width using expectation maximization algorithm. A density test was performed on each data samples to check if it is from the same underlying data distribution. Shaker (Shaker & Lughofer 2014) applied an extended

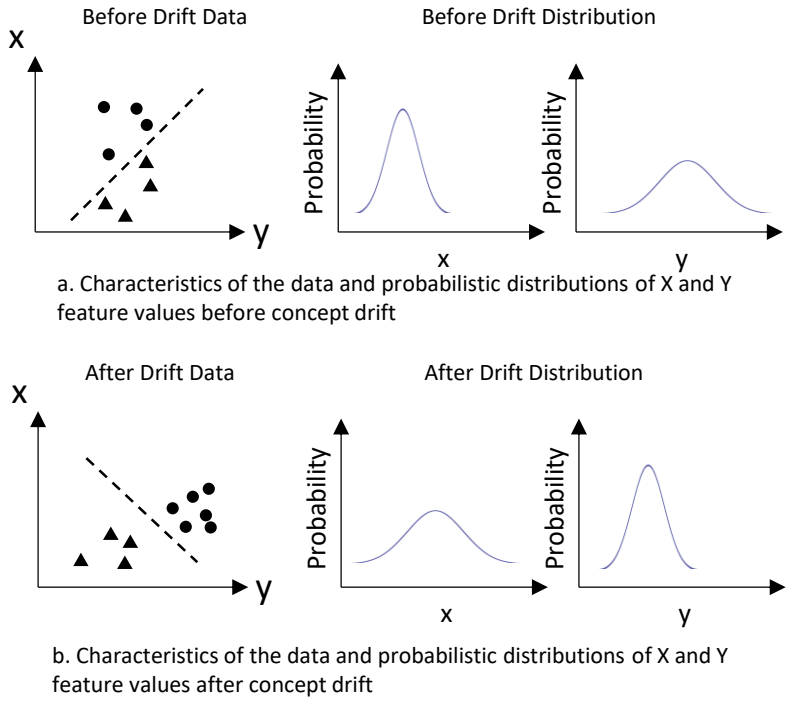


Figure 2.11. Basic principle of data distribution based statistical testing

version of Page-Hinkley test to detect and quantify concept drift for regression. An

adaptive forgetting factor are integrated based on intensity of the drift, and a local forgetting factor are used to address different drift intensity in different local regions of feature space. Siahroudi et al. (Siahroudi et al., 2018) computes multiple kernels of the data space and specifying class boundary using combined kernels. A concept drift is detected if new samples appear outside existing class boundary. Lee and Magoules (Lee & Magoules, 2012) utilized correlation information of value distribution in a windowed detection approach. Faithfull et al. (Faithfull et al., 2019) applied an ensemble of univariate change detector to a multivariate change detection problem. The ensemble outperforms pure multivariate approaches in their experiments. Liu et al. (Liu et al., 2018) proposed a three-component, statistical-test based framework for concept drift detection. The first component uses K-nearest neighbours to construct data subspaces for density estimation. The second component applies a distance function that accumulates density discrepancies in each subspace and calculate overall differences. Third component uses statistical test on density discrepancies to determine the confidence interval of concept drift occurring.

Statistical test is best suited for non-fixed space drift (Sobolewski & Wozniak, 2013) as illustrated in Figure 2.12, which shows the linear classification model rotates in a fixed data space. The overall data distribution shows no change. Since label is unavailable, it is impossible to track individual class's distribution change. Therefore, statistical test fails for this case. Also, studies using KS test may be better suited for quicker drift than slower. Since KS test drift detector comparing distributions of data from two different time interval, the sensitivity of detection is determined by length of the time interval. For a very slow gradual or incremental drift, short interval KS test will not be able to detect significant difference between consecutive intervals.

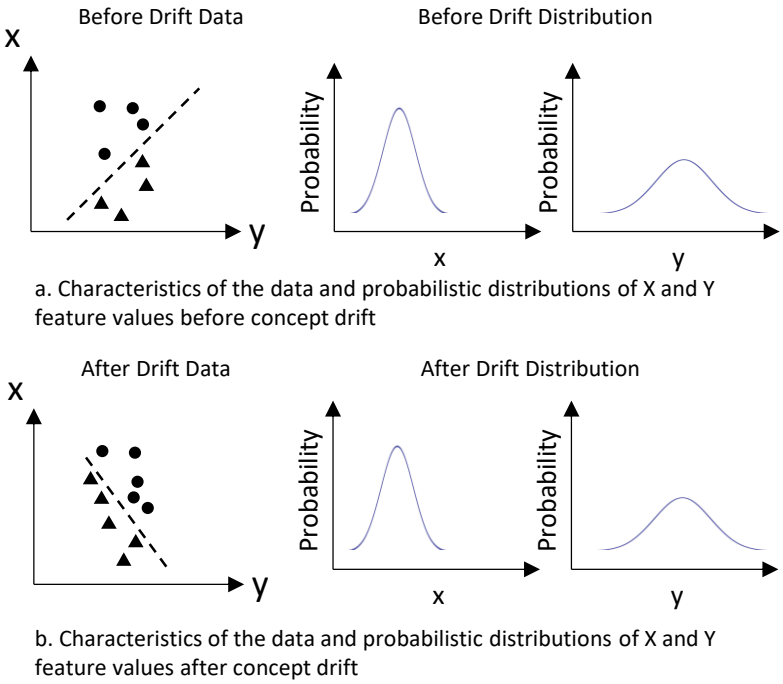


Figure 2.12. When distribution based statistical testing fails

Another group of data distribution-based method look for changes in dense regions of the data. These methods are capable of identifying uncertain suspicious samples, which need further evaluation. They define an additional 'Unknown' class label to indicate that these suspicious samples do not fit the existing view of the data (Spinosa et al., 2007). Clustering and outlier-based approaches are popular implementation strategies for detecting novel patterns, as they summarize current data and can use dissimilarity metrics to identify new samples (Kantardzic et al., 2010). Only changes in dense regions are considered. Changes outside the dense area has less impact on the overall classification performance. These changes can be considered as outliers appearing in a data stream; thus, they don't represent a concept drift. A change in the dense area, in contrast, involves much larger number of samples. The impact on performance is therefore significant. This process

is illustrated in Figure 2.13. The few triangles class that appears above the class boundary in time step t_2 is considered outlier since the impact on class boundary is small. Later in time step t_3, the same region becomes filled with the triangle class. The appearance of such dense region has a large impact on classification performance and should be considered a candidate for concept drift.

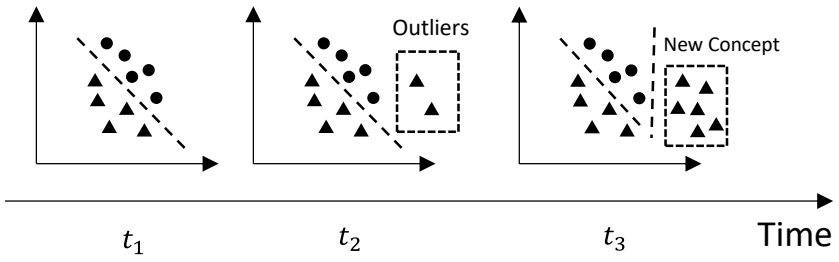


Figure 2.13. Basic principle of data distribution-based density monitoring

There are several approaches to monitor dense area of a data stream. The first one is to use clustering. Lazarescu et al. (Lazarescu et al., 2004) uses a multi-windowed approach with clustering for concept drift detection. The clusters were constructed to describe each current concept. When concept drift occurs, new clusters constructed after the drift will be significantly different from existing cluster and thus the drift can be detected. Spinosa et al. (Spinosa et al., 2007) applied K-means clustering algorithm for concept drift detection in a framework named n OnLIne Novelty and Drift Detection Algorithm (OLINDDA). K clusters were initially generated by the K-means algorithm. The overall arithmetic mean of between the initial cluster centroids were calculated. As new sample arrives, new samples are clustered into a candidacy cluster. If the mean distance between centroid of this candidacy cluster and initial clusters are smaller than the initial mean distance, then this candidacy cluster is considered valid, and no concept drift occurred. Otherwise, the new candidacy cluster forms a new concept outside the initial K-

clusters, forming a new dense data region. Kantardzic et al. (Kantardzic et al., 2010) proposed a framework that work with partially labeled data stream. Similar to OLINNDA, the framework creates initial clusters at beginning of the data stream. The centroid and radius of each cluster is remembered by the framework. New data samples is said to belong to existing clusters if it is within the radius from some existing centroids. For samples that are not within existing cluster, the framework will try to cluster them. If a new cluster emerges from these samples, it means a new dense region appeared and concept drift is detected. Haque et al. (Haque et al., 2016) proposed SAND framework, discussed in section 2 as novel class detector, is also used for concept drift detection. The ensemble created clusters and use these clusters to determine existing data regions. If a new cluster appears, the new dense region can be a new class or a new concept, depends on labels of the samples. Masud et al. (Masud et al., 2010) proposed a framework that is similar to SAND, where K-means clusters were used initially to create K clusters. New samples outside of existing clusters are counted as outliers. Similar to (Kantardzic et al., 2010), if the outliers form a new cluster, then either a concept drift or a novel class is detected.

Tu and Chen (Tu & Chen, 2009) proposed a framework that has an online component that map each data samples to a grid in the data space, and an offline component that compute grid density. Clusters are constructed based on the calculated density of each grid. A density decaying mechanism was applied to forget older samples so that new dense grid can be discovered. Sethi et al. (Sethi et al., 2016) applied the principle of the grid to concept drift detection in the Grid density-based Clustering for Classification of streaming data with Concept drift (GC3) framework. The framework detects concept change when a new grid becomes dense, or an existing grid is no-longer dense because of forgetting

mechanisms. The detection part of the framework can work without labels. An improved ensemble-based grid density framework was proposed by Sethi et al. (Sethi et al., 2014) to tackle concept drift in both spatial and temporal component of the data stream. The grid initially maps out the special characteristics of the data space using grid density clustering. A set of base classifiers were trained on each cluster so the temporal change within the cluster can be monitored as well, with the help of a few labeled samples within each cluster.

When labels are unavailable, clustering approach will not be able to generate clusters for each class individually. For stationary drift, when the global data distribution does not change, no new clusters will be generated, and no concept drift can be detected

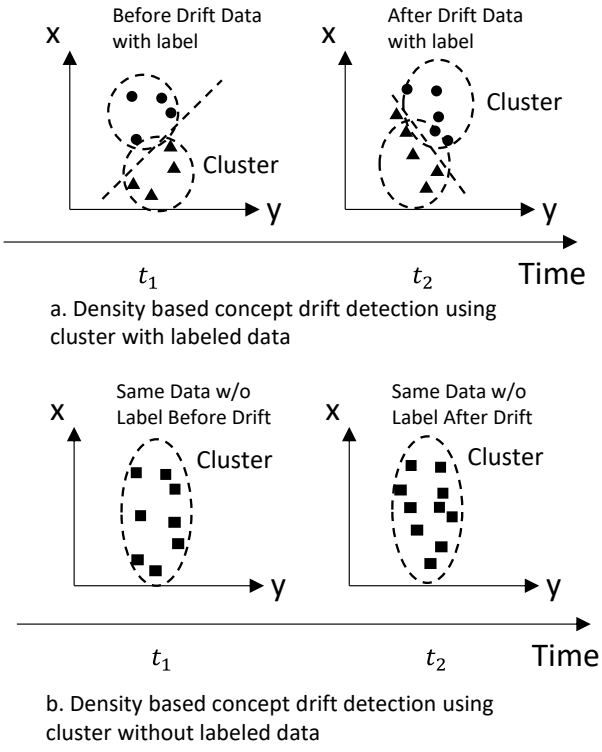


Figure 2.14. When data distribution-based density monitoring fails

using the clustering approaches. Therefore, in an unlabeled data stream these approaches are, hypothetically, best suited for non-stationary drift. Figure 2.14 demonstrated this limitation by showing a fixed space drift. When labels are available, as shown in Figure

2.14.a, clusters are generated for each class in the data and there is clearly a change in cluster distribution before and after concept drift. When labels are not available in Figure 2.14.b, clusters can only be generated on the global data distribution and there shows no change before and after. The clustering approach cannot detect such change.

Several studies use the underlying classification model's characteristics for concept drift detection. Dries et al. (Dries & Rückert, 2009) proposed three approaches in their study: a) density estimation on binary representation of the data, b) measures average margin of 1-norm SVM and c) average error rate generated by SVM. The study shows that the SVM approach (second approach) has the best precision and recall. The linear SVM creates a margin between two supports. Concept drift can be detected if there are significant changes between the margins. Sethi et al. (Sethi & Kantardzic, 2017) further improve the margin approach by extending it to partially labeled data stream. The proposed framework Margin Density Drift Detection (MD3) first trains an initial SVM on a portion of the data stream. New data that arrives between the existing SVM's support are considered critical points. The density of critical points is given by (5)

$$\rho = \frac{\#\ of\ samples\ within\ margin}{\#\ of\ total\ samples} \tag{5}$$

The assumption is that if the class boundary does not change, then the density of the critical points should remain the same over time. Concept drift is detected when a significant change in the density happens. This is illustrated in Figure 2.15. Initially in Figure 2.15.a, two classes of samples are separated by a horizontal decision boundary, with margin density 0.4 (2 critical points within margin divided by 5 total samples) and 0.5 (3 critical points within margin divided by 6 total samples) for each class respectively. After

concept drift occurs in Figure 2.15.b, the decision boundary changed, as a result, the margin density changes to 0.8 and 0.66 respectively. After retraining SVM, the new decision boundary reflects post-drift reality in Figure 2.15.c, and the margin density returns to normal value of 0.6 and 0.5 respectively. Demšar and Bosnić (Demšar & Bosnić, 2018) computes multiple model explanations, which is composed of attribute-value contributions for prediction outcomes, for a given classifier. Concept drift is detected if significant changes occurs in the composition of these attribute-value contributions.

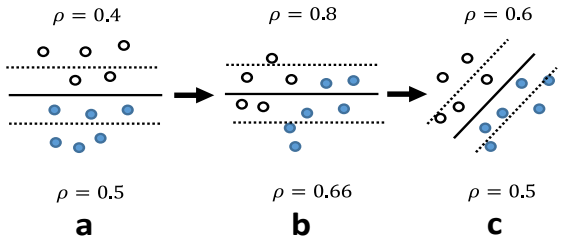


Figure 2.15. Illustration of Margin Density drift detection

When applying approaches that uses classification characteristics to unlabeled data, concept drift can be detected only if changes occur close to the class boundary. This is illustrated in Figure 2.16. If a new concept appears far away from the boundary such as

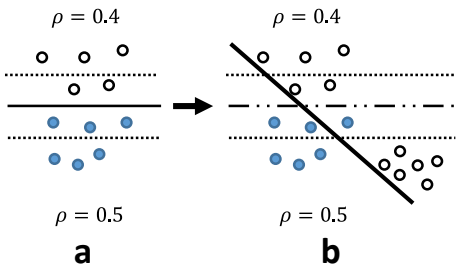


Figure 2.16. Margin Density failed to detect concept drift

those in non-stationary drift shown in Figure 2.34.b, the margin density will not change. The number of samples inside the margin are the same between Figure 2.16.a and 2.16.b. Therefore, this approach is best suited for stationary drift. For speed of change, margin density approaches such as (Dries & Rückert, 2009) (Sethi & Kantardzic, 2017) may not be suited for very slow drift as the density is calculated, depending on window size. Since the density is calculated window by window, for slow drift each consecutive window might not have significant change between each other. A memory of historical density measure is needed to change detection for very slow drifts.

Several studies use other mathematical constructs or algorithms for concept drift detection. Da Costa et al. (da Costa et al., 2017) proposed Multidimensional Fourier Transform (MDFT). The framework quantifies variations in data spaces using Shannon's

Table 2.4. Comparison of Experimental result compiled from (Sethi et al., 2016) and (Sethi & Kantardzic, 2017)

Datasets / Algorithm	EM (Zliobaite, 2013) with weighted F-score	MAGIC (Bock et al., 2004) with weighted F-score
GC3(Sethi et al., 2016)	0.7333	0.9627
Clustering(Kantardzic et al., 2010)	0.643	0.774
SVE (Street & Kim, 2001)	0.688	0.789
WE (Wang et al., 2003)	0.564	0.655
Datasets / Algorithm	**EM (Zliobaite, 2013) with Accuracy**	**Covtype (Blackard & Dean, 1999) with Accuracy**
MD3-SVM(Sethi & Kantardzic, 2017)	66.9%	71.3%
MD3-RS(Sethi & Kantardzic, 2017)	67.7%	75.4%
ECDD (Ross et al., 2012)	68.4%	74.2%
HDDDM (Ditzler & Polikar, 2011)	66.4%	74.9%
L2F3(Krawczyk & Woźniak, 2015)	74.04%	75.76%
IOCSVM(Li et al., 2009)	70.42%	71.08%

and Von Neumann's Entropies. In an unlabeled stationary drift environment, the global variations of data do not occur. This approach is not suited for stationary drift. Cabanes and Bennani (Cabanes & Bennani, 2012) proposed an unsupervised concept drift detection framework that utilize self-organizing map (SOM). In the study the data stream is divided into regular intervals (windows), and a SOM is constructed for each interval. Based on SOM, the density and variability of the data is computed to describe the neighborhood of constructed SOM. These measurements are compared between each interval and concept drift is detected if significant changes happen. Depends on parameters, this approach may or may not be sensitive enough for stationary drift or slow drift. However, SOM should be able to detect sudden and non-stationary drift easily. Krawczyk and Woźniak (Krawczyk & Woźniak, 2015) applied weighted one class SVM classifier for concept drift detection, inspired by previous one-class data stream classification framework by Zhang et al. (Zhang et al., 2009) and Liu et al. (Li et al., 2009). Since one-class classifier predicts whether a data sample belongs to a certain class, new samples are tested by each classifier to see whether they fit into existing concept. If a significant number of samples do not fit into existing concept, then there is a potential concept drift in the data stream. This approach is best suited for non-stationary drift. If all new samples appear within existing one-class classifier, then this approach will fail to detect concept drift.

2.3.4 Comparisons among Data Distribution Based Approaches for Concept Drift Detection

To evaluate performance of data distribution-based approach, several surveyed approaches' experimental results on benchmark dataset are compiled. First three columns of Table 2.4 show the results compiled from (Sethi et al., 2016), comparing drift detection

approach GC3 (Sethi et al., 2016) (grid density based) to Clustering (Kantardzic et al., 2010) (clustering based), SVE (Street & Kim, 2001) (performance ensemble based) and WE (Wang et al., 2003) (performance ensemble based). Next 3 columns of Table 2.4 also show the results compiled from (Sethi & Kantardzic, 2017) and (Krawczyk & Woźniak, 2015), comparing MD3 (Sethi & Kantardzic, 2017) (margin density based) to ECDD (Ross et al., 2012) (performance moving average), HDDDM (Ditzler & Polikar, 2011) (semi-supervised ensemble), L2F3 (Krawczyk & Woźniak, 2015) (one-class classifier), IOCSVM (Li et al., 2009) (one-class SVM). All results use labeled or partially labeled data. When labels are available, data distribution-based detection approaches can have performance on par or better than performance-based detection approaches. This is evident in the comparison among GC3, a distribution-based approach, SVE and WE, both performance-based approach. It is also clear the L2F3 and IOCSVM, both one-class based drift detection suited for non-fixed space drift, outperform MD3, margin density-based detection suited for fixed space drift. This may be because EM and Covtype has more non-fixed space concept drift than fixed space concept drift. More evidence of this can be obtained by examining GC3 results under EM, which has the highest weighted F-score among the four approaches. GC3 is suitable for non-fixed space concept drift as it is a grid density-based approach. The reason GC3 scores the best may be because EM's concept drift is mainly of the non-fixed space type.

2.4 Implications of "No Free Lunch Theorem" in This Dissertation

A simple explanation for the "No Free Lunch Theorem" is that a complex problem often does not have one silver-bullet solution that solves everything (Ho & Pepyne, 2002). The theorem applies to the complex problem of detecting and handling concept drift in data

stream classification. Therefore, this dissertation does not attempt to find a comprehensive solution that tries to "solve" concept drift. Instead, the dissertation looks for improvements in separate aspects of data stream mining: from lower the cost of data stream mining, to detect different types of concept drifts, to visualize changes in data stream for data stream mining explanations. Each investigated aspect has its own unique situations and challenges. This dissertation thus weighs different approaches mentioned above and choose the most applicable strategies. This process provides the foundations for discussion in the following chapters.

CHAPTER 3. SELECTING SAMPLES FOR LABELING IN UNBALANCED STREAMING DATA ENVIRONMENTS[2]

In a stream data environment, time is an important constraint. The data mining process needs to keep up with high volume of incoming data in real time. To complicate the problem at hand, sometimes a process called concept drift will occur in a streaming data environment, in which the data's distribution, classification or association will change overtime (Widmer & Kubat, 1996). In this case the static data mining model might not be suitable for the drifted data and for classification purpose a new classifier should be trained every time there are significant drifts in the data stream. To train the new classifier the new incoming data sample needs to be labeled. However, labeling may be time consuming and very expensive. Therefore, we want to reduce the amount of streaming data points to be labeled as much as possible.

For a more or less evenly distributed data set in which every class has about the same amount of data points, previous work has shown sampling 10% of data for labeling is sufficient to have a new classifier trained (Widmer & Kubat, 1996)(Kantardzic et al. 2010). The scenario is different in an extreme unbalanced data set where the minority class occupies only 1-10% of the entire data samples. Potentially to obtain every minority sample

[2] This chapter has been published at Hu et al, 2013

one needs to sample 99 majority data. To achieve a meaningful amount of minority class, a huge number of samples need to be selected and labeled.

In this chapter we are looking for ways to reduce the total number of labeling required in an extremely unbalanced data stream scenario. The default approach is to select random samples from a pool of data points. In order to obtain acceptable classification results later, a minimal required number of minority class data points are needed. Therefore, potentially the random approach will need to label large amounts of streaming data points, mostly of the majority class, to get a handful of minority class points. In our approach, we assume that the minority class may cluster inside the data space. Once the first minority class is obtained, we actively search for nearby data points to get more minority class. Approaches such as the one proposed by Chen and Tu, (Chen and Tu, 2007), will be useful because the minority class will have high density in some of the grids. We can focus on sampling these grids to get enough samples for minority class and reduce the total number of samples needed for labeling.

3.1 Related Work on Data Stream Labeling

Concept drift is a problem unique to stream data. In the traditional data mining task, all data points are present at one time and the data distribution is fixed, whereas in a streaming data environment new data could be arriving at any given time. It is never certain that one data model trained on initial data will be suitable throughout the streaming process (Tsymbal, 2004). Some adaptive system is required to handle those changes. Wang et al proposed a system that copes with concept drift using ensemble method (Wang, 2003). In their system, the weight on each of the vote from the committees of classifiers changes when the actual accuracy of one classifier becomes significantly different from the

expected accuracy. Kolter and Maloof proposed a similar system in their research where a dynamic weighted majority adapts to change of performance in the ensemble classifiers (Kolter & Maloof, 2003). However, weighted voting and majority voting have their limitations in that the underlying ensemble classifiers never changed. Tsymbal et al. created a new system where new classifiers are built over a time period and the best models are selected to be included in the ensemble learner (Tsymbal et al., 2008). In the research they have shown that their system outperforms both dynamic weighting total and dynamic majority voting.

To create a new classifier for the ensemble method, labeled samples are needed for the training process. In real world applications, however, the ensemble classifiers often need to deal with unlabeled data. Unlabeled data provides two challenges. First, we do not know the accuracy of our existing system with unlabeled data, therefore making detecting concept drift, or deciding when to train a new classifier, difficult (Zhang et al, 2009). Second, sample labeling is expensive and time consuming, but this task is required because labeled training samples are needed for building new classifiers for the ensemble learner. In an effort to reduce cost of the labeling process, Kantardzic et al combined clustering with ensemble classifiers to detect concept drift and decided when to train a new classifier used for ensemble learner based on partially labeled streaming data (Kantardzic et al, 2010). In their research a density-based clustering algorithm was used to detect the change of data distribution. If a new cluster emerges, it means there are enough data that potentially not fit in existing ensemble classifiers and the system should train a new model for the new data distribution. They have shown that by actively detecting concept drift, their system is more efficient and accurate than systems that train models over a fixed period amount of

time. They have also shown that their system needs roughly only 10% of the sample labeled in order to have a good classification result.

Even though the number of samples needed to be labeled is reduced by previous work, labeling is still a performance bottleneck because manually labeling is frequently necessary. Random sampling was used for sampling imbalanced data set under streaming data environment (Kantardzic et al, 2010). The problem is more apparent in an extremely unbalanced data set where minority class is less than 10% of the total data instances. A large number of data instances are needed in order to obtain sufficient number of minority class samples. However, if minority class samples are clustered, then finding such clusters can help increase the chance of selecting minority class instances. Tu and Chen created a grid density clustering algorithm which can be used in locating minority class clusters (Tu & Chen, 2009). In their research they divide the data space into grids and fit data points into the grids. If there is enough density in one grid, then that grid will become a cluster. If several adjacent grids all have enough density, then these grids will together form a larger cluster. In our research, our methodology is inspired by their approach, and it will be discussed in detail in the next section.

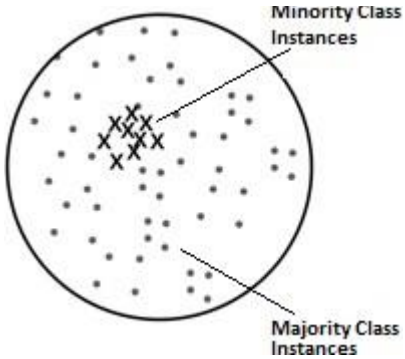


Figure 3.1. Demonstration of an unbalanced 2D data set.

3.2 Grid-based Labeling Approach

The eventual goal of this research is to help increase the efficiency of stream data classification by reducing the cost of labeling. For our approach we assume the minority class forms a cluster. Figure 3.1 demonstrates the scenario. The basic idea of our approach is:

1. We select a random instance in a set of streaming data points available for training a new classifier in the ensemble learner.
2. Once we selected an instance that belongs to the minority class, stop the random selection process and look for data points close to the selected minority class point.
3. Continue step 2 to select a certain number of data points close to the minority class instance selected, and then if more minority samples are required, continue 1 and 2.

We chose a basic version of the Grid Density Clustering algorithm by Tu and Chen (Tu & Chen, 2009). The grid density algorithm provides a fast and efficient way to identify dense areas of data points. If the data forms a cluster, the grid will be able to detect the density of the data and map out a profile of the cluster. It is not constrained by shape compared to density algorithm based on a radius (Kantardzic et al, 2010). It provides a simplified definition of distance between data points and therefore able to quickly locate and sample data points in the region of minority class cluster. Our basic version of the grid density clustering process is demonstrated in Figure 3.2.

Using the grid density clustering, we proposed a new approach for sampling data points to train new classifier in streaming ensemble learner. The control approach was random data selection where a random instance was selected for labeling until there were enough samples for both the majority class and minority class. The other four all used grid density cluster algorithm to try to find the area in data space where minority class were concentrated.

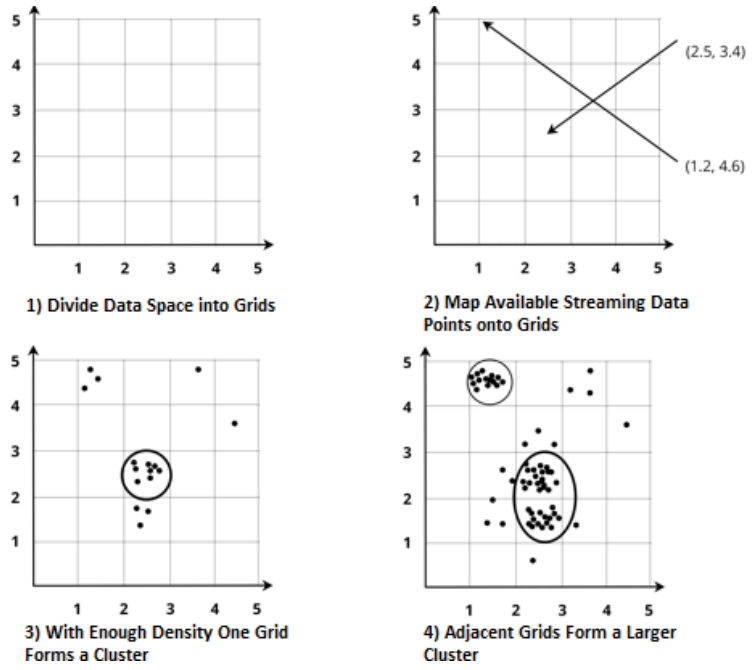


Figure 3.2. The basic Grid Density algorithm for clustering

We defined the percentage of minority class in a set number of data points as P. We defined a data set as extremely unbalanced when P < 10%. Our goal was to find at least M minority class instances with fewest overall data points sampled. The expected number of total sampled data points for the control approach is therefore M/P. When P < 10%, M/P is much larger than M, meaning the control approach is inefficient on extremely unbalanced dataset. Assume the data points has i dimensions and each dimension was divided into j grids. The total number of grids is j i. However, data points usually won't fill the entire data space and minority data points will only occupy a small number of grids. Therefore we defined the number of occupied grids as k where k <= j i and the number of grids that contain minority class as km .We also defined Pkm (i) as the probability of finding a minority class in grid i. Define Max(Pkm) as the maximum value of all Pkm.

A. Random Selection

This was the control approach. A data point from the sample was randomly selected for sampling. The random selection continued until we had obtained a required number of minority data points.

B. Random Grid Selection

This was an alternative to Random Selection in order to take advantage of the benefit of the gird. In this approach, a random occupied grid was first selected. Then a random data point was selected from that grid. The probability of selecting a minority class point is:

$$p = \sum_{km}^{i} \left(\frac{1}{k}\right) * \boldsymbol{P_{km}(i)} >= \mathbf{(1/k)} * \boldsymbol{Max(P_k)}$$

The ideal scenario of this approach is when data points are closely clustered and minority class points are also clustered with other minority class points. In this case k is small and Max(Pkm) is close to 1. As a result, p is much larger than P and the expected value N/p is much smaller than N/P.

C. Grid Search

This is our proposed approach. As mentioned above, we assumed that minority class points form a cluster. Therefore, if one grid containing the minority class was found, then potentially most of the other minority class data points were also in that grid. The process for sampling is shown in Figure 3.3.

Procedure Grid Search:

Input: List of available data points *d*, **list of grid cells** *g*, **percentage of minority class** *p*, **required total number of labeled samples** *T*
Output: *s*, **list of samples labeled**

```
While (|s| < T), do:
    Let dr = data point randomly selected from d in grid cell g[i];
    Label dr with human expert;
    Add dr to s
    If (label of dr is of majority class), continue;
    Let count = (1/p) * # of samples in g[i];
    While (count >= 0), do:
        Let dr = data point randomly selected from d in grid cell g[i];
        Label dr with human expert;
        Add dr to s
        count --;
Return s
```

Figure 3.3: Algorithm for Grid Search strategy of labeling samples

Figure 3.3 can be summarized as the following:

1. *A data point was randomly sampled from the set of currently available data points.*
2. *If the sampled instance was of majority class, continue 1. Otherwise go to 3.*
3. *If the sample was of minority class, then select no more than 1/P more samples from that grid.*
4. *If the required number of samples for labeling is not met, go to 1.*

The expected number of data points to get the first minority class is 1/P. At most 1/P samples were selected from a grid and among these samples, (1/P)*Pk are of minority class. The expected total number of data points sampled is s * (2/P) where s is the number of iterations. If minority data points were indeed clustered, then Max(Pk) is close to 1 and (1/P)* Max(Pk) is large such that s, the number of iterations, is small.

D. Combining Grid Search with Random Grid Selection

The two approaches B and C were combined into a new approach. The process for sampling is shown in Figure 3.4.

Procedure Grid Search:

Input: List of grid cells *g*, **percentage of minority class** *p*, **required total number of labeled samples** *T*

Output: *s*, **list of samples labeled**

```
While (|s| < T), do:
    Let gr = grid randomly selected from g
    Let dr = data point randomly selected from gr;
    Label dr with human expert;
    Add dr to s
    If (label of dr is of majority class), continue;
    Let count = (1/p) * # of samples in gr;
    While (count >= 0), do:
        Let dr = data point randomly selected from gr;
        Label dr with human expert;
        Add dr to s
        count --;
Return s
```

Figure 3.4: Algorithm for Combining Grid Search with Random Grid Selection of labeling samples

Figure 3.4 can be summarized as the following:

1. *A grid was first randomly selected and a sample from that grid was randomly selected.*
2. *If the sampled instance was of majority class, continue 1. Otherwise go to 3.*
3. *If the sample was of minority class, then select no more than 1/P more samples from that grid.*
4. *If the required number of samples for labeling is not met, go to 1.*

The total expected number of samples selected is:

$$s * \frac{1}{p} \; where \; p = \sum_{km}^{i} \left(\frac{1}{k}\right) * \boldsymbol{P_{km}(i)} >= \left(\frac{1}{\mathbf{k}}\right) * \boldsymbol{Max(P_k)}$$

where s is the number of iterations.

As discussed before, p will be large when k is small and Max(P_{km}) is close to 1. Therefore 1/p is small compared to 1/P while $P_{km}(i)$ is large, making s, the number of iterations, small.

E. Projected Grid Search with Dimentionality Reduction

High dimensionality creates a problem in our basic grid density algorithm in that there will be too many grids for data points. For instance, for a data set with 50 dimensions, there will be at least 250 grids assuming each dimension needs to be divided into at least two sections. Potentially for a small data set there will be only 2 or 3 data points in each grid, making the number of iterations s in the above approaches very large. To tackle such issue, we proposed preprocessing the data to reduce the dimensionality, and then construct grids on the preprocessed data space. The preprocessing we used was standard Principal Component Analysis. The sampling method we used on the preprocessed data was Grid Search (Section 3.2 C).

3.3 Experiments on Different Labeling Strategy with Different Dataset

To evaluate the effectiveness of each of the above labeling strategy, experiments were carried out over a synthetic dataset and two real world datasets. Since our goal is to make the labeling process more efficient, the evaluation metric for our approach is the total number of samples that was selected for labeling. Clearly the fewer samples needed the faster a new classifier can be trained and integrated into the streaming data ensemble learner. Therefore, we are looking for approaches that sample the least amount of data instances to achieve the required number of minority class instances.

3.3.1. Data Sets

A two-dimensional synthetic data was used to prove our concept. The synthetic data was generated such that x and y were randomly assigned an integer. If x was within range [20, 30] and y was within range [60, 70], then the generated data point was a minority class until there are 100 minority class within this region. Any extra data points in this range were labeled as majority class. If x was within range [30, 90] and y was in the range

Table 3.1. Synthetic Data Set Description

Case	Min.	Maj.	Class Dist.	x	y
10,000	100	900	0.01:0.99	[20,90]	[10,80]

[10, 60] and [70, 80] then the data point was labeled as majority class. The total x range was [20, 90] and the total y range was [10, 80]. The data set description is shown in Table 3.I. The data distribution is visualized in Figure 3.5.

Two real world data sets, Yeast and Satimag from the UCI Machine Learning Repository, was used to test our approaches. The two data sets were selected because they have extremely unbalanced classes based on our definition and all attributes are numerical, which simplify the definition of distance and grid size. Table 3.2 lists the basic description of the data sets.

Table 3.2. Real world data set Yeast and Satimag Description

Data Set	Cases	# of minority	Dimensions
Yeast	1484	44	9
Satimag	4435	415	36

The Yeast data set has 9 dimensions and 10 classes. The first feature of the data set is the sequence name of the strain and was ignored. The remaining 8 features are numeric and were considered in the experiment. To make testing simple, the data set was converted to a two-class data set such that class ME1 was selected as the minority class and the rest classes were deemed as the majority class. The numeric values were normalized between 0 and 100. The SATIMAG data set has 36 numeric dimensions and 6 classes. Class 4 was selected as the minority class and the rest majority class. The numeric values were normalized between 0 and 100 as well.

All the data points were labeled in these data sets. To create unlabeled data, the class label dimension of the data set was ignored when data was mapped onto the grid. The class label was only used when a data point was sampled out of the data points on the grid in order to check if the instance was a majority class or a minority class.

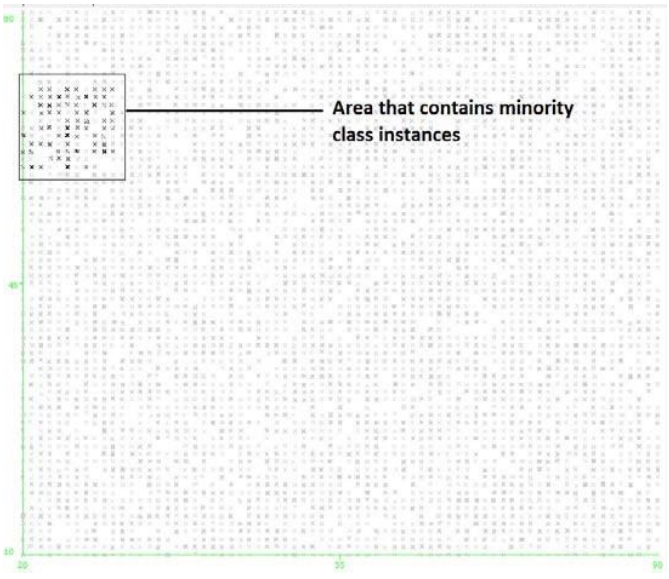


Figure 3.5 Synthetic unbalanced 2D data set visualization. X-axis starts at 20 and y axis starts at 10.

3.3.2 Data Sampling Result

In the synthetic data set experiment the number of samples required for majority and minority class, N, was set as 20. Figure 3.6 shows the result of different approaches on a varying grid size. Each grid has equal distances on the edges so that a grid with size 2 in a two-dimensional space means it is a 2 by 2 grid. Similarly, in n dimensional space a size m grid means it is a grid with length m on each edge of the grid. For the synthetic data, Approach E was not tested here because the dimensionality was not necessary for reduction.

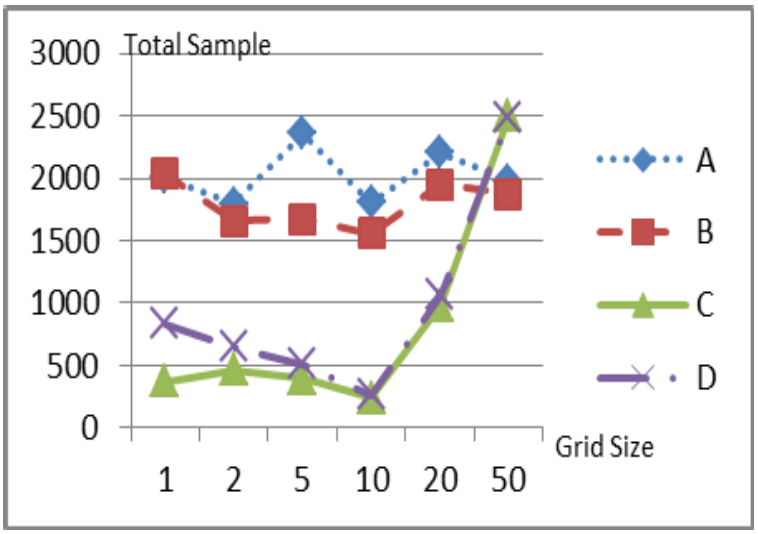


Figure 3.6. Total Number of data instances required to obtain 100 minority class samples under various grid size using Random Search (A), Random Grid Selection (B), Grid Search (C), Combining B and C (D).

As shown in Figure 3.6, Grid Search (C) and Combining Grid Search with Random Grid Selection (D) outperformed the Random Search (A) and Random Grid Selection (B) as expected until the grid size became very large. As the grid size increased, more samples were contained in each grid and the benefit of the grid diminishes. The result with large grid size being worse than random selection is most possibly because of sampling from a grid with only a few minority class instances but lots of majority class instances. In

approach C and D this scenario results in over-sampling in a certain grid with only few minority class samples.

From the synthetic data we observed that the result varies when different grid sizes were applied. Four factors, the number of samples, the percentage of minority class in the data set, number of dimensions and grid size all potentially impact how many samples are contained in each grid, which increases or diminishes the grid's ability of accurately mapping out the region of dense minority class population. In two extreme cases, when each grid only contains one sample, or when one grid contains all the samples, our proposed approach degrades into the random selection approach. However, between the two extreme cases, there could be a range of grid size where our approach can show improvement. To test such claim, the sampling algorithms were tested on real world data.

The Projected Grid approach (E) was introduced on real world data sets because of the high dimensionality problem stated before in Section III.E. Principal Component analysis was used to aggressively reduce the number of dimensions to 2 to show the difference between dimensionally reduced and non-reduced data. Then the preprocessed data was used instead of the original data for sampling. The Projected Grid was compared to Grid Search on non-projected data. Each algorithm was run 20 times and the average total number of sampled data instances for each approach was recorded.

For the Yeast data set, the required number of minority class samples was 20, given that there is only 44 minority class samples total. The sampling result with varying grid size is shown in Table 3.3 and it was plotted in Figure 3.7.

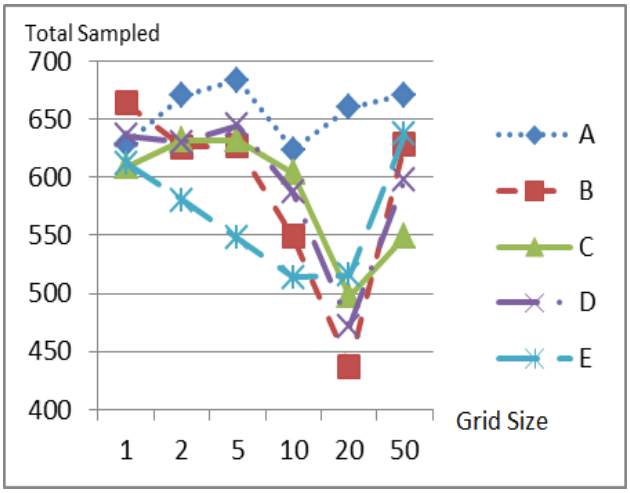


Figure 3.7. Experiment result of Yeast data (Table III) plotted to the number of samples needed to obtain 100 minority samples from the Yeast dataset under different grid size using Random Search (A), Random Grid Selection (B), Grid Search (C), Combining B and C (D)

Table 3.3. Total Number of data instances required to obtain 20 Yeast minority class samples under various grid sizes using Random Search (A), Random Grid Selection (B), Grid Search (C), Combining B and C (D)

Grids		Non-projected Results				Projected Results	
Grid Span	# of Grids Containing Samples	A	B	C	D	# of Projected Grids Containing Sample	E
1	1450	629	664	609	636	927	612
2	1449	670	626	632	630	498	580
5	1419	684	627	632	645	151	548
10	1008	624	549	604	587	53	514
20	337	660	437	498	472	20	516
50	49	671	629	549	598	6	637

Comparing our proposed Grid Search (C) to Random Approach (A), there are reduction in number of samples required in each Grid Span. Comparing C and B, improvements were made in Grid Span 1 and 50. Comparing our proposed Combined Grid Search (D) to Random Approach (A), there are reduction in number of samples required in each Grid Span. Comparing D and B, improvements were made in Grid Span 1 and 50. Projected Grid Search was slightly better than the non- projected approaches at grid span 2, 5, 10 and 50. At the optimal grid span in our experiment, Grid Span 20 and 50 for non-projected grids, our proposed approach (C) had an average improvement of 19.4% and the Combined Grid Search (D) had an average improvement of 17.6%. When grid span was between 2 and 20, Projected Grid Search (E) had an average improvement of 18.2%. Approach E does not have large gain versus the other approaches because the dimensionality of the Yease data set is still relatively small.

For the Satimag data set, the required minority class number is 100 and class 4 was considered the minority class while all other classes were grouped as the majority class. The result is listed in Table 3.4 and plotted in Figure 3.8.

In this high dimensionality case, when grid size is small each grid contains only one data point. This is shown in Table IV where the number of grids that contain samples is equal to data set size. The grid sampling approach degrades into random selection as the results from approach B, C and D showed no meaningful difference with random selection until the total grid number was brought down by a very large grid size. On the other hand, the projected grid reduced the number of grids to below 1000 and improved the results. However, when grid span was at 50, the projected grid has too few grids (5 grids total).

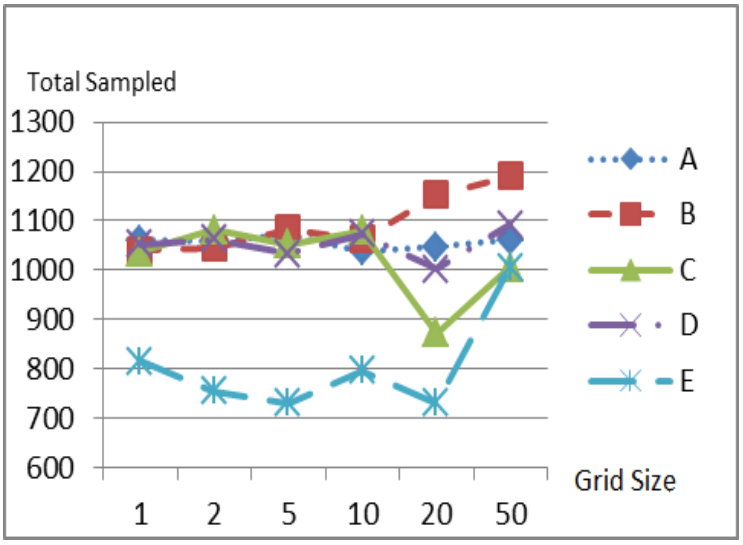


Figure 3.8. Experiment result of Satimag data (Table IV) plotted to the number of samples needed to obtain 100 minority samples from Satimag dataset under different grid size using Random Search (A), Random Grid Selection (B), Grid Search (C), Combining B and C (D)

Table 3.4. Total Number of data instances required to obtain 100 Satimag minority class samples under various grid sizes using Random Search (A), Random Grid Selection (B), Grid Search (C), Combining B and C (D)

Grids		Non-projected Results				Projected Results	
Grid Span	# of Grids Containing Samples	A	B	C	D	# of Projected Grids Containing Sample	E
1	4436	1061	1040	1073	1052	1742	815
2	4436	1057	1045	1081	1063	755	755
5	4436	1073	1084	1052	1033	180	731
10	4433	1039	1062	1080	1074	55	796
20	3750	1047	1153	891	1002	20	732
50	1688	1061	1191	1006	1092	5	1006

When the number of grids are low, the result was not as good in that too many samples are in each grid and over-sampling within grids occurs. In this experiment the optimal number of non- projected grids was between 3750 and 1688, and our proposed approach (C) had an average improvement of 5.3% and the combined (D) had an average improvement of 0.7%. The result was not impressive due to high dimensionality. On the other hand, the optimal number of projected grids in our experiment was between 1742 and 20, which had average improvement of 27.4%. Projected Grid improved the result significantly.

3.3.3 Dimensionality Reduction Analysis

The Projected Grid described above reduces the original Satimag data into two dimensions. It is unclear whether varying the number of dimensions will have an effect on the result of Projected Grid. This analysis was to explore such effect by fixing the size of the grid but reducing the dimensionality of Satimag data to various numbers of dimensions.

The grid size was fixed at 20, an optimal grid size for Projected Grid. Principle Component Analysis was used to reduce the Satimag data set into various dimensions. The result is shown in Table 3.5

The number of dimensions has an impact on the number of grids and therefore affects the sampling result. When dimensions were reduced to 1, Projected Grid shows only 10.6% improvement compared to the optimal 31.7% improvement achieved at reducing dimensions to 4. When dimensions were reduced to a still relatively large number, the results show no significant improvement over random selection approach. This again confirms the observation that too few dimensions created too few grids while too many dimensions resulted in too many grids. Both cases diminish the grid ability to locate

minority class clusters. According to our result, if Satimag was reduced to 4-16 dimensions using Principal Component Analysis, we see average 28.9% improvement over the random selection process.

Table 3.5. Total Number of data instances required to obtain 100 Satimag minority class samples using Projected Grid which reduce the original data into various dimensions.

Dimension	# of Grid Containing Samples	A	E
1	6	1057	944
2	20	1051	771
4	147	1061	725
8	891	1050	750
16	3231	1092	776
24	4373	1051	1045
32	4434	1090	1055

CHAPTER 4. A FRAMEWORK OF ADAPTIVE PREPROCESSING IN CONCEPT DRIFTING DATA STREAM MINING [3]

In real world applications, well preprocessed data can potentially increase the performance of the learning model significantly (Crone et al, 2006). In some situations, such as multimedia (video, voice, images, etc.) stream mining, preprocessing is a required step to increase the quality of the data (Kotsiantis et al, 2004). Therefore, preprocessing plays an important role in improving the final quality of adaptive stream mining frameworks.

Despite the importance of preprocessing, not many studies have been done on how the preprocessing step of a stream mining framework should be handled when there are changes in a data stream. Majority of studies tie preprocessing and model retraining together: only when the model needs to be adjusted is the preprocessing step updated. This approach assumes the only reason for decreases in model quality to be that the underlying data model has changed. While most of the time such assumption is valid, in some cases the data model does not change but instead the data values were not correctly preprocessed. This is when the preprocessing step needs to be adjusted to correctly handle the new data. By doing so the framework can still use existing models. One study demonstrated that it is

[3] This chapter has been published at Hu and Kantardzic, 2016

necessary to adaptively adjust preprocessing and separate it from modeling to have the best overall classification quality (Zliobaite & Gabrys, 2012).

In this Chapter we introduce Smart Preprocessing for Streaming Data (SPSD) approach that separates minmax normalization of numerical features from classification modeling. SPSD is different from previous study in that it does not re-normalize for each new chunk of data. Instead SPSD calculates two metrics. Metric 1 is the percentage of samples that fall outside of existing min-max range. Metric 2 is the percentage of difference between new sample values and recorded historical min-max values. When these two percentages reach above their threshold values, SPSD triggers a re-normalization using the latest minmax value in the stream. The metrics are used to avoid unnecessary re-normalization when there are noise and outliers in a stream. We demonstrate that in some cases SPSD can maintain comparable accuracy of stream mining framework without the need of retraining a new model, which reduces costs associated with model generation. The contributions are the following:

- We formulate the concept of smart preprocessing for numerical features.
- We developed a framework for preprocessing through re-normalization in streaming data environment.
- We demonstrate through experimental evaluation that data stream mining can benefit from smart normalization.

4.1 Related Work on Preprocessing in Stream Data

Yan (Yang et al, 2006) proposed two algorithms that are able to perform efficient and fast dimension reduction on large scale streaming data. The algorithms improved the existing Orthogonal Centroid algorithm to make it scalable to handle streaming data. Reddy et al, 2013 proposed an algorithm approach to preprocess web usage data, which is a type

of streaming data. They presented several techniques that identify sessions and users of a web usage log. Zliobaite & Gabrys, 2012 proposed a framework where adaptive preprocessing is used to adjust to changes in the data stream. Their study demonstrates that there are benefits in separately handling preprocessing and model. For every chunk of data, they employed 5 different combinations of handling preprocessing and models: the "old-old" uses old model and old preprocessing; the "new-old" uses new preprocessing and old model; the "old-new" uses old preprocessing and retrained model; the "new-new" uses new preprocessing and retrained model; the "select" select the best performance amount the 4 combination above. In their experiments they identified "select" to perform the best as it combines the benefits of all other 4 approaches. Within the "select" approach, there were cases when "new-old" or "old-new" were selected, thus demonstrating the benefit of decoupling preprocessing with learning model.

4.2 Smart Preprocessing for Streaming Data (SPSD)

We propose SPSD approach that re-normalize the data when needed, without changing the underlying model. The goal was to improve accuracy of a stream data mining framework by only adjusting the normalization step, thus reducing the number of times a new model is trained. The approach actively measures the amount of changes that have occurred in the current chunk. SPSD only calls for renormalization when the change amount exceeds some threshold value to avoid unnecessary re-normalization on noisy samples or samples with outliers.

As stream data samples arrives, they are grouped into equal sized chunks. All operations on the data samples are based on the current available chunk. The merits of chunk based approach is that it is capable of adapting to various types changes in data

stream (Street & Kim, 2001)(Wang et al, 2003).Chunk based approach also makes it easy to evaluate the quality of the framework.

Traditional metrics such as accuracy and F-score can be simply calculated within each chunk (Wang et al, 2003). The overall accuracy of chunk-based frameworks can be estimated by averaging all accuracy measures across all chunks. The first chunk of data was used to set the minmax parameter for normalization and send the normalized data to the underlying learning model for training. The first chunk was set as a reference point whose min-max values would be used as referenced min-max range when compared to by later chunks. This reference point is denoted as Po (mino, maxo).

```
Procedure Metrics:
Input: Current chunk of data cur, referenced minimum values for each dimension
refmin, referenced maximum value refmax, metric 1 threshold m1, metric 2
threshold m2, chunk size size.
Output: true, re-normalize and false, no re-normalizatin.
Let metric1 = false, metric2 = false
Let metric1count = 1
For each c in cur, do:
        For each dimension d of cur[c], do
                If ( c[d] < refmin[d] )
                        metric1counter++;
                        break;
                If ( c[d] > refmax[d] )
                        metric1counter++;
                        break;
If ( metric1counter / size  >  m1 ) metric1 = true;
For each c in cur, do:
        For each dimension d of cur[c], do:
                If ( ( refmin[d] - c[d] ) / refmin[d] > m2 )
                        metric2 = true;
                If ( ( c[d] - refmax[d] ) / refmax[d] > m2)
                        metric2 = true;
Return metric1 && metric2;
```

Figure 4.1: Metrics algorithm for smart normalization.

The approach uses two metrics that measure numerical feature value changes that appeared in the new data chunk:

- Metric 1: The percentage of samples in the new chunk that have at least one dimension fall outside of the referenced min-max value range.
- Metric 2: The maximum percentage of difference between new sample's values in each dimension and the referenced min-max value for that dimension.

Metric 1 is to separate noises and outliers from actual changes in a data stream. Metric 2 is to reduce the number of re-normalizations needed to speed up the approach. A threshold value for each metric was used, and the framework only calls for are-normalization when both metrics pass their respective threshold values. The algorithm for calculating metrics and determining whether the framework needs renormalization is described in Procedure Metrics in Figure 4.1. Procedure Metrics iterates through one chunk of data and calculate metric 1 and 2 separately.

For a new chunk of data, the samples were tested using Procedure Metrics. If Procedure Metrics returned true, re-normalization would update the recorded minmax values by the new min-max values found in the current data chunk. SPSD would normalize the current chunk of data using the new min-max value and send the normalized data to the underlying learning model for classification. The current chunk would replace the first chunk as the new reference point in the data stream. The new minimum value and maximum value of the chunk would form the new referenced min-max range. This reference point is denoted as Pi (mini, maxi), where i is the chunk number. If Procedure metrics returns false, then SPSD would not trigger renormalization in the current chunk. All data samples in the chunk were instead normalized using Pj (minj, maxj), where j is the previous reference chunk number. This process continues as more chunks of data come through the stream. The entire framework is described in Procedure SPSD in Figure 4.2.

Procedure SPSD initializes normalization using chunk i=0 then renormalize new chunks of data based on decision made in Procedure Metrics.

4.3 Experimental Results

In this section we applied SPSD approach to three datasets, two synthetic datasets and one real world Electricity Market (EM) dataset. We used synthetic datasets to show proof of concept and to compare two scenarios: normalization without model change versus normalization with model change. When applying minmax normalization, a scaling factor is applied to the datasets. As result the original model might change. We use the EM dataset for testing the approach and comparing with four traditional stream mining frameworks. In all experiments we used Support Vector Machine (SVM) as the underlying learning model

```
Procedure SPSD:
Input: Current chunk of data cur, metric 1 threshold m1, metric 2 threshold m2,
chunk size size, chunk number i.
Let refmin  = array( dimension of cur );
    refmax = array( dimension of cur );
If ( i  ==  0 ) do:
        refmin = minimum values in each dimension of cur;
        refmax = maximum values in each dimension of cur;
        Send cur to underlying learning model;
Else do:
        If ( Metrics(cur, refmin, refmax, m1, m2, size)) do:
                refmin = minimum values in each dimension of cur;
                refmax = maximum values in each dimension of cur;
                Normalize cur using refmin and refmax
        Else do:
                Normalize cur using refmin and refmax
        Send cur to underlying learning model
```

Figure 4.2: SPSD algorithm

4.3.1. Datasets

The two synthetic datasets were numerical, two-dimensional and have two classes. In both datasets, the first 5000 samples were generated within range [0, 10] on both dimensions. Then the upper boundary of the range was increase by 1 for each 5000 samples (e.g. 5000–10000 samples were generated within range [0, 11] and so on).

In total there were 55, 000 samples generated and the entire dataset's range was [0, 20]. The difference between the two datasets is the decision boundary, one used equation (4.1) as decision boundary while the other used equation (4.2).

$$y=x \tag{4.1}$$

$$y=0.8x+2 \tag{4.2}$$

If a data sample fell above the decision boundary, it was labeled as class 1 otherwise class 0.

EM dataset is a popular data set for streaming mining research (Harries & Wales, 1999). It contains 45, 312 samples. It is a near balanced dataset with two classes denoting whether the price of electricity has gone up or down. It has seven dimensions with five of them numerical and the rest two are date and time values. We removed the date and time dimensions, making the dataset fully numerical.

4.3.2 Results of Experimenting SPSD on Synthetic Dataset

In our experiments we compared SPSD with two baseline methods: 1) one that does not retrain learning model nor re-normalize the data, the “no-change” method and 2) one that re-normalizes and retrains the model in every chunk, the “all-change” method. We picked these two methods because they represent two extremes in data stream mining. “no-

change" methods never responds to change in dataset, while the "all-change" method always tries to adapt.

We applied our approach using 5% threshold value for metric 1, 5% for metric 2 and 2500 as the chunk size. We selected 5% for each metric because the data was generated with at least 5% change in range, and we wanted to capture all these changes. We implemented our approach using python and the results are shown in Figure 4.3.

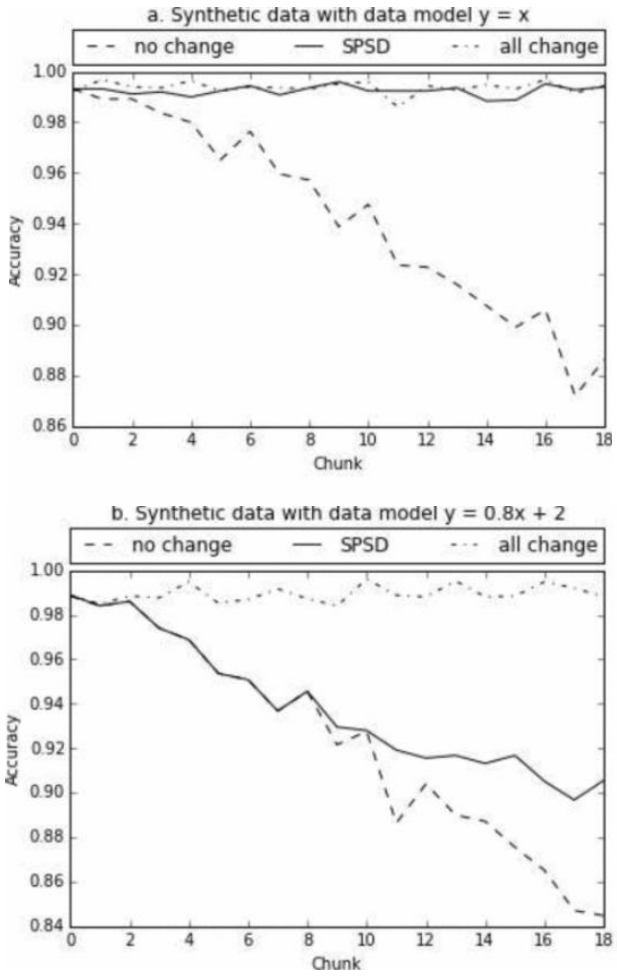


Figure 4.3. Accuracy curve on synthetic data. A). Accuracy for the y = x decision boundary. B). Accuracy for the y = o.8x+2 decision boundary.

Figure 4.3 shows clearly that as data gradually increased in range, the “no-change” degraded along with the data change. This is expected because “no-change” did not adjust its model according to the data range change. The “all-change” method remained at very high accuracy because it was constantly adapting and retraining. SPSD had consistent high accuracy in the first dataset as shown in Figure 4.3a. In the second case, SPSD degraded overtime as well but at a slower rate than the “no-change” approach.

As data increased from range [0, 10] to range [0, 20], the decision boundary of normalized data gradually moved down 0.1 unit. This change is illustrated in difference between Figure 4.4(A) and 4.4(B).

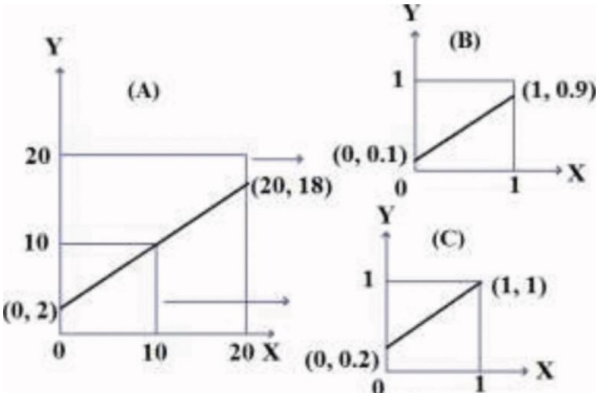


Figure 4.4. Decision boundary changes after re-normalization.

The synthetic data demonstrated that in certain cases when normalization does not affect the overall model of the normalized data, then smart normalization is enough to maintain the accuracy of the data as shown in Figure 4.3.a. When normalization does affect the model, as shown in Figure 4.3.b, retraining the learning model is the best approach.

4.3.3 Results of Experimenting SPSD on Real-world Dataset

In order to have good accuracy with the result. We first tuned 3 parameters of SPSD: chunk size, metric 1 and 2 thresholds. A good chunk size should provide high accuracy for the initial SVM model using the first chunk of data. The chunk size should be sufficiently large so that a good model could be trained, but it also shouldn't be too large so that there were only a few chunks for the entire data set. We tested 5 chunk sizes: 2000, 2250, 2500, 2750, 3000. The result is shown in Table 4.1. We can see that chunk sizes between 2000 and 3000 does not affect accuracy much. The difference between the best and the worst is only 2.56%. We chose chunk size 2500 (18 chunks in total) as it gave the best initial model accuracy (82.56%).

Table 4.1: Accuracy (in %) on first chunk of EM data with different chunk size

Chunk Size	2000	2250	2500	2750	3000
Accuracy of Initial SVM (%)	79.95	79.91	82.56	81.56	80.93

Next, we need to determine what threshold values of the two metrics produce the best overall result. Sensitivity analysis was performed on the two metrics. The first test we are looking to find optimal metric 1 value. We used 1%, 5%, 10%, 15%, and 20% for metric 1 and a fixed 10% threshold for metric 2. The result accuracy curve is shown in Figure 4.5.a.

We found that 1% for metric 1 produced better accuracy result from chunk 1 to 5. Therefore, we pick 1% as threshold value for metric 1. We now vary metric 2 by using 1%, 5%, 10%, 15% and 20% value. The result is shown in Figure 4.5.b. For metric 2 the Figure 4.5.b shows that setting 2 at 1% improve the accuracy of the SVM from chunk 1 to 5. For the rest of the stream all metrics didn't show any difference. To get the best overall result,

we picked 1% for both metrics, meaning for each chunk of data if 1% of samples falls outside of existing min-max range and if the new min-max is at least 1% more than the current range, then SPSD triggers are-normalization. After selecting appropriate chunk size and metrics, we then applied our approach to compare with "no-change" and "all-change" approach. Figure 4.6 shows the experimental result.

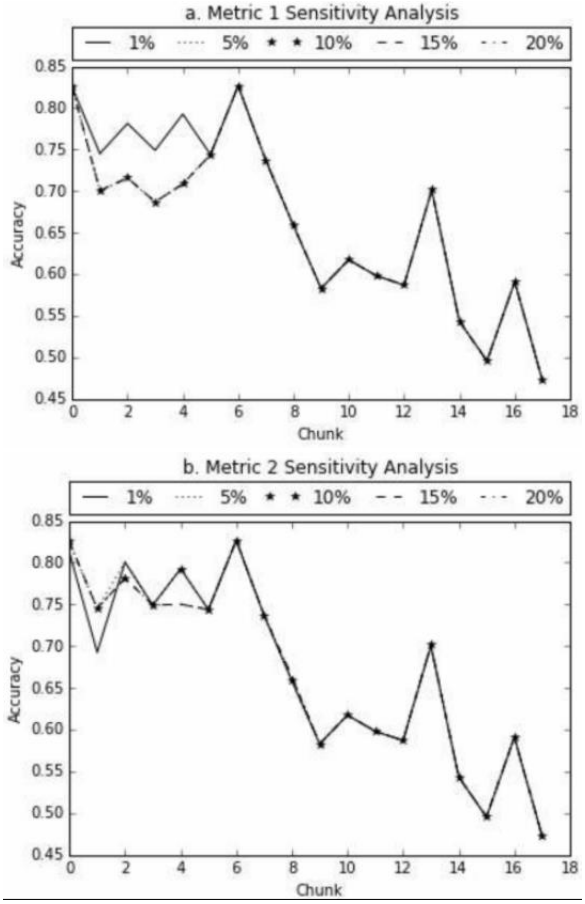


Figure 4.5 Sensitivity analysis for two metrics of SPSD. A) accuracy curves for varying metric 1 threshold. B) accuracy curves for varying metric 2 threshold.

From Figure 4.6, the "no-change" approach' accuracy degraded overtime as more data arrives. That is evidence that there might be changes in the data stream and an adaptive

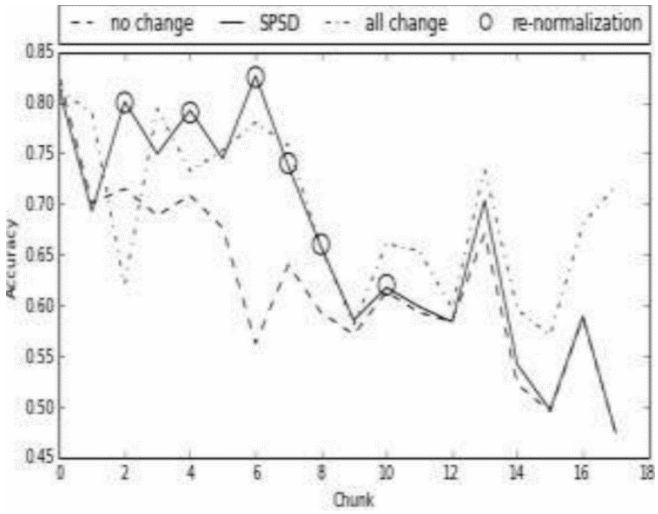


Figure 4.6 Comparison of SPSD with "no-change" and "all-change" on the EM data set using chunk size 2500.

mining framework is needed. The "all-change" approach performed erratically at the beginning of the stream, from chunk 0 to 3, then stabilized with accuracy consistently above the "no-change" approach for the rest of the stream. SPSD out-performed the "no-change" approach significant at the beginning of the stream from chunk 0 to 8. After the initial chunks SPSD rapidly dropped accuracy but still with small improvement over the "no-change" approach from chunk 9 to 15. Eventually at the end of the stream the "no-change" and SPSD approach converged. Upon close inspection, renormalization happened at chunk 2, 4, 6, 7, 8 and 10. Frequent re-normalization at the beginning means that there were significant data changes at the beginning of the stream. The high accuracy of SPSD indicates that these changes did not affect the underlying data model. This claim is also supported by the accuracy of SPSD compared to "all-change" between chunk 0 and 8. In those 9 chunks of data, SPSD was able to score higher accuracy than "no-change" 4 times. This means that retraining does not necessary produce better results in the first 9 chunks.

Starting at chunk 6 other changes in data started to appear. The change was significant enough that it sent all three approaches' accuracy down. From chunk 10 to 13 and after chunk 15 the “all-change” approach was able to adjust itself by generating new SVM model. Such adjustment resulted in significant higher accuracy of “all-change” compared to those of SPSD's and “no-change's”. When the underlying model changed, SPSD alone was not sufficient to maintain quality of the framework. All in all, although SPSD was not the overall best performing framework of all three, its accuracy in the first 9 chunks of data is high enough to justify keeping the underlying model unchanged for the first half of the entire dataset. SPSD was able to achieve this with only 5 re-normalizations performed out of 9 chunks. This result strongly demonstrates that smart preprocessing can reduce the number of re-preprocessing and retraining in a data stream mining framework.

4.3.4 Comparison of SPSD with Tradition Data Stream Mining Framework

We compared SPSD with SVM against four other traditional chunk-based stream mining frameworks: SEA (Street & Kim, 2001), AWE (Wang et al, 2003), ACE (Nishida & Omori, 2005) and MAE (Jiang & Lu, 2014). These approaches are all chunk-based ensemble classifiers that are able to detect and adjust to concept drift in the dataset. Jiang & Lu, 2014 compared the four approaches in their study on MAE framework. In their experiment, the chunk size was set to 500 for all frameworks. The maximum ensemble size was set at 25, meaning once the ensemble has 25 classifiers, newly trained models will replace older models using the respective replacement algorithm specified by each framework. We applied SPSD with chunk size 500 and 1% as both metric threshold values to EM dataset. Then we compared the resulting accuracy curve with those of Jiang & Lu. The comparison is shown in Figure 4.7.

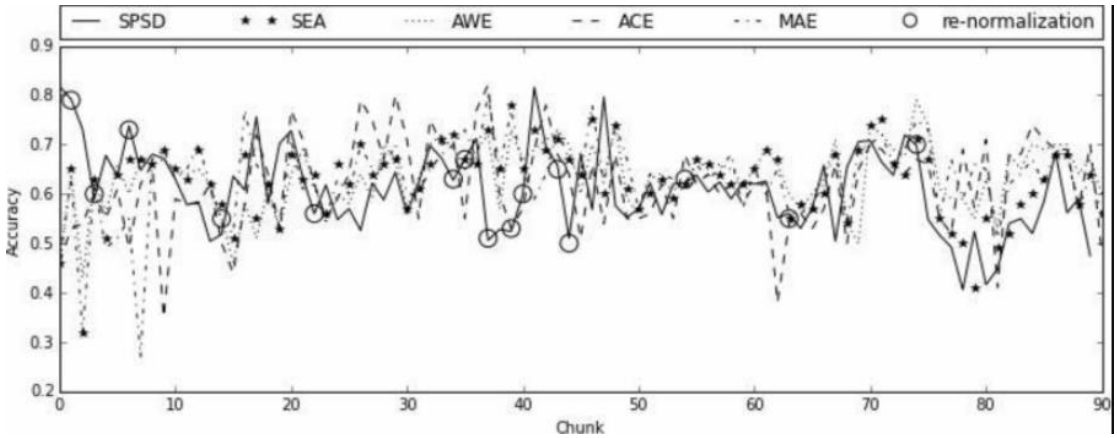


Figure 4.7 Comparison of SPSD with four traditional stream data mining frameworks

First, we inspect the re-normalization pattern for the SPSD. The majority of the re-normalizations happened at the first half of the data stream. Before chunk 30 (chunk 6 in previous experiment), there were 5 re-normalizations. Between 30 and 50 (chunk 10 in previous experiment) there were 7 renormalizations. This is consistent with previous experimental result: several re-normalizations in the beginning of the stream, then in the middle of the stream very frequent re-normalization. This again indicates that there were significant changes at the first half of the data stream.

We compared the accuracy curved of SPSD against the other four frameworks. From Figure 4.7 we can see that SPSD was able to produce higher accuracy than AWE,

Table 4.2: Percentage of chunks SPSD could and could not improve with each framework and potential accuracy if SPSD is integrated with each framework

SEA		AWE		ACE		MAE	
No Benefit*	Benefit**	No Benefit	Benefit	No Benefit	Benefit	No Benefit	Benefit
57.7%	42.3%	65.6%	34.4%	51.1%	48.9%	61.1%	38.9%

* Percent of chunks that do not benefit from SPSD

** Percent of chunks that do benefit from SPSD

ACE and MAE between chunk 0 and 10, and only produced lower accuracy than SAE in chunk 3, 6, 8 and 9. This again indicates that at the beginning of the stream, the underlying

model does not change. Retraining does not provide added benefit. Between chunk 10 and 70, SPSD ran in the middle of the pack most of the time, slightly under-performing from chunk 30 to chunk 40. In our previous experiment this section of the data was between chunk 6 and chunk 8, which was a period of steep decrease in accuracy for SPSD. The other four framework also ran very closely with each other, with occasional spikes from ACE. After chunk 70, the SPSD was mostly the least accurate, followed by SEA.

Given the available accuracy on each chunk, we investigated what percentage of chunks SPSD could potentially eliminate retraining of models if integrated with traditional framework. In our comparison, if SPSD produced comparable accuracy than a particular framework in a chunk, we count the chunk as one that does not require new model. Otherwise, we count the chunk as not being able to benefit by SPSD. The result is shown in Table 4.2. The framework that potentially benefit the most from integrating SPSD is ACE, where almost 50% of all chunks of data does not require training new models. The framework that benefits the least from SPSD is AWE, which still could have one third of all chunks not training new models. This proves our hypothesis that data streaming mining framework is able to benefit from SPSD by not requiring to retrain new models in each chunk of data

CHAPTER 5. SLIDING RESERVOIR APPROACH FOR DELAYED LABELING IN STREAMING DATA CLASSIFICATION[4]

Concept drifting data streams require the data mining framework to be able to detect changes in the stream and adapt to them so that the learning model is kept up-to-date (Hoens et al, 2012). Numerous studies have been done on designing such adapting data mining frameworks (Farid et al, 2013)(Brzezinski & Stefanowski, 2013)(Rutkowski et al, 2014). These frameworks continuously monitor the data stream for concept drift. Once a drift is detected, the frameworks adapt to the change by training new models or updating existing incremental models. Often the training process requires certain amount of labeled data to be effective. Most of the previous studies assumed that the required labels are available at the time before the training of a new model. This is not the case for many real-world data streams, in which human experts are required to take time and perform the labeling. For instance, a framework for detecting spam emails often needs to adapt its learning models to new spam patterns. The adaptation usually does not happen immediately because the framework needs enough people to identify their emails as spams and report them. Lots of samples from the new spam pattern need to be reported in order to have a good sample size. In cases like this there will most likely be a delay between the time when changes in data stream occur and the time when labels arrive. We call such cases,

[4] This chapter has been published at Hu and Kantardzic, 2017

where building a new model is necessary in response to concept drift but the required labels are not immediately available, the delayed labeling problem.

A naive solution of the delayed labeling problem will be requesting labels immediately at the time of concept drift (Krempl et al, 2014). Then the framework waits for the labeling process to finish before building any updated models. We call this the wait-and-train approach. This solution has risk of having outdated models during the waiting time. If the occurrence rate of concept drift is faster than the labeling process, the models of wait-and-train framework may be permanently outdated. Furthermore, if requested labels never become available, then the models will never be updated. Clearly, a more robust solution is needed other than wait-and-train.

We propose Sliding Reservoir Approach for Delayed Labeling (SRADL) framework that addresses the problem. Our approach employs a novel method of storing and managing available labeled samples. SRADL contains three components. Each component handles different aspects in a streaming environment with delayed labeling: label reservoir that keeps track of the arrival of labeled samples, change detection that monitors concept drift, and semi-supervised learning that updates the framework's predictive models. Our hypothesis is that SRADL will give better classification results in a delayed labeling setting when compared to the naïve wait-and-train approach. The contributions of the chapter are the following:

1. We formulate and implement a streaming data classification framework that handles delayed labeling.

2. We show that the framework can produce better result than the naïve approach.

There has been researches that mentioned delayed labeling problem. Those studies recognize that labels can be delayed, but they do not offer an entire framework to solve the problem. Mesterharm, 2005 focused on solving the problem of delayed label feedback. A delayed label feedback problem is where a learning model is trained using labeled samples. The learning model cannot be tested because labeled samples for testing are not available. The study focused on modifying existing learning framework to compensate for the delay. Zliobaite, 2010, proposed a change detection framework that is able to detect data changes with unlabeled data, thus reducing how much the framework relies on labeled data in order to adapt to concept drift. Masud et al, 2010, demonstrated the problem of delayed labeling in novel class detection problem. It addresses the fact that labels are not always available in a real-world streaming data environment. Their approach is able to utilize unlabeled data to reduce the need on labeled samples for novel class detection.

5.1. Delayed labeling problem

In a data stream, certain amount of labeled data samples is needed to be labeled for training new supervised or semi-supervised learning models when concept drifts occur (Hoens et al, 2012). A request for labels on selected data samples will be made prior to the training. If the labeling is not delayed, these requested samples will be labeled immediately, hence a new model can be trained shortly after. In a delayed labeling setting, the labels will not be immediately available and the amount of waiting time might or might not be known. When the labels do arrive, there are two scenarios in which labels are made available, illustrated in Figure 5.1.

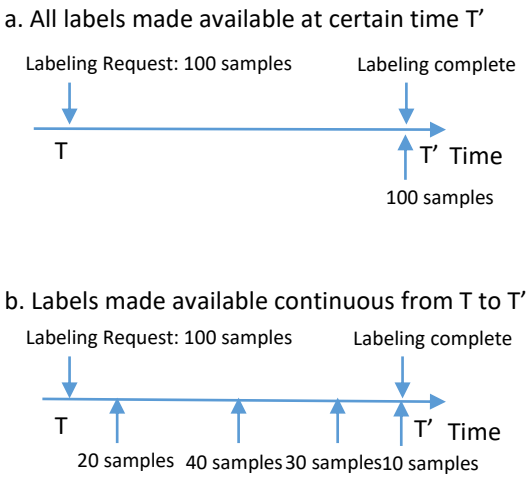


Figure 5.1 Illustration of two scenarios of delayed labeling. a) All labels become available after delay time T. b) Available labels trickle in until all labels become available after delay time T.

Figure 5.1 assumes that a concept drift occurred at time T. To train new learning models after concept drift, 100 unlabeled data samples are required for human labeling. Figure 5.1.a shows the first scenario where the labeling process completes, and 100 samples were obtained at T'. Figure 1b shows the second scenario where parts of the 100 samples arrive incrementally over time, completing the labeling process at T'. In either case, traditional streaming mining methodology might need to wait until all requested labels are available at T'. Between T and T', these frameworks are still using the model trained before T, which is likely outdated because of concept drift. In a real-world application, the interval of T and T' might potentially be very long, thus reducing the overall performance of the framework. Therefore, the main challenge of delayed labeling is how to keep learning models up to date after a concept drift occurs without immediately available labels. The goal of solving the delayed labeling problem is to

maintain the prediction performance during the waiting time so that the overall performance of the framework remains high.

5.2. Sliding Reservoir Approach for Delayed Labeling (SRADL)

SRADL uses a chunk-based approach to handle concept drift detection and model training (Fan, 2004). A chunk-based approach divides data streams into fix-sized groups of data samples, or "chunks". The framework then processes the data stream chunk by chunk. It also initializes itself by first using a chunk from the stream as the initial training dataset. The SRADL framework has three main components: Concept Drift Detection, Semi-supervised Learning, and Labeled Sample Reservoir. The structure of the framework is shown in Figure 5.2.

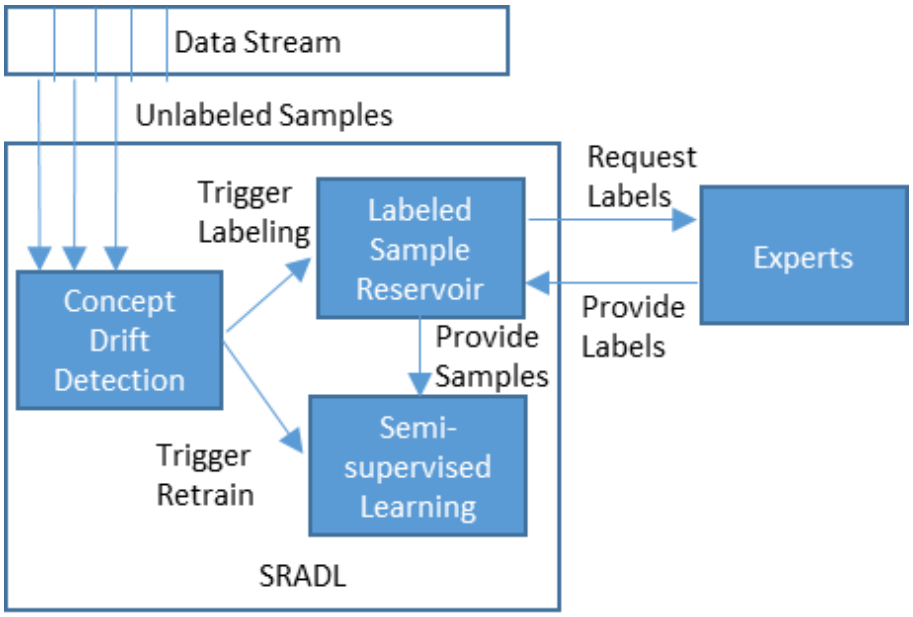


Figure 5.2. Overview of the SRADL framework.

The data from the stream are first sent through the Concept Drift Detection component. This module uses unsupervised approach to detect changes in the data stream (Ryu et al., 2012). Once detected, it signals the Semi-supervised Learning component to

start training a new model. The Semi-supervised Learning component then immediately trains a new model based on current unlabeled samples and stored labeled samples inside the Labeled Sample Reservoir. Concept Drift Detection also signals Labeled Sample Reservoir to make a labeling request. As labeled samples arrive in the future, they are stored and managed by the Labeled Sample Reservoir.

5.2.1 Labeled Sample Reservoir in SRADL

The Labeled Sample Reservoir is an ordered, fixed-size list of labeled samples. Let R denotes the list:

$$R = \{r_n: n=size\ of\ reservoir\}$$

where r_i is a 4-tuple in the form of:

$$r_i = (S_i, L_i, RT_i, AT_i)$$

S_i is a data instance sampled from the data stream to be labeled. L_i is the labeling result of the sample. RT_i is the time at which the labeling was requested. It is instantiated when the sample is sent to experts for labeling. AT_i is the time at which the label actually arrived. It is instantiated when a labeled sample returns to the reservoir from an expert. In a delayed labeling scenario, $RT_i \leq AT_i$.

R list is sorted by RT as the primary key and AT as the secondary key. The size n is the number of samples needed by the learning algorithm to successfully train and test a model. For example, if a learning model requires 100 samples to be labeled out of every 1000 unlabeled samples, then n = 100.

The reservoir is initialized using labeled samples from the initial training dataset. Since we assume the initial training dataset is labeled, the RTs and ATs of these samples are instantiated to be 0. Every time a new labeled sample arrives, it replaces the oldest labeled sample in the reservoir according to RT first and AT second. In the extreme case, a particular newly arrived sample r' can have RT' earlier than all other samples in the reservoir. This means that the time it took to finish labeling r' is so long that later requested labels already occupy the entire reservoir. In this case r' is considered too out-of-date and is discarded.

Since not all samples in the data stream are to be labeled, the Labeled Sample Reservoir can employ any labeling selection criteria, such as criteria used in (Wang et al, 2012) and (Žliobaitė et al, 2013). The decision of which criteria to use should be determined by the nature of the dataset and the needs of the specific real-world application. To simplify our approach, we selected samples by random.

5.2.2 Concept Drift Detection in SRADL

SRADL's Concept Drift Detection module can use any concept drift detection algorithm. In this study SRADL employs a density-based concept drift detection approach similar to Ryu et.al. Density based detection assumes that samples of the same class form clusters. Each cluster C is defined by a radius rad_c and a cluster density d_c:

$$rad_c = \textit{longest distance between sample and its cluster center.}$$

$$d_c = \textit{number of samples in cluster / rad}$$

Euclidean distance is used for the calculation of rad_c

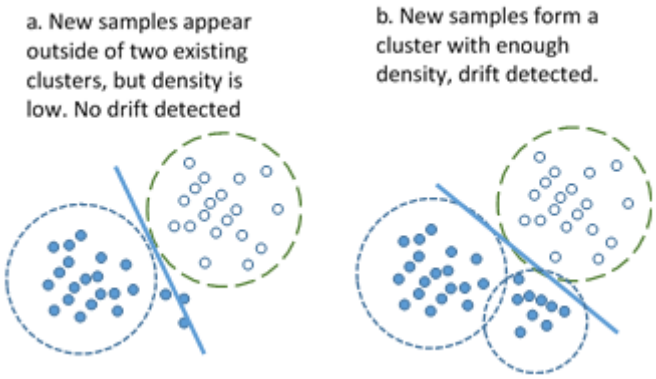


Figure 5.3. Illustrating density-based concept drift detection.

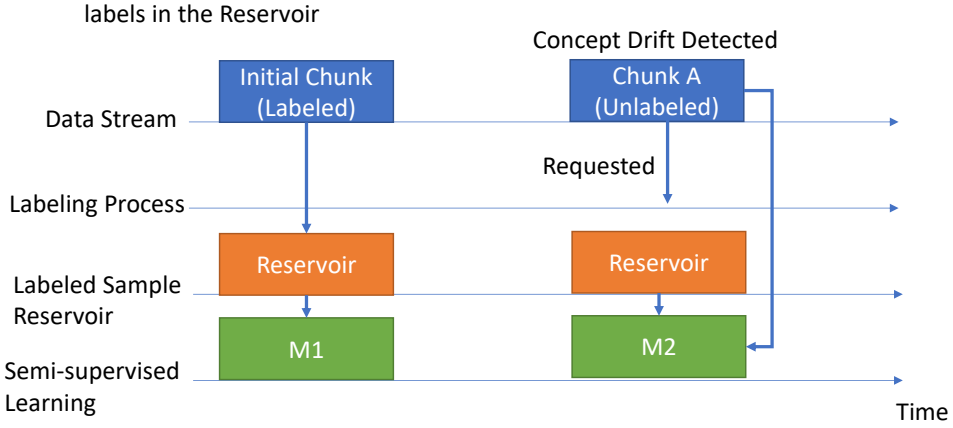
Initial clusters of samples are obtained from the initial training set of the framework. K-means clustering algorithm is used (Hartigan & Wong, 1979). As new sample s arrives, if its distance from the center of any existing cluster C is less than rad_c, then the sample is included in cluster C. If there does not exist any cluster that s can be included in, then s is considered an un-assigned sample, denoted by ~s. As time progresses, more and more ~s can appear. SRADL will try to cluster ~s after each chunk of data. When some of the ~s samples form a new cluster, SRADL determines that a potential concept drift has happened. The detection process is illustrated by Figure 5.3. In Figure 5.3.a, two existing clusters of samples are divided by a classification model. Some newly arrived samples fall out of the existing clusters, but the density of the new samples is low. The learning model does not need adjustment. After some time more samples arrived. The new samples form a third cluster as shown in Figure 5.3.b. This event signals the framework that a potential drift has occurred. A new learning model is trained in response.

5.2.3 Semi-Supervised Learning using Labeled Sample Reservoir

When concept drift is detected, SRADL immediately requests for labeling on samples from the current chunk of data. At the same time Semi-supervised Learning component uses labeled samples from the Labeled Sample Reservoir and unlabeled samples from the current chunk to train a new semi-supervised model. Any semi-supervised learning algorithm can be used in this component, such as (Li et al, 2009)(Wang et al, 2010)(Bennett & Demiriz, 1999). In this study, SRADL is implemented with S3VM (Bennett & Demiriz, 1999).

After the new model is trained, a performance evaluation is done on the new model when previously requested labels arrive later. This model-training-performance-evaluation process is visually illustrated in Figure 5.4. The "Data Stream" axis denotes the data stream through time. The "Labeling Process" axis denotes the labeling process through time. The "Labeled Sample Reservoir" and "Semi-supervised Learning" denotes the status of the two components through time. At the beginning of the stream (Figure 5.4.a), the first chunk of data is used for initial training. Its samples are partially labeled and put into the reservoir. An initial model M1 is also trained. At Chunk A, Concept Drift Detection detects a change in the stream. It signals Semi-supervised Learning to train a new model and at the same time it signals the SRADL framework to request for labeling on the current chunk of data. Semi-supervised Learning trains a new model M2 using labels from the reservoir and unlabeled samples in Chunk A. As requested labels arrive later in time (Figure 5.4.b), they are added to the reservoir and are used to test M2. If M2 is determined to be performing well, the model is kept unchanged. Otherwise, Semi-supervised Learning repeats a similar process to Figure 5.4.a to try to train a new model M2'. M2' is trained using reservoir labels and unlabeled samples from the current chunk in the stream (different from the chunk used

a. At the start of the stream, the first chunk of data is used to train the initial model (M1). When concept drift is detected, labels are request and new Model M2 is requested using labels in the Reservoir



b. Continue from a, requested labels from chunk A starts to arrive. They are added to Reservoir and used for testing and refining the model

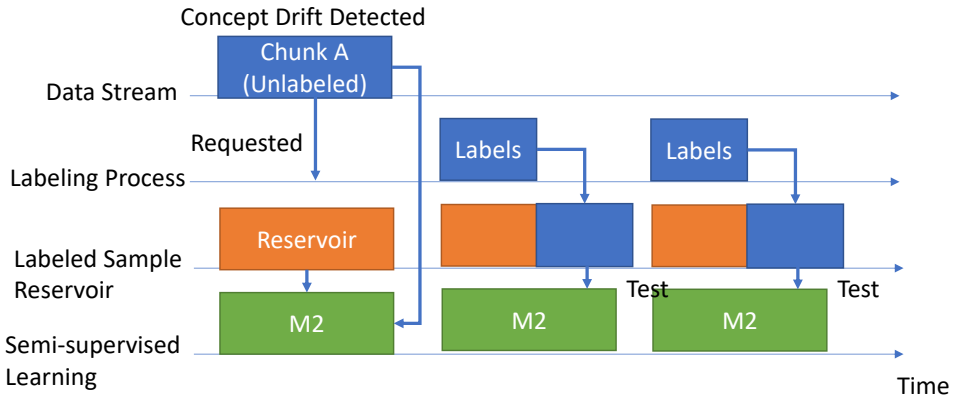


Figure 5.4 Illustration of building and evaluating a model after concept drift

to train M2). SRADL also requests for more labels from the M2' chunk. Model After training, M2' undergoes the same evaluation process as M2 (Figure 5.4.b). In the extreme case when required labels never become available, SRADL is still able to train new models using labels in the reservoir. However, the evaluation process will not be able to carry out since there is no labeled samples to test the performance of the new model.

SRADL uses a performance threshold P to determine whether a learning model is low performing or not. Any model with performance below P will be retrained. P is a parameter that balances between computational intensity and performance. The value of P

is up to specific applications because it is difficult to determine the optimal P without the prior knowledge about the data. For example, an application for predicting which color will be trendy in fashion can have a lower P value than an application for predicting weather. To keep matters simple, in this study the value of P is determined empirically.

5.3 Experimental Results

5.3.1 Datasets

Two datasets were used in the experimentation: Rotating Hyperplane and Spam. Rotating Hyperplane dataset (Fan, 2004) is created with 10,000 samples. It is a binary class dataset with 10 numerical features ranging between 0 and 1. A high dimension hyperplane divide the dataset into its two classes. Concept drift is created by rotating the hyperplane. When generating the dataset, parameter K determines how many drift events occur and parameter T determines how much rotation is done for each drift. Our dataset was generated using K = 4 and T = 1.0. Spam dataset is a real-world dataset. It is a text data converted numerical dataset, where each feature is the occurrence rate of a particular word in an email. The dataset has 500 features with two classes: spam and not spam. It has 9324 samples in total. Our change detection algorithm detected 11 possible concept drifts in the Spam dataset. These two datasets were selected because they contain a good number of concept drift.

5.3.2 Experimental set up

Two scenarios of labeling arrival time (Figure 1) were both explored. The labeling process was simulated by first hiding all class labels from the framework and only revealing the labels for samples that are requested to be labeled. The delay time is measured

by number of chunks between label requesting time and label finishing time. For example, a 6-chunk-delay problem when labeling is requested at chunk #5 will finish at chunk #11. For the first scenario, all requested labels are made available only after a pre-defined delay, as shown in Figure 1-a. To be precise, for n-chunk-delay experiment, if change were detected on the m_{th} chunk, all K requested labels will be made available on the $(m+n)_{th}$ chunk. The second scenario is where labels are made available incrementally over a period of time (Figure 1-b). Each chunk after the m_{th} chunk will get K/n number of labels. All K requested labels will still be made available on the $(m+n)_{th}$ chunk. The delay times for each experiment are arbitrarily chosen such that we can compare the performance of SRADL against other approaches in various length of delay. In real world scenarios, the delay time can vary for each application, and it is most likely determined by how long it takes the experts to finish labeling the data.

We compared SRADL with three other data stream mining approaches: a) static, b) no-delay, and c) wait-and-train. The static approach assumes there is no further changes in the data stream. The learning model was trained in the initial chunk and remained unchanged throughout the stream. This approach was used to show that concept drifts exist in the selected datasets. It provides a lower bound of performance. No-delay approach obtains labels immediately after requested, after which an updated model can be immediately trained. This approach was to give an upper bound of performance. Wait-and-train approach is the naïve solution to delayed labeling problem. It waits for the labeling process to finish and only trains a new model after all requested labels arrive. Performance was measured in area under the prediction accuracy curve, calculated by the Trapezoidal Rule that simulates integrating of the curve.

5.3.3 SRADL Experiment with Synthetic Dataset

For the synthetic dataset the chunk size was chosen to be 300. This chunk size was chosen such that the initial model can obtain the highest accuracy. The threshold performance value P was empirically set to be 75% accuracy based on the average accuracy of the static model throughout the data stream, which is 75%.

Figure 5.5 shows the experimental results of labeling scenario 1. The vertical line in the figure denotes the time when concept drift was detected. In Figure 5.5.a and 5.5.b, we can see that SRADL first performed slightly better than the naïve approach. Since the naïve approach waits for the labeling process to finish, at the beginning it had the same

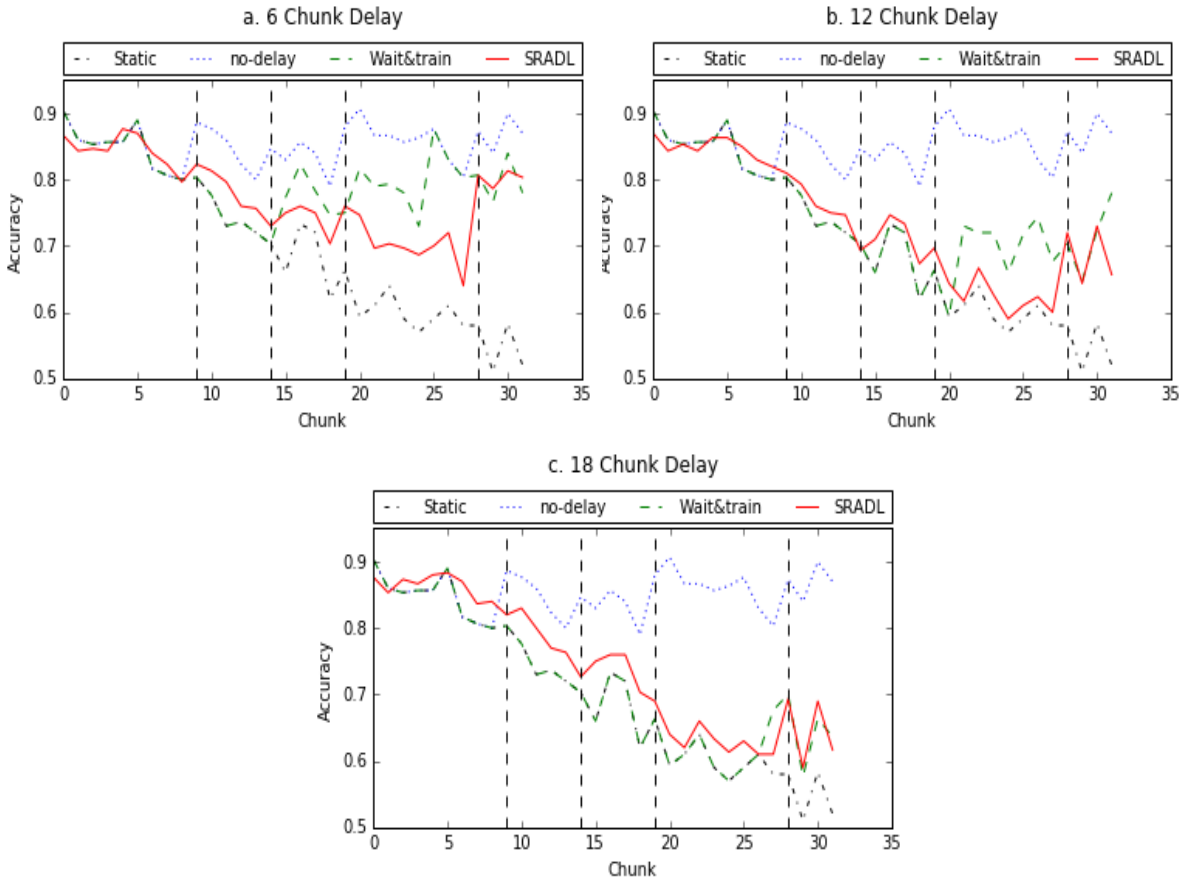


Figure 5.5. Experimental results of Hyperplane data with labeling scenario 1. Chunk size 300. Vertical line shows time of concept drift.

degrading performance as the static approach. After retraining, the wait-and-train bounced back nicely and even out-performed SRADL. In Figure 5.5.c, it shows that for larger delays SRADL was able to perform slightly better than the naïve approach from the beginning to the end. Table 5.1 computes the area under the curve presented in Figure 5.5. In 6 and 12 chunk delays, the area under the curve showed that SRADL performed worse than wait-and-train by 3.1% and 1.5% respectively. In 18 chunk delay, Table 5.1 shows SRADL had a 3.6% increase in performance. The small improvement of SRADL compared to wait-and-train can be explained by the lack of new labels to update the reservoir during the labeling process. Since no new labels are available during the waiting time, SRADL and wait-and-train has the same knowledge about the data stream. Semi-supervised learning trained using outdated label with new unlabeled samples produced worse models than its wait-and-train counterpart, which used supervised learning models on all requested labels.

Table 5.1 Area under the curve for labeling Scenario 1 experiments with rotating hyperplane

Delay	Static	No-delay	Wait&Train	SRADL
6	718.5	871.2	817.3	791.0
12	718.5	871.2	761.6	749.5
18	718.5	871.2	732.4	759.4

Table 5.2 Area under the curve for labeling Scenario 2 experiments with rotating hyperplane

Delay	Static	No-delay	Wait&Train	SRADL
6	718.5	871.2	817.1	809.4
12	718.5	871.2	761.6	785.7
18	718.5	871.2	732.4	787.7

However, with larger delay, the S3VM semi-supervised learning algorithm was able to overcome such drawback.

Figure 5.6 shows the accuracy curve of Scenario 2. Figure 5.6.a shows that SRADL performed similarly to wait-and-train until the chunk #20, where wait-and-train started to outperform. In 12 and 18 chunk delay experiments (Figure 5.6.b and 5.6.c), SRADL greatly outperformed wait-and-train for the entire dataset. In Table 5.2 we can see that for 6 chunk delay SRADL performed worse by merely 0.9% while in the other two cases it out-performed wait-and-train by 3.1% and 7.5% respectively.

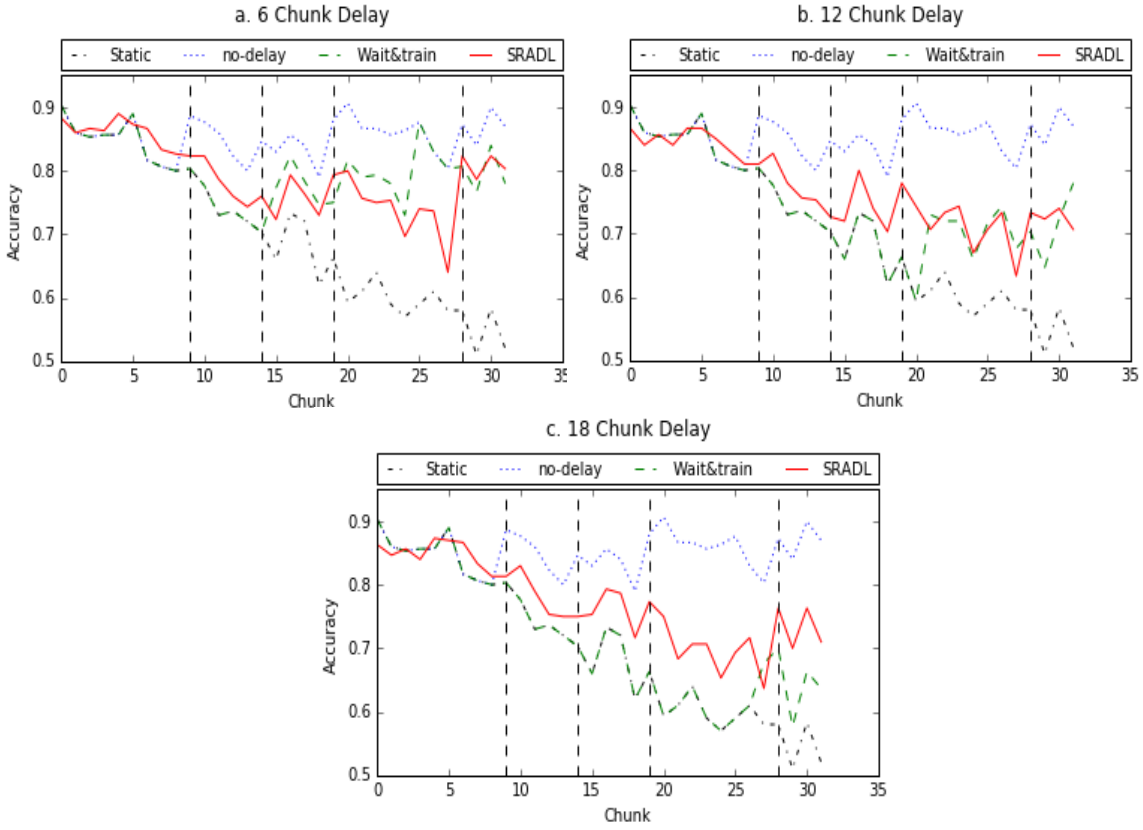


Figure 5.6. Experimental results of Hyperplane data with labeling scenario 2. Chunk size 300 Vertical line shows time of concept drift.

When new labels constantly update the reservoir, SRADL is able to effectively utilize the new information by integrating them into the latest models. SRADL especially

showed its benefits on larger labeling delay. The naïve approach of wait-and-train is limited to an outdated model for a long period of time while SRADL improves the model immediately.

5.3.4 SRADL Experiment with the SPAM Dataset

In the Spam experiment the chunk size was chosen at 200. In this dataset the average accuracy of static model is around 50%, which is too low of a performance to be a meaningful threshold. Therefore, the threshold performance value P is again set to be 75% accuracy. Shown in Figure 5.7 is the result of labeling scenario 1 experiment of the Spam dataset. SRADL performed much better at 6 chunk delay as shown in Figure 5.7a. The most performance gain came between chunk 15 and chunk 30. In 12 chunk delay, SRADL performed worse than the wait-and-train approach. Specifically, in Figure 5.7.b, SRADL performed similarly compared to wait-and-train until chunk 30-44 where SRADL fell below the naïve approach. For 18 chunk delay SRADL has a similar result than the wait-and-train approach for the entire stream.

Table 5.3 Area under the curve for labeling Scenario 1 experiments with SPAM

Delay	Static	No-delay	Wait&Train	SRADL
6	473.1	1815.3	1396.3	1490.0
12	473.1	1815.3	1375.6	1280.7
18	473.1	1815.3	1251.5	1267.3

Table 5.3 summarizes area under the curve computed from Figure 5.7, The worst performance loss of SRADL comes from 12 chunk delay, where SRADL performed worse than naïve case in the 12-chunk delay case by 6.9%. However, performance is gained in the other two cases. In 6 chunk delay case. SRADL outperformed wait-and-train by 6.7%. And in 18 chunk delay case SRADL outperformed wait-and-train by 1.2%. The result show that SRADL has more performance advantage than wait-and-train when the delay time is short. For longer delays, SRADL did not seem to benefit from semi-supervised learning algorithm in labeling scenario 1 since no new knowledge is gained about the data stream during the label waiting time.

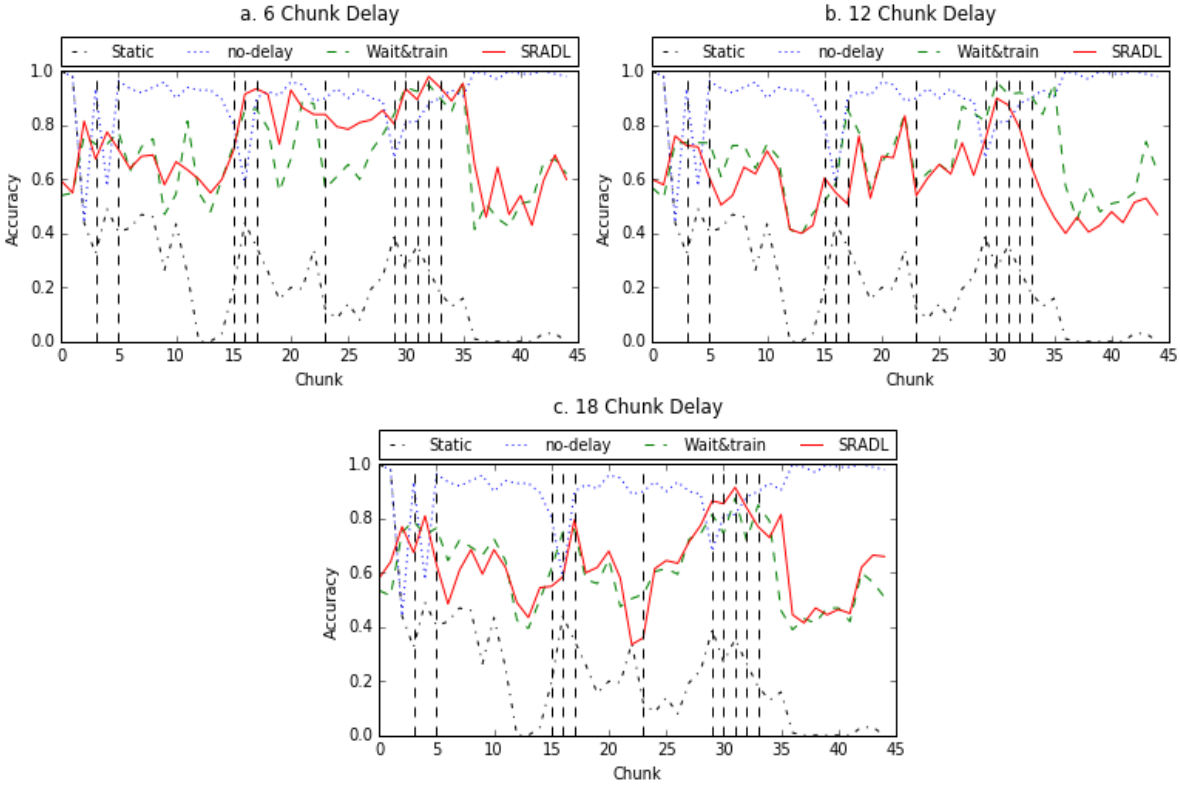


Figure 5.7. Experimental results of Spam data with labeling scenario 1. Chunk size 200. Vertical line shows time of concept drift.

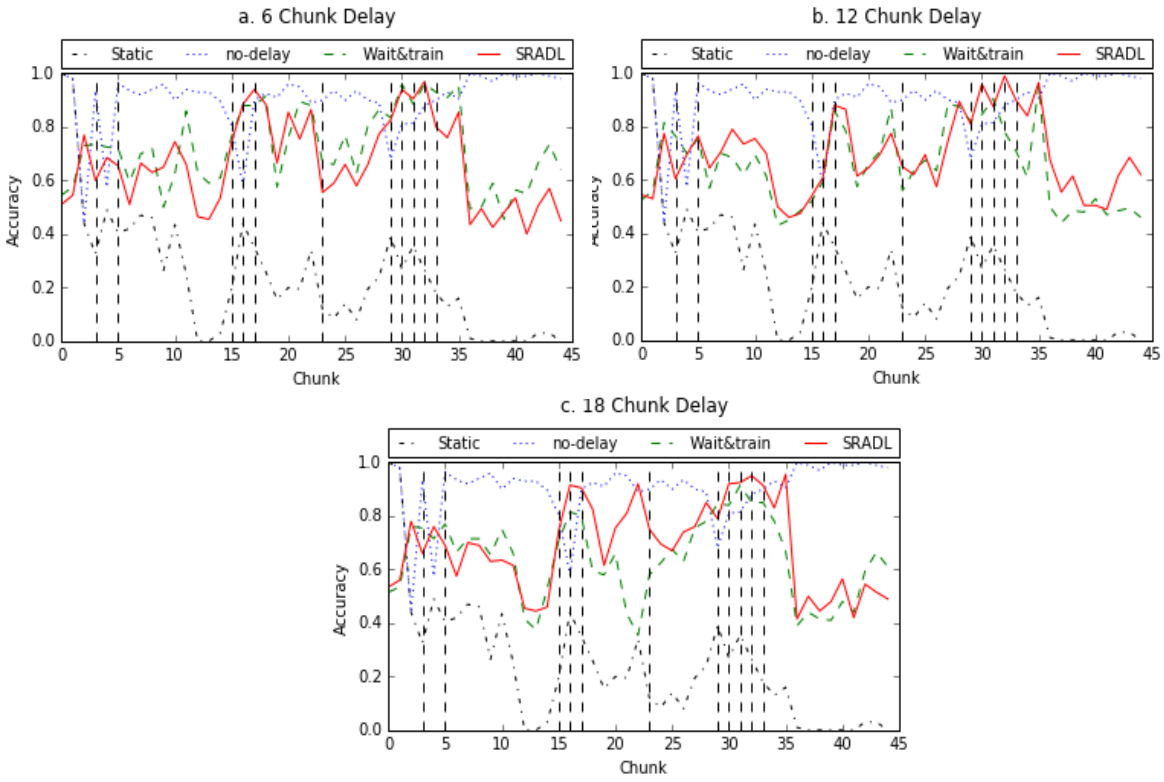


Figure 5.8. Experimental results of Spam data with labeling scenario 2. Chunk size 200. Vertical line shows time of concept drift.

Scenario 2 results are shown in Figure 5.8. For small delays of 6 shown in Figure 5.8.a, SRADL had no large improvement over the wait-and-train approach. SRADL performed slightly worse for the majority of the stream. In Figure 5.8.b, SRADL performed slightly worse between chunk 20 and 30, but outperformed from chunk 5 to 15 and from chunk 30 to 40. In Figure 5.8.c SRADL clearly outperformed wait-and-train between chunks 15-40, a vast majority of the entire dataset.

Table 5.4 summarizes area under the curve computed from Figure 5.8. SRADL performed slightly worse on 6 chunk delay with 3.5% less area under the curve when compared to wait-and-train. On the 12-chunk delay, SRADL outperformed by 1.9%. On 18-chunk delay, SRADL scored the most performance gain, 7.5% more area than wait-and-train. The result shows that the SRADL semi-supervised learning is able to utilize

Table 5.4 Area under the curve for labeling Scenario 2 experiments with Spam.

Delay	Static	No-delay	Wait&Train	SRADL
6	473.1	1815.3	1412.7	1362.7
12	473.1	1815.3	1367.8	1394.6
18	473.1	1815.3	1295.0	1393.4

small amount of labeled data trickled in through time. Although limited in numbers, these labeled data enable better models to be trained when compared to the previous scenario. As the delay time lengthen from 6 chunks to 18 chunks, the advantage of better model earlier is reflected in the performance increase by SRADL.

Both synthetic and real-world experiment results showed that different labeling scenarios have different effect on SRADL and wait-and-train. For labeling process that return all the labels all together, wait-and-train is the better approach. Whereas for labeling process that can return small number of labels from time to time, SRADL performs better. SRADL also universally benefits from larger chunk delays since the naïve approach keeps the outdated models for longer periods of time in these cases.

CHAPTER 6. HEURISTIC ENSEMBLE FRAMEWORK FOR CONCEPT DRIFT DETECTION IN DATA STREAM CLASSIFICATION[5]

Majority of existing concept drift detection algorithms, while performing well with labeled data, may show limitations when presented with unlabeled data. The complex nature of concept drift brings many more types of drift other than the presence of distribution change (Minku et al., 2010) (Khamassi et al., 2019). A drift can occur abruptly, or slowly over time (Gama et al., 2014). Drifts can occur repeatedly, or occur periodically (Webb et al., 2016). Algorithms that focus on one aspect of concept drift characteristics will inevitably fail when such characteristic does not appear in a particular drift. For instance, an algorithm that focuses on detecting new data distribution will be able to detect change in Figure 6.1.c most of the time even without class labels. However, such algorithm might fail at detection concept drift in Figure 6.1.a, because the characteristic of change in distribution required by the algorithm does not occur. Another algorithm, such as SVM Margin (Dries & Rückert, 2009), might be able to detect changes in Figure 6.1.a but not in Figure 6.1.c. A better strategy is therefore to combine the strength of both algorithms so that both types of concept drift scenario can be detected.

[5] This chapter has been published at Hu and Kantardzic, 2022

This chapter proposes Heuristic Ensemble Framework for Drift Detection (HEFDD) in streaming data. HEFDD applies the theory of ensemble to bring together multiple algorithms, each performing best at detecting some, but not all, types of concept drifts. A novel hierarchical voting mechanism ensures that different types of concept drift can be detected, even when global voting does not produce a majority. Such voting mechanism also helps to reduce the number of false alarms raised by individual detection algorithms. Main contributions of this paper are:

- Formulate a heuristic ensemble framework for unlabeled detection of various types of concept drift.
- Outline criteria of algorithm selection for implementing the ensemble
- Experimentally show the advantage of the ensemble approach compared to variety of single detection algorithm.

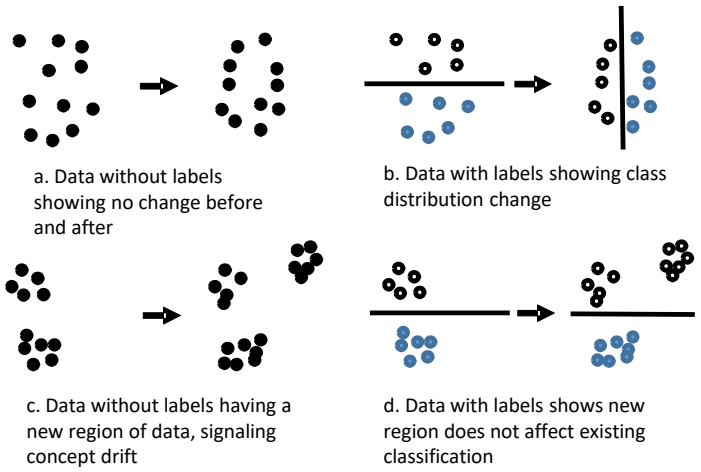


Figure 6.1. Detecting various types of concept drift is difficult without label

6.1 Heuristic Analysis on Related Concept Drift Detection Algorithm

As mentioned in Chapter 2, The majority of current state of the art concept drift detection algorithms can be divided into two groups: a) performance based and b) distribution based. Performance based approaches require labeled data, because they track

the performance of the classification result. Without labels it is difficult to obtain an accurate performance evaluation. Data distribution-based approach has the advantage of being able to process both labeled and unlabeled data. When distribution based approach works with labeled data, its detection can be on par with that of performance based approach (Dries & Rückert, 2009)(Sethi & Kantardzic, 2017). When working with unlabeled data, however, distribution-based approaches often have limits on what types of concept drift each can detect. This is because most of the techniques are primarily based on assumptions of single type of concept drift detection, while the other types are rarely detected. Since the study focuses on unlabeled concept drift detection, distribution-based approaches will be reviewed in detail. Distribution based approaches can be further divided into three groups: a) Statistical test based, b) density based and c) Learning model based. Algorithms in each group employs similar detection techniques, which means they have

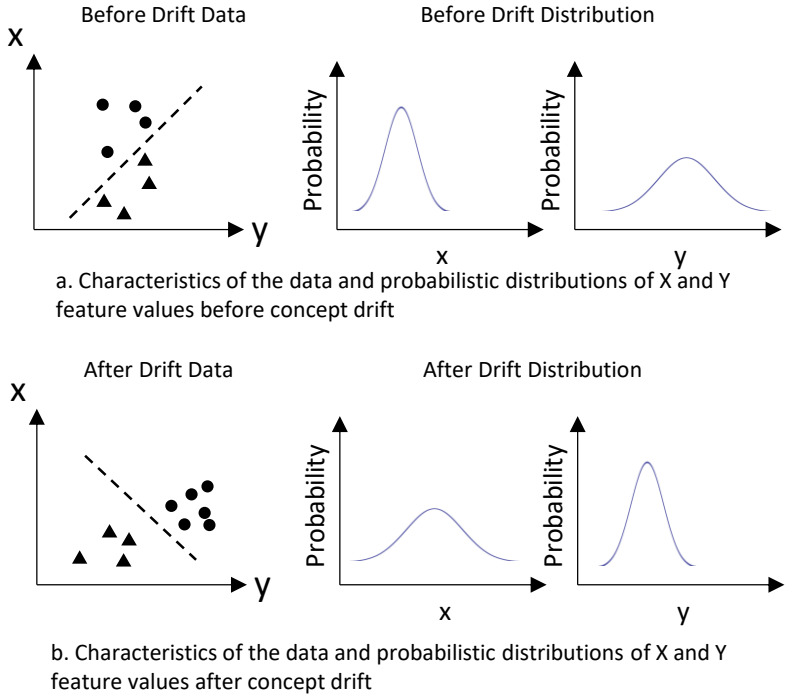


Figure 6.2. Basic principle of data distribution based statistical testing

similar limitations on the types of concept drifts they can detect. This section will examine their limitations on which types of concept drift each group of algorithm is able to detect.

6.1.1 Statistical Test Based Approaches

Statistical-test-based approach utilize statistical test to track significant data distribution change. This is illustrated in Figure 6.2 using two-dimensional data (X, Y). The actual underlying class boundary of the demonstrated two-dimensional data would be unknown if class labels are not available. However, there are changes in the probability distribution of the two features before and after the drift, as shown in Figure 6.2. If such features pass some statistically test to be significant different, then a concept drift can be detected without label of data samples. Kifer et al. (Kifer et al., 2004) applied Kolmogorov-Smirnov (KS) test for concept drift detection. Glazer et al. (Glazer et al., 2012) applied KS test for detection of change of high-density area in high dimensional data. The study modified classic minimum-volume set (MV-set) estimators for density estimation and enables KS test to be applied to high dimensional data. The author also noted that change in high density area is directly related to concept drift detection in streaming data. dos Reis et al. (dos Reis et al., 2016) modified KS test to be able to perform incrementally. Besides using KS test, Song et al. (Song et al., 2007) applied kernel density to detect concept drift. This study identifies suitable kernel width using expectation maximization algorithm. Kernel density test was performed on each data samples to check if it is from the same underlying data distribution. Siahroudi et al. (Siahroudi et al., 2018) computes multiple kernels of the data space and specifying class boundary using combined kernels. A concept drift is detected if new samples appear outside existing class boundary. Lee and Magoules (Lee & Magoules, 2012) utilized correlation information of value distribution in a

windowed detection approach. Faithfull et al. (Faithfull et al., 2019) applied an ensemble of univariate change detector to a multivariate change detection problem. The ensemble outperforms pure multivariate approaches in their experiments. Sobolewski and Wozniak (Sobolewski & Wozniak, 2013) proposed a KS test concept drift detection framework with that can work on unlabeled data stream. The approach acknowledges that concept drift that does not change the global distribution cannot be detected. This study thus demonstrated that KS test drift detector, when applied to unlabeled data, is not suitable for detecting fixed space drift

6.1.2 Density Based Approaches

Density based methods look for changes in dense regions of the data. These methods are capable of identifying uncertain suspicious samples, which need further evaluation. They define an additional 'Unknown' class label to indicate that these suspicious samples do not fit the existing view of the data (Spinosa et al., 2007). Clustering and outlier-based approaches are popular implementation strategies for detecting novel patterns, as they summarize current data and can use dissimilarity metrics to identify new samples (Kantardzic et al., 2010). Lazarescu et al. (Lazarescu et al., 2004) uses a multi-windowed approach with clustering for concept drift detection. The clusters were constructed to describe each current concept. When concept drift occurs, new clusters constructed after the drift will be significantly different from existing cluster and thus the drift can be detected. Spinosa et al. (Spinosa et al., 2007) applied K-means clustering algorithm for concept drift detection in a framework named n OnLIne Novelty and Drift Detection Algorithm (OLINDDA). K clusters were initially generated by the K-means algorithm. The overall arithmetic mean of distances between the initial cluster centroids

were calculated. As new sample arrives, new samples are clustered into a candidacy cluster. If the mean distance between centroid of this candidacy cluster and initial clusters are smaller than the initial mean distance, then this candidacy cluster is considered valid, and no concept drift occurred. Otherwise, the new candidacy cluster forms a new concept outside the initial K-clusters, forming a new dense data region. Kantardzic et al. (Kantardzic et al., 2010) proposed a framework that work with partially labeled data stream. Similar to OLINNDA, the framework creates initial clusters at beginning of the data stream. If a new cluster emerges from the data stream, it means a new dense region appeared and concept drift is detected. Masud et al. (Masud et al., 2010) proposed a framework where K-means clusters were used initially to create K clusters. New samples outside of existing clusters are counted as outliers. Similar to (Kantardzic et al., 2010), if the outliers form a new cluster, then either a concept drift or a novel class is detected.

The limitation of clustering-based approach is that if global data distribution does not create new dense regions, then the detection will fail. This is explained in detail using algorithm by Kantardzic et al (Kantardzic et al., 2010). The algorithm assumes that samples of the same class form clusters. The detection process is explained in Section 5.2.2. The case for detection failure for this algorithm is demonstrated in Figure 6.3. In Figure 6.3.b, all new samples fall within the existing cluster, which does not trigger drift

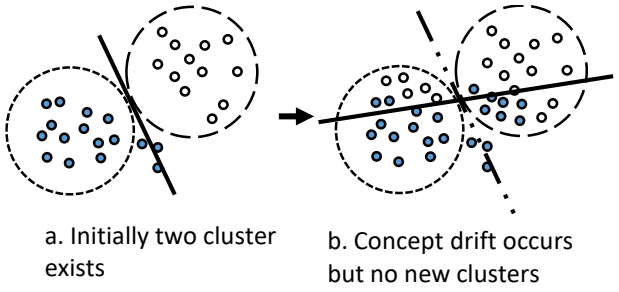


Figure 6.3 Clustering based concept drift fail to detect when drift is stationary.

detection. Without labels, it is also impossible to differentiate differences of the new samples from current samples within the cluster. Therefore, this approach cannot detect drift under this scenario.

Tu and Chen (Tu & Chen, 2009) proposed a framework that has an online component that map each data samples to a grid in the data space, and an offline component that compute grid density. A density decaying mechanism was applied to forget older samples so that new dense grid can be discovered. An ensemble-based grid density framework was proposed by Sethi et al. (Sethi et al., 2016) to tackle concept drift in both spatial and temporal component of the data stream. A grid of the data space is generated by dividing each feature of the data into fixed intervals, forming a grid. As samples arrive, they are assigned to their specific grid cell based on their feature values. The number of samples in each cell is recorded, and the cell becomes dense if the number rises beyond a threshold *p*. If at a moment a normal cell becomes dense, then it is clear that more data are arriving in the data space occupied by the cell, a potential concept drift. The grid detection has similar limitations to the clustering-based approach. The Grid Based Detection monitors changes in the shape of the data. If all new data only arrive in existing dense grids, then the overall shape of the data will not change. Since the data is unlabeled, it is impossible to track the shape of individual class of data. This limitation is shown in Figure 6.4. Figure 6.4.a shows the initial decision boundary between the two classes with three dense grids. Figure 6.4.b, new samples arrive mostly in existing dense grids, but clearly decision boundary is shifted. This scenario cannot be detected by Grid Based drift detection. For this reason, in an unsupervised environment, this algorithm is best when used for non-stationary drift

6.1.3 Learning Model Based Approaches

This group of detection algorithms use the underlying classification model's characteristics for concept drift detection. Dries et al. (Dries & Rückert, 2009) proposed three approaches in their study: a) density estimation on binary representation of the data, b) measures average margin of 1-norm SVM and c) average error rate generated by

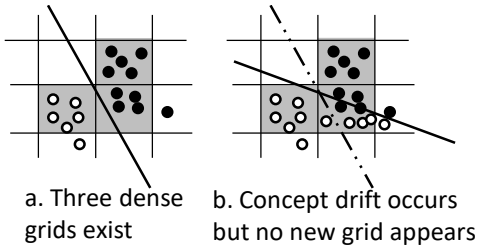


Figure 6.4. Grid Based Drift Detection failed to detect drift

SVM. The study shows that the SVM approach (second approach) has the best precision and recall. The linear SVM creates a margin between two supports. Concept drift can be detected if there are significant changes between the margins. Sethi et al. (Sethi & Kantardzic, 2017) further improve the margin approach by extending it to partially labeled data stream. The approach (MD3) computes the density within a margin created by decision boundary and support of a linear SVM trained on original training data. Once the SVM is trained, the detection process does not require labels. The margin density is calculated by equation (6.1).

$$\rho = \frac{\#\text{samples with abs}(w \bullet x_i + b) \leq 1}{\#\text{samples}} \qquad (6.1)$$

#samples is the number of data samples within the current time step in data stream. The separating hyperplane of the SVM is defined by $w \bullet \mathrm{x} + b = 0$. The function abs() calculate the absolute value. For each sample i, if $abs(w \bullet x_i + b) \leq 1$,

then it falls between the defined margins. The maximum and minimum ρ are monitored

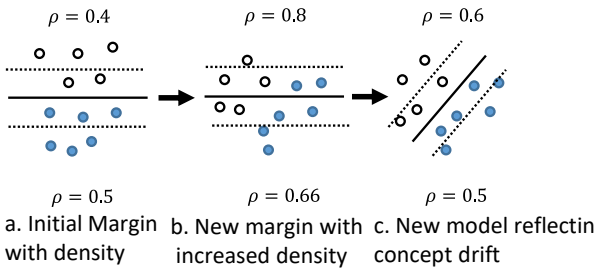


Figure 6.5. Illustration of Margin Density drift detection

for the current chunk. If ρ falls outside of the existing min-max range, then a potential drift is detected. The process is illustrated in Figure 6.5. Initially in Figure 6.5.a, two classes of samples are separated by a horizontal decision boundary, with margin density 0.4 (2 out of 5 samples inside margin) and 0.5 (3 out of 6 samples inside margin) for each class respectively. After concept drift occurs in Figure 6.5.b, the decision boundary changed, as a result the margin density changes to 0.8 (4 out of 5 samples inside margin) and 0.66 (4 out of 6 samples inside margin) respectively. After retraining SVM, the new decision boundary reflects post-drift reality in Figure 6.5.c, and the margin density returns to normal value of 0.6 and 0.5 respectively.

This approach is best used for stationary drift since the density is only calculated within a narrow region encompassing the decision boundary. The margin density will not change if data suddenly appear in a new region outside of the margin. This is shown in Chapter 2, Figure 2.16

6.2 Heuristic Ensemble Framework for Drift Detection (HEFDD)

Many distribution-based concept drift detection algorithms have limitations on which type of concept drift they are able to detect when labeled data aren't available. The

limitations are the result of their assumption about specific nature of the concept drift. For instance, Clustering and Grid approaches assume that concept drift occurs when new dense regions in the data space appear. But there are many cases where drift can occur in existing data regions. On the other hand, Margin Density approaches assume concept drift will be captured around existing data models, but there can be changes happening far away. Kolmogorov-Smirnov (KS) Test can detect samples from different populations but cannot identify changes from the same global population. The proposed framework, Heuristic Ensemble Framework for Drift Detection (HEFDD) aims to detect all types of concept drift in data streams by combining the strength of multiple concept drift detection algorithms. HEFDD divide data stream into chunks, where each chunk is a fixed number of data stream samples (Sethi & Kantardzic, 2017). This strategy is used because it is most flexible, as it can work with both chunk-based detection algorithm and incremental detection algorithm (dos Reis et al., 2016).

6.2.1 Ensemble and Heuristic Voting

The ensemble of HEFDD is consists of individual concept drift detection algorithms, illustrated in Figure 6.6.a. Each DD (DD1 to DD4) stands for a single Drift

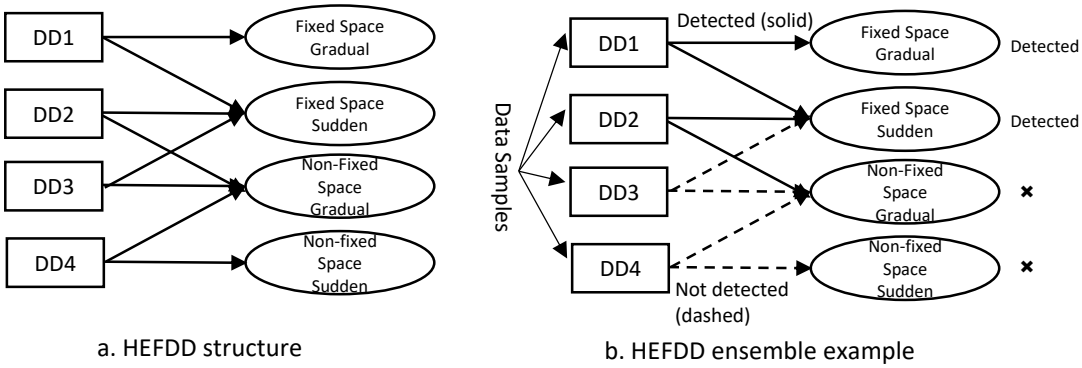


Figure 6.6. HEFDD Ensemble Structure

Detection algorithm. The arrows in Figure 6.6.a denote which concept drift type each DD is able to detect (DD1 is able to detect Fixed Space Gradual and Fixed Space Sudden, DD2 is able to detect Fixed Space Sudden and Non-fixed Space Sudden and so on). This is determined by analyzing theoretical approach used in a specific technique and on the experimental results of each concept drift detection algorithm. DD1 may have similar assumption on concept drift to Margin Density, and therefore it can only detect Fixed Space drifts. On the other hand, DD4 may have similar assumption to Clustering, so only Non-fixed Space drift can be detected. Each algorithm votes on the type of concept drift it can detect. Votes are not counted globally. Instead, each type of concept drift keeps track of votes from the subset of techniques that can detect it. In Figure 6.6.a, Fixed space Sudden and Non-fixed Space Gradual both potentially have three techniques' vote while Fixed Space Gradual and Non-Fixed Space Sudden both potentially have one technique's vote. As long as a concept drift type obtains a majority voting on its own total vote count, it is detected by HEFDD. This is further illustrated in Figure 6.6.b. As a chunk of data samples arrives, they are directed through each of the DDs. Solid arrows following the DD1 and DD2 means a concept drift is detected in the current chunk whereas dashed arrows following DD3 and DD4 means a concept drift is not detected. Based on heuristic voting, Fixed Space Gradual (FSG) is detected because it received one vote out of one total vote. Fixed space Sudden (FSS) is also detected because it received two votes out of three total votes, reaching a majority. Non-fixed space Gradual (NFSG) is not detected because it only received one vote out of three, and finally Non-fixed Space Sudden (NFSS) is also not detected by not receiving any vote.

6.2.2 Component Selection for HEFDD

The selection of algorithm that constitute the ensemble is based on two principles:

a) type of concept drift to be detected

b) heuristic analysis of each algorithm

In this study six types of concept drift are identified by combining the speed and distribution classifying criteria, outlined in Section 2. Ideally, there should be several detection algorithms in the ensemble that is able to detect each of the six concept drift types. To increase confidence of detection and reduce false positive, each concept drift type is better to have more than one algorithm's vote. In our review of algorithms in Section 3, we identified the limitations of Margin Density, Clustering, KS Test and Grid

Table 6.1. Different concept drift detection algorithms detect different types of drift

Datasets / Algorithm	FSS	FSG	FSI	NFSS	NFSG	NFSI
Margin Density (Sethi & Kantardzic, 2017)	YES	YES	YES	NO	NO	YES
Clustering (Kantardzic et al., 2010)	NO	NO	NO	YES	YES	YES
KS Test (Sobolewski & Wozniak, 2013)	NO	NO	NO	YES	YES	YES
Grid (Sethi et al., 2016)	NO	NO	NO	YES	YES	YES

concept drift detection algorithms, which are summarized in Table 6.1. Margin Density can detect all Fixed Space concept drift while Clustering, KS Test and Grid can detect only Non-fixed Space drifts. An ensemble of these algorithms is therefore able to detect all six types of concept drift. At the same time, non-fixed space type drifts will have three algorithm voting for improved detection through the voting process. If two algorithms reach a majority vote, then there is a higher confidence that such type of concept drift is truly happening.

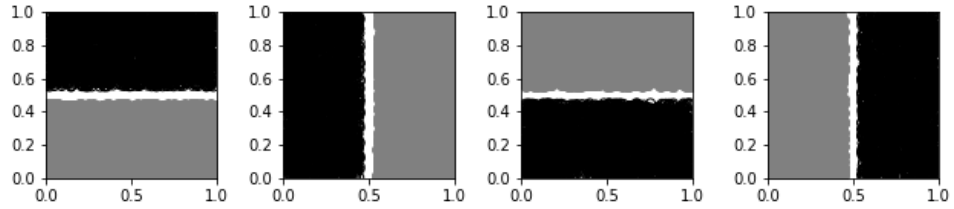
6.3 Experiments

Experiments were conducted using both synthetic and real-world dataset. Synthetic datasets were created so that each of the proposed type of concept drift can be accurately simulated. These datasets were used to confirm our analysis of each individual algorithms in Table 6.1 and prove the concepts of HEFDD approach. Real-world datasets were used to show the advantage of HEFDD methodology when compared to a detection strategy using only a single drift detection algorithm.

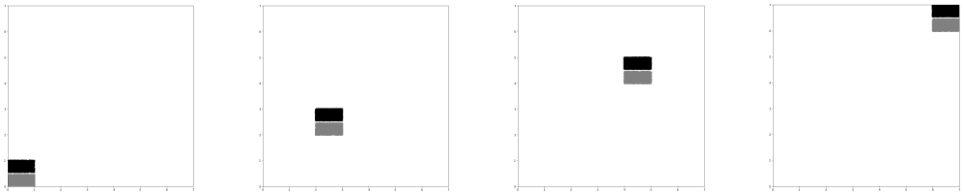
6.3.1 Datasets

Synthetic dataset was 2-dimensional input with the third dimension a binary class label. Fixed space and non-fixed space drifts were created as illustrated in Figure 6.7. Fixed space data were created with normalized input in range [0, 1]. A margin width of 0.1 was defined between the two classes of samples. The margin rotates in place as more streaming samples were generated. Non-fixed space data were created by moving the data generation range from [0, 1] toward [6,7]. In both fixed and non-fixed drifts, the rate of change is also modified to create abrupt, gradual and incremental drift. In abrupt drifts, all margin rotation and data movement are completed in one chunk of data, while in gradual and incremental drift they take three chunks to complete. In total, 6 separate synthetic data sets were created, covering all six concept types in Table 6.1. Each dataset has 20 chunks, where each chunk has 500 data samples. The speed of changes is generated by controlling how fast concept drift occurs. In the sudden drift datasets (FSS and NFSS), three separate concept drifts occur in chunk 5, 10 and 15. In the gradual and

incremental drift datasets, three separate concept drifts occur from chunk 4 to chunk 6, from chunk 9 to chunk 11 and from chunk 14 to chunk 16.



a. Fixed space drift was created by rotating the class boundary in place



b. Non-fixed space drift was created by moving the data to a new area in the data space

Figure 6.7. Synthetic data with fixed-space and non-fixed space drift

Three real-world datasets were used in the experiments. Electric Market (EM) dataset is a real-world dataset that keeps track of the rise and fall of electricity price (Zliobaite, 2013). Forest cover (Covtype) dataset contains forest cover type data from US Forest service (Blackard & Dean, 1999). Covtype was modified into a binary class dataset by selecting two classes with the most samples. SPAM dataset is a text dataset

Table 6.2. Dataset used in experiments

Name	# Samples	# Input Attribute	# Classes	Chunk Size
Synthetic (FSS, FSG, FSI, NFSS, NFSG, NFSI)	10,000	2	2	500
EM	45312	7	2	1500
Covtype	218515	54	8	2000
SPAM	6213	500	2	500

from SpamAssasin data collection (Blake & Merz, 1998). It contains a numerical representation of 6213 emails where some of these are spam emails. Although all available datasets contain class labels, the drift detection process does not take them into consideration to simulate unlabeled streaming data. Table 6.2 summarize the characteristic of each dataset and the size of data chunk used in the experiments.

6.3.2 Experimental Results

Table 6.3 shows the experimental results of four concept drift detection algorithm and their EFDD ensemble using synthetic data. For the fixed-space datasets, only Margin Density algorithm is able to detect any concept drift, which is expected result based on analysis in section 3. Since Margin Density constitute the only vote for fixed-space drift,

Table 6.3. Different concept drift detection algorithms detect different types of drift in synthetic data sets

Datasets / Algorithm	FSS	FSG	FSI	NFSS	NFSG	NFSI
Actual Drifts	5, 10, 15	4-6, 9-11, 14-16	4-6, 9-11, 14-16	5, 10, 15	4-6, 9-11, 14-16	4-6, 9-11, 14-16
Margin Density (Sethi & Kantardzic, 2017)	5, 10, 15	4, 5, 10, 15	4, 6, 9, 10,11, 14, 16	-	-	-
Clustering (Kantardzic et al., 2010)	-	-	-	5, 10, 15	4, 9, 14	4, 5, 6, 9, 10, 15, 16
KS Test (Sobolewski & Wozniak, 2013)	-	-	-	5, 10, 15	4, 9, 14	4, 6, 10, 11, 14, 15
Grid (Sethi et al., 2016)	-	-	-	5, 10, 15	4, 9, 14	4, 5, 6, 9, 10, 11, 14, 15
HEFDD	5, 10, 15	4, 5, 10, 15	4, 6, 9, 11, 14, 16	5, 10, 15	4, 9, 14	4, 5, 6, 9, 10, 15

EFDD produces the same detection of this drift type as Margin Density. Among the three

drift speeds, Margin Density is able to detect all cases of sudden drift. It has some detection delay and redundant detection for the FSG and FSI dataset. In the non-fixed space dataset, Clustering, KS Test, and Grid are able to detect any drift, also as expected. All algorithms detect NFSS and NFSG drifts without delay or redundancy. However, in NFSI, both algorithms created many redundant detections. EFDD is able to reduce the number of redundant detections requiring that at least two algorithms reaching a majority.

The exact location of concept drifts for synthetic data is determined in advance, but this is not the case for real-world dataset. To overcome this problem, a performance-based algorithm DDM (Gama et al., 2004) is chosen as the golden standard since DDM is able to detect all types of concept drift using labeled samples. Table 6.4 shows concept drift detection for real-world data stream comparing with DDM's detection result using labeled data, considered to be gold standard for the other five unlabeled drift detection methodologies. The detection column shows the chunk number where drift is detected, and the accuracy column shows the corrected detection compared to DDM. Looking at the exact chunk number, HEFDD only correctly detected two chunks out of five from DDM. However, different algorithms have different sensitivity for concept drift. In addition, gradual and incremental drift may have a delayed detection because the change is very slow. For this reason, it is more useful to also consider adjacent chunks detection as hit when comparing drift detection result. That is, if chunk detection is only off by one chunk when compared to golden standard approach DDM, it is still considered the same detection. To better show detections that are adjacent, a notation A(B) is adopted where A is chunk number detected by any algorithm other than DDM and B is chunk number

detect by golden standard DDM. The accuracy column is calculated with adjacent chunk included.

Table 6.4. Experimental Result on Real-world Datasets

Datasets / Algorithm	Electric Market		Spam		Covtype	
	Detection	Accuracy	Detection	Accuracy	Detection	Accuracy
DDM (Gama et al., 2004)	4, 7, 10, 13, 24	100%	4, 10	100%	5, 9, 13, 16, 20, 23, 32, 34, 40, 42, 47	100%
Margin	5	20%	1, 4, 12	50%	4, 5, 8, 10, 18, 24, 42	36.3%
Clustering	1, 7, 10	40%	2, 6, 7, 10	50%	14, 17, 29, 33, 36, 38, 39, 40, 44, 46	54.5%
KS Test	2, 7, 10	40%	4, 7	50%	5,7, 10, 18, 22, 23, 27, 33, 39, 47	54.5%
Grid	7, 8, 9, 10	40%	7, 9, 10	50%	12, 13, 14, 20, 22, 26, 30, 33, 39, 40, 43	63.6%
HEFDD	5, 7, 10	60%	1, 4, 7, 10, 12	100%	4, 5, 8, 10, 14, 18, 33, 39, 40, 42	63.6%

Table 6.5. Impact of concept drift detection on data stream classification performance

Datasets / Algorithm	Electric Market	Spam	Covtype
DDM	75.2%	69.2%	63.3%
Margin	58.8%	68.2%	56.1%
Clustering	66.7%	65.2%	62.4%
KS Test	62.3%	58.2%	59.9%
Grid	64.6%	59.7%	63.1%
HEFDD	68.8%	68.3%	63.1%

Each real-world dataset was classified chunk by chunk using a SVM with RBF kernel. In the first chunk of the data stream, an initial SVM was trained. A new SVM would replace the old one when concept drifts were detected in each chunk. On the chunks that does not detect concept drift, accuracy of the SVM predictions were measured and the average accuracy of the entire data stream were recorded. Table 6.5 shows the results.

6.3.3 Discussion

The synthetic dataset shows that the heuristic analysis of each algorithm matches their respective concept drift detection types. Margin density is able to detect all Fixed Space types but fails on all non-fixed space types. Clustering, KS Test and Grid is able to detect all non-fixed space types but fails on Fixed Space types. The speed of drift impacts the accuracy and precision of each detection algorithms. Ideally, concept drift detection algorithm should be able to detect a concept drift as early as possible and at the same time maintain a high precision and accuracy of detection. For example, in FSI ideal detection should detect concept drift at the beginning of chunk 4, 9 and 14. Margin density has chunk 6, 11, and 16 as redundant drift detection since they are the same drift started in chunk 4, 9 and 14. In real-world applications, because concept drifts are detected in these chunks, a streaming classification framework will likely spend resources on training new models. HEFDD helps by reducing the redundant detection in NFSI so that less resources are wasted.

The Electric Market column of Table 6.4 shows detection results for the EM dataset. After taking into account of drift sensitivity, EFDD is able to detect three drifts compared to DDM at chunk 5(4), 7(7) and 10(10). HEFDD detections at chunks 7(7) and 10(10) came from detection majority by Clustering, KS Test and Grid while chunk 5(4) is contributed by Margin. The EM stream dataset shows mostly non-stationary drift in the beginning, while stationary drift starts to appear at the middle of the stream. This confirms our hypothesis that real-world datasets may contain more than one type of concept drifts. Together, HEFDD is able to detect 60% of DDM's detections, more than any individual algorithms that made up the HEFDD ensemble. The Spam column in

Table 6.5 shows Margin density correctly detected 4(4) with 12(10) as near miss being two chunks away from chunk 10. Clustering, KS Test and Grid produced majority at chunk 7 and a correct detection at chunk 10(10). Together, EFDD detected 100% DDM's detection at 4(4) and 10(10), but with false positive at chunk 1, 7 and 12. The Covtype colum in Table 6.5 shows Margin density correctly detected 4(5), 5(5), 8(9), 10(9), 24(23) and 42(42). Since chunk 4, 5 correspond to DDM detection of chunk 5 and chunk 8, 10 correspond to DDM's chunk 9, these four detections by margin density really are detecting only two drifts. EFDD is able to detect more drift than Margin Density and Clustering and detect the same number of drift with Grid. Counting false positive, EFDD has the lowest number of false positives compared to the other three algorithms. Averaging three dataset's detection accuracy, statistical p-value test using z-score shows HEFDD has significance improvement at $p < 0.05$.

Figure 6.8 summaries and compares the result of all four experiments. Figure 6.8.a plots the accuracy of correctly detected concept drift as listed in Table 6.4. In most cases EFDD is able to detect more drifts than any individual algorithm. Exceptions are in

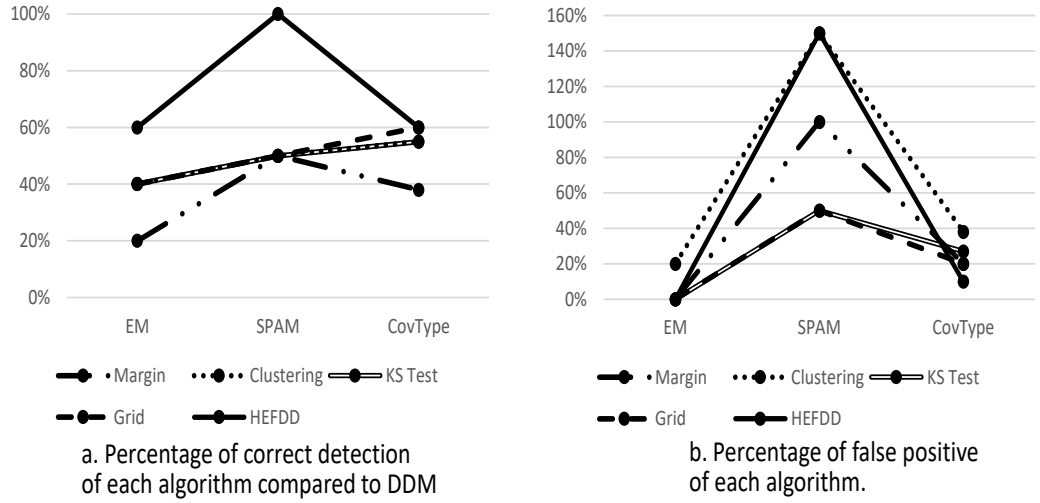


Figure 6.8. Summary of experimental results

hyperplane where EFDD detected the same number as Margin Density and in Covtype where EFDD equals Grid. Figure 6.8.b shows the percentage of false positive detected by all four algorithms. The percentage is produced by dividing number of false positive detection by number of drifts detected by DDM. For instance, for hyperplane, there is one false positive by Margin Density and four drift detected by DDM. The false positive percentage is therefore 25%. The result shows the ability of EFDD to reduce the number of false positives compared to individual algorithms. In hyperplane, EFDD has more false positives than Cluster, KS Test and Grid because the latter three algorithms could not detect any drift. For real-world data sets, EFDD is able to reduce false positive in EM and Covtype dataset.

The goal for achieve high concept drift detection performance is to increase the prediction performance of data stream classification framework. Average accuracy in Table 6.5 shows HEFDD can achieve higher accuracy when compared to individual algorithms. Column Electric Market shows that DDM achieve the highest accuracy while HEFDD achieve the best performance when compared to individual drift detection algorithm. It is able to outperform Margin Density by 10% because the latter is only able to detect one concept drift. The majority of EFDD's detection in this dataset comes from Clustering and Grid and thus its performance is similar to the other two. It is still much lower than DDM because EFDD failed to detect a concept drift later in the stream. Column Spam shows each algorithm has similar overall performance with HEFDD performing slightly under DDM but above all other algorithms. The SPAM dataset has an important drift at chunk 4, therefore any algorithm that detects it relatively early will

have good performance. Covtype column shows Margin Density has the lowest performance because it fails to detect most of the concept drift.

CHAPTER 7. IMPROVING PERFORMANCE OF SRADL IN DELAYED LABELING DATA STREAM MINING USING HEFDD FRAMEWORK

This chapter aims to continue to refine the Sliding Reservoir Approach for Delayed Labeling (SRADL) framework that is developed for delayed labeling problem in Chapter 5. The experiment results in Chapter 5 shows that using only semi-supervised learning and historical labeled data, the improvement on classification performance is not significant. Noticeable increase in classification accuracy occurs only when a small portion of newly labeled data is received in the continuous-labeling scenario. As discussed in Chapter 5, although newly unlabeled data contain information about data distribution after concept drift, this information is unknown to the learner because it is unlabeled. Historically labeled data contains information before concept drift. Semi-supervised learners infer data distribution information only from labeled data. Therefore, when new labels are not available, SRADL is still training models using mostly information before concept drift. Furthermore, the original SRADL framework utilizes the clustering-based concept drift detection algorithm. The algorithm, as shown in Chapter 6, can only detect non-stationary drift.

To improve SRADL, it is necessary to integrate more up-to-date information of the concept drift to the learning process. The framework Heuristic Ensemble Framework for Drift Detection (HEFDD), introduced in Chapter 6, can be used for this purpose. HEFDD

is able to not only detect concept drift, but also provide insights on what type of concept drift is detected. With this knowledge, a framework can potentially have different strategies for each concept drift type. For instance, a non-stationary drift calls the framework to look for changes in global distribution whereas a stationary drift makes the framework to respond to local changes. Some concept drift types might not even need new models but new preprocessing of the data, as shown in Chapter 4.

This chapter proposes an improved SRADL framework that integrates HEFDD as concept drift detection unit (SRADL-HEFDD). After the type of concept drift is determined, the SRADL-HEFDD framework will build model differently for different types of concept drift. The goal is to increase the performance of SRADL framework when new labels are completely unavailable, the delayed labeling scenario one in Chapter 5.

7.1 SRADL-HEFDD Framework

The SRADL-HEFDD continues to employ a chunk-based approach, meaning samples will be processed through the framework in fix-sized groups. The chunk-based approach is selected because of it can be used readily to construct a temporary understanding of the current data environment within the data stream. The disadvantage of the chunk-based approach is the selection of chunk size, which ultimately can have significant impact on the framework performance. The overall components of the SRADL-HEFDD framework are shown in Figure 7.1. The HEFDD unit replaces the concept drift detection unit in original SRADL. Once concept drift is detected, it triggers labeling, which request human expert to label newly arrived samples. At the same time, historical samples are stored in the Labeled Sample Reservoir to be used in Semi-supervised learning. The concept drift also triggers new model training. Different from original SRADL, new model

is trained after a training strategy is selected. In SRADL-HEFDD, HEFDD provides information on concept drift types. Based on characteristics of the concept drift, a model training strategy best suited for the current concept drift type is selected. Currently, the strategies are not automatically generated but are curated by human expert. This means that for a selected group of few concept drift types, a model training strategy is developed by hand then integrated into SRADL-HEFDD.

The difficulties of having a fully automated strategy selection are that sometimes the underlying assumption of data distribution, required by semi-supervised learning unit, changes when different concept drift types are involved. For instance, in non-stationary drift, semi-supervised learn may assume that samples belong to the same class will form distinctive clusters. This assumption helps assign temporary labels to unlabeled samples when they arrive outside of current distribution and are dense enough to be detected by

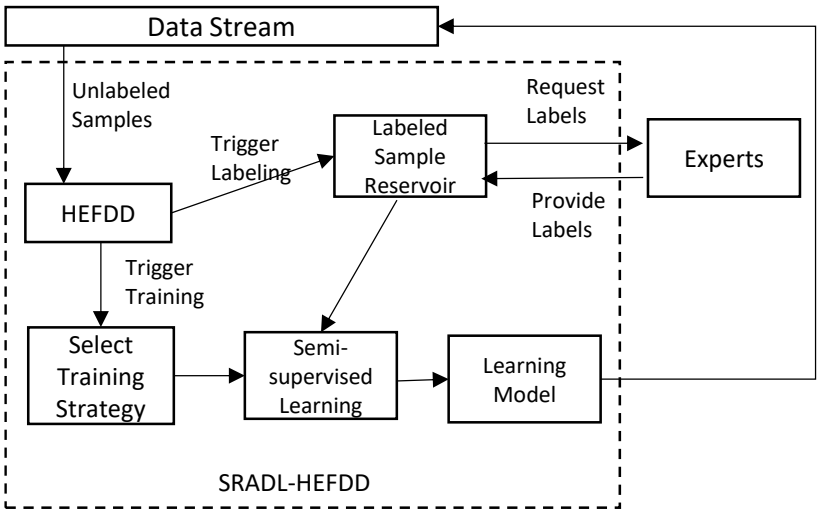


Figure 7.1. Overview of SRADL-HEFDD

clustering-based detection algorithm. In stationary drift, this assumption is useless because there is no new cluster being created after the concept drift. In order to create an automated strategy selection, a set of unified assumption needs to be created for concept drift and

semi-supervised learning. It is however possible to select a set of concept drift types that cover most of the cases. This proposal tries to formulate strategies for stationary versus non-stationary concept drift. The following two sections will discuss the proposed strategy in detail.

7.1.1 Synthetic Labeling Strategy for Fixed Space Concept Drift

K Nearest Neighbor (KNN) labeling strategy (Kang et al, 2006) is employed for fixed space concept drift. The strategy takes the majority label among the labeled "neighbors" and assign it to the unlabeled data sample. The neighbors are defined to be the Kth closest labeled samples to the unlabeled sample. The process is illustrated in Figure

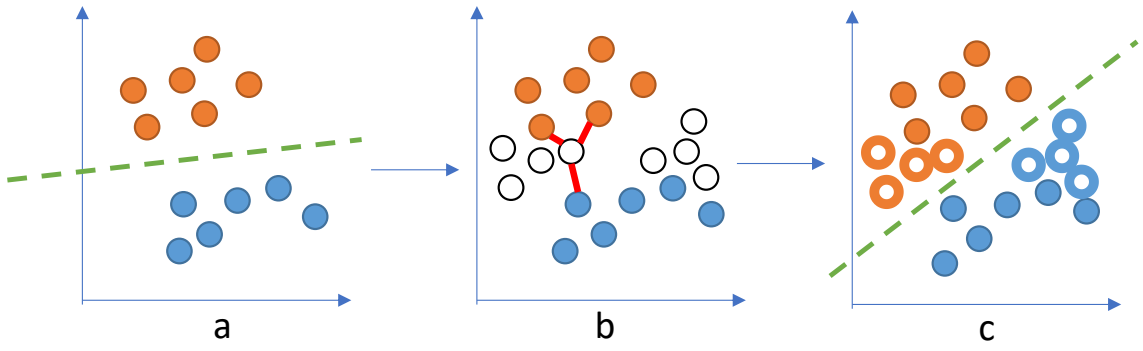


Figure 7.2. Illustration of KNN synthetic labeling strategy using K = 3

7.2. When new unlabeled samples arrive in Figure 7.2.b, one of the unlabeled samples has 3 labeled samples as its neighbors (marked by red line). Since the orange class is the majority label in the neighborhood, the unlabeled samples is then synthetically labeled as orange.

In a fixed space concept drift, the overall shape of the data distribution does not change much. The classification boundary rotates within the dataspace (Figure 2.4). New unlabeled samples will arrive near existing labeled samples, allowing accurate estimate of

labels based on its neighborhood. The better the learning model separates the two classes of samples, the more accurate synthetic labels can be obtained from the KNN strategy.

7.1.2 Synthetic Labeling Strategy for Non-fixed Space Concept Drift

KNN strategies might not work well for non-fixed space drift because of the case illustrated in Figure 7.3. The unlabeled samples are a continuation of the orange class

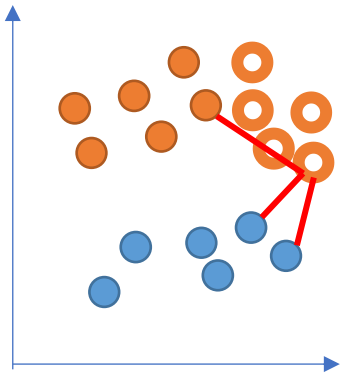


Figure 7.3. KNN synthetic labeling strategy using K = 3 potentially fails to synthetically label unlabeled data in non-fixed space concept drift

samples. If KNN is applied, however, the end of the continuation will result in data samples been assigned to the blue class. To avoid such case, assigning labels based on maximizing similarity matrix or minimizing dissimilarity matrix is chosen as the synthetic labeling strategy for non-fixed space drift. A similarity matrix computes the pair-wise distance between unlabeled samples and labeled samples (Fritsch & Ickstadt, 2009). Ideally, an unlabeled sample should be synthetically assigned with the same label as its most similar labeled counterpart (Wang et al, 2013). The label assignments that maximize the similarity matrix thus meets such requirement. There are many ways to maximize similarity matrix or minimize dissimilarity matrix. Figure 7.4 demonstrates synthetic labeling using minimum spanning tree for minimizing dissimilarity matrix (Want & Zhang, 2007).

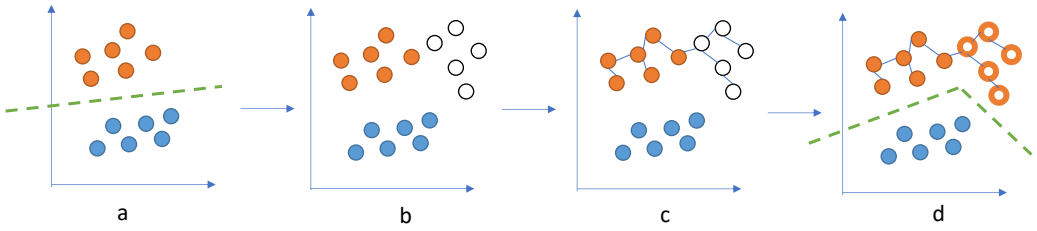


Figure 7.4. Demonstrate similarity matrix labeling strategy using minimum spanning tree

When unlabeled data arrive in Figure 7.4.b, a dissimilarity matrix is calculated. To minimize the matrix, a minimum spanning tree are constructed in Figure 7.4.c. The final synthetic labels are assigned according to the connection within the minimum spanning tree in Figure 7.4.d

7.2 Experiments

Experiments were conducted using stream data set Hyperplane (Fan, 2004) and

Table 7.1. Datasets used in experiments

Name	# Samples	# Input Attribute	# Classes	Chunk Size
Hyperplane	10000	10	8	300
SPAM	6213	500	2	500

Spam Assassin (SPAM) (Blake & Merz, 1998). The dataset profiles are shown in Table 7.1. Preprocessing was performed the same as HEFDD experiments in Section 6.3.

7.2.1 Experimental Setup

SRADL-HEFDD will be compared against two other approaches for delayed labeling problem: SRADL and wait-and-train. SRADL without HEFDD improvement uses S3VM for semi-supervised learning. More detailed description of SRADL is in

Chapter 5. Wait-and-train is the default approach for delayed labeling, where the framework waits for all labeled data to be available before training a new sample. Each approach will be tested using the two datasets. Classification accuracies will be measured data chunk by data chunk. The final average accuracy of all data chunks will be used for comparison.

Two scenarios of the delayed labeling problem will be implemented (Figure 5.1). Scenario 1 has all labels available after a fixed delay period while scenario 2 has labeled data trickle in throughout the delay period. Delay period is measured by how many data chunks has passed between concept drift is detected and labeling is complete. Three different delay periods, 6-chunk, 12-chunk, and 18-chunk are also tested in the experiment.

7.2.2 Experiment Result

Table 7.2 shows the result of Hyperplane dataset in both delay scenarios. In labeling scenario 1, wait-and-train outperform SRADL and SRADL-HEFDD. This is mostly consistent with results obtained in Table 5.1. The results further demonstrate that a total lack of labels after concept drift do not benefit semi-supervised learning or synthetic labeling. In labeling scenario 2, SRADL and SRADL-HEFDD could outperform wait-and-train in the 12 and 18 chunk delay experiments. This means that a small number of labeled samples after concept drift can improve semi-supervised learning. Notably, SRADL-HEFDD is able to produce better prediction accuracy than the default SRADL. In all experiments in both scenario 1 and 2, SRADL-HEFDD consistently outperforms the default SRAD. The largest gain made by SRADL-HEFDD against SRADL is 2.9% in 18-chunk delay scenario 1 experiment.

Table 7.2. SRADL-HEFDD, SRADL and Wait & Train average accuracy using hyperplane dataset

a. Delayed Labeling Scenario 1			
Algorithm	Average Accuracy for Different Delay Length		
	6 chunk	12 chunk	18 chunk
Wait & Train	76.4%	71.5%	68.5%
SRADL	72.4%	68.1%	64.3%
SRADL-HEFDD	73.4%	69.3%	67.4%
b. Delayed Labeling Scenario 2			
Algorithm	Average Accuracy for Different Delay Length		
	6 chunk	12 chunk	18 chunk
Wait & Train	76.4%	71.5%	68.5%
SRADL	73.3%	71.9%	70.9%
SRADL-HEFDD	75.7%	73.3%	72.4%

Table 7.3 summaries the results from the SPAM dataset experiments. Consistent with Table 5.3, SRADL and SRADL-HEFDD had equal or slight worse performance than wait-and-train in labeling scenario 1. However, SRADL-HEFDD still was able to perform better than the default SRADL in the 6-chunk and 12-chunk experiments. In labeling scenario 2, both SRADL and SRADL-HEFDD outperformed wait-and-train significantly. SRADL-HEFDD outperforms SRADL consistently throughout all experiments, with largest gain of 11.2% in the 18 chunk delay experiments.

7.3 Discussion

The above results show that SRADL-HEFDD, like SRADL, could not produce better predictive models when there are no new labeled samples available, as in delayed labeling scenario 1. Although in theory the synthetic labeling strategy can produce helpful labeled samples, it is proven difficult to perform well in a complex data set. As

Table 7.3. SRADL-HEFDD, SRADL and Wait & Train average accuracy using SPAM dataset

c. Delayed Labeling Scenario 1			
Algorithm	Average Accuracy for Different Delay Length		
	6 chunk	12 chunk	18 chunk
Wait & Train	50.2%	24.8%	20.6%
SRADL	46.2%	20.7%	20.6%
SRADL-HEFDD	49.3%	21.0%	20.7%
d. Delayed Labeling Scenario 2			
Algorithm	Average Accuracy for Different Delay Length		
	6 chunk	12 chunk	18 chunk
Wait & Train	50.2%	24.8%	20.6%
SRADL	61.2%	49.8%	44.3%
SRADL-HEFDD	64.9%	53.5%	55.5%

discussed in Chapter 5, historical labeled samples could not represent the current data environment after a concept drift. If the synthetic labels could not be assigned accurately, then SRADL-HEFDD will not outperform wait-and-train.

What SRADL-HEFDD improves on compared to the default SRADL is the ability to capitalize on small number of labeled samples in delayed labeling scenario 2. In all experiments in scenario 2, SRADL-HEFDD consistently outperforms SRADL. This shows that the synthetic labeling strategies, being able to react to different types of concept drift, can outperform default semi-supervised learning. It thus demonstrates again that No Free Lunch Theorem exist in data stream mining.

CHAPTER 8. EXPLAINABLE DATA STREAM MINING: WHY THE NEW MODELS ARE BETTER

Previous chapters introduced multiple frameworks to tackle concept drift. These frameworks react to changes in data stream by updating the learning model using after concept drifts are detected. Such strategy is common among many other data stream mining frameworks (Wares et al, 2019). The performance measure, being accuracy or f-score, usually indicates that dynamic models, updated after detected concept drifts, performs better than static models. There is little understanding in how this improvement is achieved. The process is akin to a black box, where the inner working of the machine learning algorithms and the frameworks is unknown.

Many explainable AI/ML algorithms have been developed in recent years. The majority of the studies focus on explain models for static data mining tasks, which involves only to explain how a decision is made through the prediction process. In data stream application, because the data is dynamic, it is also important to explain how the data is changed and how the model reacts to the changes in data, call change explainability. There are several problems that are unique to explainability of data stream mining compared to static model explainability. One problem is how to explain that the updated model is better fit for the data after concept drift compared to the old model. Such explanation will help to determine in what way the updated model improves from the old model within the current

data environment. Another problem is how to explain what triggered the detection of changes in the data. Since concept drift detections are mostly done using algorithms (Agrahari and Singh, 2021), the validity of these detection should also be explained to avoid false-positive detections. A third problem is when concept drift is absent or not detected, how to explain whether the current model still works well, because false-negative concept drift detections are possible.

This chapter formalize a visualization framework that aims to solve the above problem. Current model explanation is produced using an improved version of current explainable machine learning frameworks. Key metrics from the explanation of current and previous models are then compared. The comparison results are then compiled into visualizations that explain what has changed between the two models. Our previous study, which provide explanation of what has changed in the data stream itself, will meet the third requirement. By showing changes in the underlying data distribution together with what has changed between models, the framework can produce explanation on what triggered the changes in the model

8.1. Related Work

Several methods and strategies have been proposed to explain the data mining process. According to Adadi, A., & Berrada, 2018, these strategies can be classified into three categories: complexity related, scope related and model related explainability.

Complexity related strategies achieve interpretability by employing less complex machine learning methods. The less complex the methods are, the easier it is to be interpreted by users. Letham et al., 2015, proposed a model called Bayesian Rule Lists

(BRL). The model is based on decision tree. It is to provide a interpretable model to be used by domain experts. Caruana et al., 2015, described an application for the pneumonia problem using a learning method based on generalized additive models. By using real medical data and case studies, they showed the interpretability of their method. Xu et al., 2015, proposed a model for describing the content of images. The model is attention based where the most important features of the image are emphasized. Visualizations were used to show how their result can be explainable. Ustun & Rudin, 2016, presented a data-driven scoring system called SLIM. SLIM used sparse linear model to score a certain decision from a machine learning system. The scores are used to provide users with qualitative understanding of the system.

Scope related strategies achieve explainablility by either trying to understand the entire model behaviour, or by understanding each sample of data and corresponding predicted result. The former is called Global Interpretability and the later Local Interpretability. For global interpretability, Kuwajima et al, 2019, proposed a framework that improves interpretability of deep learning neural networks. The framework uses features analysis to understand the inference process within a deep learning framework. Yang et al., 2018, proposed a global interpretation model based on interpretation tree built using recursive partitioning, called GIRP. Their experiments show that their method is able to discover whether a learning model is overfitted to unreasonable degree. Zupon et al., 2019, proposed an approach that provides a global, deterministic interpretation by combine traditional bootstrapping model learning with explainable method. Their method is able to make representation learning interpretable. Nguyen et al., 2016, proposed an approach for image recognition. The approach is based on activation maximization,

which can generate preferred input for neurons in a neural network. The generated input is then used to interpretable model for image recognition. For local interpretability, Ribeiro et al., 2016, proposed Local Interpretable Model-Agnostic Explanation (LIME). LIME is used to approximate a black-box learning model locally in the area of interest. Thus, for a small number of decisions, LIME can explain the decisions of a previously difficulty-to-explain model. Another framework called anchors (Ribeiro et al., 2016) extends LIME using decision rules. Lei et al, 2018 proposed leave-one-covariate-out (LOCO) technique. LOCO is used to generate local models to measure local feature importance.

Model related strategies are explainability methods that either explain a single type of model (model specific) or can be applied to any machine learning models (model agnostic). Many interpretability methods for critical applications uses model specific methods. Ghosal et al, 2018, proposed an explainable deep vision network to identify crop stress and disease. The framework not only can predict but also explain which visual symptoms are used for prediction. Lundberg et al., 2018, presented an explainable framework for preventing hypoxaemia during surgery. The prediction model provides risk factors in the decision-making process in real time during general anaesthesia. Their experiments suggest that these risk factors are consistent with current understanding of anaesthesia. Klauschen et al, 2018, proposed an approach to estimate the effect of lymphocytes on tumor. The approach uses heat map to show where the most impactful area of the image is for decision making. Lee et al, 2019, applied an explainable deep learning algorithm for detecting acute intracranial haemorrhage from small datasets. The prediction results are integrated with an attention map and a prediction basis from

training data to enhance explainability. Lundberg et al, 2017, proposed a unified approach for explainable AI. The approach is in parallel from model training. Once a learning model is trained, the same training data, along with the learning model, is given as input to their framework named Shapley Addictive Explanations (SHAP). The framework dissects each prediction made by the model and provide an explanation based on feature importance.

For model agnostic method, the approach itself is independent from any models used. Model agnostic methods include visualization, knowledge extraction, and influence method. Visualization aims to understand the decision process through illustrative graphs. Knowledge extraction aims to generate rules from models that do not originally provide rules, such as artificial neural networks. Kuo et al, 2018 proposed an approach that utilized domain expert in explaining pattern recognized from clinical data mining. Their approach uses association rule mining and Bayesian Networks for explanation generation. Zhang et al, 2014, proposed an explainable framework for recommendation system. The framework, called Explicit Factor Model (EFM), extract product features. When recommendation is made based on user sentiment analysis, these features are given a score and used explain the reason behind recommendation. McInerney et al, 2018, formulate a recommendation engine that provides explainable exploitation and exploration aspect of a recommendation. The framework aims to balance suggesting similar product (exploitation) and provide new product (exploration) while explain the reason behind each decision. Hu et al., 2018, proposed an explainable neural computing framework using stack neural model networks. The framework uses modules of sub neural network to break down prediction tasks into subtasks. The process of analysing

subtasks provide users insight into how intermediate results are obtained, therefore enhance explainability compared to tradition neural network. Previously mentioned SHAP method can also belong to this category. SHAP has a component that analyses feature importance for each sample and calculate how much each feature contribute to the decision.

8.2. Data Stream Explainability (DSE) framework: Explaining Changes in Data Stream Mining Models

The DSE framework produces visualization that utilize the SVM margin to show how concept drift makes old model in the stream outdated and how new model reflects the current data environment better. The DSE is a chunk-based framework, which split streaming data into fixed sized groups of samples, called chunks. The framework will detect concept drift in a chunk and will decide that the existing model no longer reflect the underlying data distribution. Thereafter, a new model is trained using the current data after concept drift and the old model is replaced. DSE thus tries to visualize and explain what has happened within each chunk.

8.2.1 Visualizing Margin Density

The Margin Density algorithm for concept drift detection was explained in detail in Section 6.1.3. In order to visualize high dimensional data samples and models, all visualizations are produced after the data space has been projected to a two-dimensional space using Principal Component Analysis. All discussions onward assume that the transformation has been performed prior to visualization.

To visualize the SVM model, the equation for the classification boundary and the two support vectors $f_1(x)$ and $f_2(x)$ are first obtained from the training of an SVM learning model. Contour of classification boundary is produced by finding points that satisfy equation 8.2:

$$w \bullet x_i + b = 0 \qquad (8.2)$$

The x_i denotes the vector for the data points, $w \bullet x_i + b$ denotes the linear equation of the classification boundary in kernel space. Similarly, the support vectors' contours can be obtained by equation 8.3

$$w \bullet x_i + b = 1 \text{ and } w \bullet x_i + b = -1 \qquad (8.3)$$

Since the data points for the classification boundary and support vectors are in high dimension, these pointes are projected into 2D space using the same PCA vectors as the data samples.

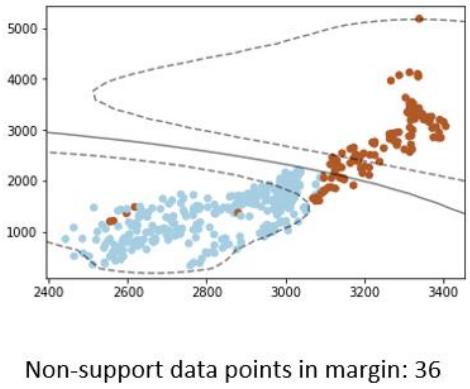


Figure 8.1. Illustration of visualization of support, class boundary and number of data samples within the SVM margin.

An example visualization produced is presented in Figure 8.1. The two axes show the data value range. The blue and brown dots show the two classes of samples. The two

dotted curve shows the support vectors, while the solid curve shows the class boundary. Notice that the support and class boundary are both non-linear, this is due to the RBF kernel. The number of data points within the margin is shown on the bottom of the visualization graph. Ideally, the SVM classification boundary should separate all blue samples to one side and brown samples to the other side. When this is not the case, it shows which samples the learning model predicts the wrong labels. The number of samples within the SVM margin is shown at the bottom of the graph. The visualization therefore presents the shape of the SVM model, how well it classifies the current chunk of data and how much margin density is in one visualization.

8.2.2 Visualizing Changes of a Model

Visualization such as the one shown in Figure 8.1 are produced chunk by chunk, regardless of the existence of concept drift. When no concept drift is detected, the visualization helps to explain how the current model is still suitable for the current data distribution. In an ideal situation, the visualization between two different chunks without concept drift should show that the margin density does not increase or decrease significantly and that the classification boundary of the SVM can separate the two classes well. However, it is possible for visualization to show the existence of the concept drift, but the concept drift detection in the explained framework failed to detect it. In this case, the proposed framework can help explain why a new model should have been trained.

In the case of concept drift detection, the explained framework typically trains a new model using the chunk of data when a drift occurs. Due to this new model being trained, our proposed framework will produce two visualizations: one with the outdated

model and one with the new model. An example is demonstrated in Figure 8.2. The visualization on the left shows the current model does not classify well. There are many samples within the margin, and many samples are misclassified. The left visualization shows that the outdated model is no longer suited for the current data distribution. The visualization on the right shows the new updated model, trained after concept drift. There is clearly improvement in both the margin density.

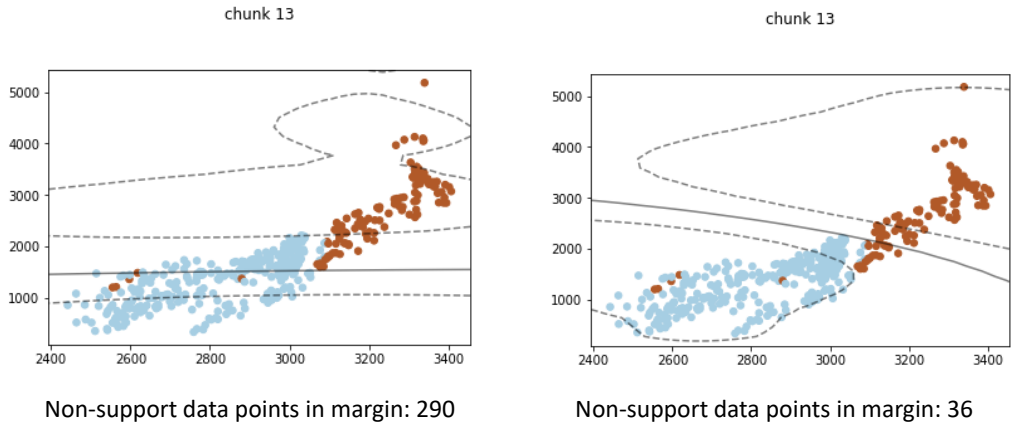


Figure 8.2. Illustration of visualization of models before and after concept drift

8.3. Explanation Evaluation*

The framework is applied to several real-world datasets listed in Table 8.1. The Forest Cover Type (Blackard & Dean, 1999) dataset (Covtype) predicts the forest cover type from cartographic variables. This dataset is selected because it has been shown to contain multiple types of concept drift (Zhukov et al, 2016). The original dataset has 54 features and 7 classes. To avoid the complications of predicting multiple class labels, only samples from 2 classes were used for model training, and explanation, making it a binary prediction problem. Electric Market (EM) dataset is a real-world dataset that keeps track of the rise and fall of electricity price (Zliobaite, 2013). Some features, such as date

and time, of the original dataset are useless for concept drift detection. These features are removed to avoid their interference to the detection result. After the selection of the features, out EM dataset contains 6 features.

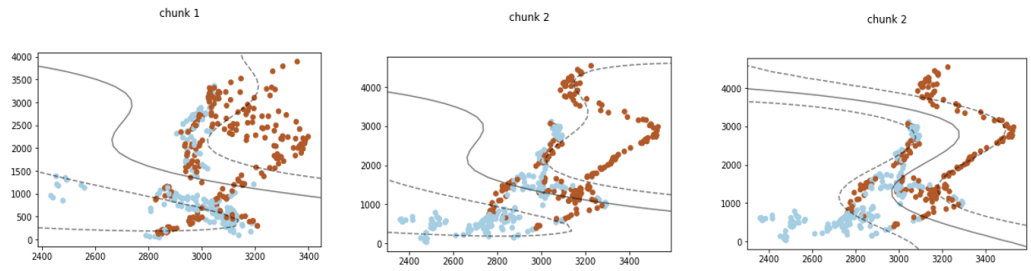
Using the above dataset, the framework produced multiple explanations. These explanations are then compiled into a survey so that users can evaluate the effectiveness of the explanation.

Table 8.1. Overview of the datasets used in experiment

Data Set	# of Samples	# of Features
Forest Cover Type (Covtype)	250K	54
Electric Market (EM)	45K	6

8.3.1 Survey Preparation

A survey is prepared using visualizations produced by applying DSE to the above two data sets. For optimal detection of concept drift, Covtype dataset was split into 40 chunks while EM dataset was split into 25 chunks. In survey 1 and 2, 11 and 9 visualizations from Covtype were randomly selected respectively. In survey 3 and 4, 10 visualizations from EM dataset were randomly selected respectively. For data mining experts, visualizations along might be enough, but for non-expert users, mere visualizations might not be enough to explain the situation. Therefore, written explanation made by machine learning researchers are provided on the bottom of the visualization. Thus, the visualization and the provided explanation constitutes a survey question. An example of the question is show in Figure 8.3. Since the explanation is provided, the survey does not measure how well users can formulate their own explanation based on the visualization. Instead, the effectiveness of the visualization is assessed on how much the average user agree with the written explanations. If non-expert



Explanation: The new model in new chunk reduced the number of samples in the margin, meaning the decision of the model is more precise. It also appear to better separate the two classes.

Figure 8.3. An example question from the explanation effectiveness survey showing visualizations of chunks with concept drift

users rated similarly as expert users in the survey, it shows that the visualizations can reduce the gap of understanding between non-expert and expert users. For each of the survey question, users assign a rating of 1 to 5 on how much they think they agree with the explanation provided within the survey. The meaning behind the rating is listed in Table 8.2, which is also explained to the surveyed users.

Table 8.2. Meaning behind survey scoring

Score	Meaning
1	The explanation is completely wrong. The visualizations do not reflect the conclusion provided by the explanation.
2	The explanation is somewhat wrong. The visualizations do not support the explanations but there are some elements in the visualization that shows why the explanation is produced
3	The explanation has some merit but also has some problems according to the visualizations.
4	The explanation is somewhat correct. The visualizations support the explanation, but there are some problems in the visualizations that may need to be considered.
5	The explanation is completely correct. The visualizations support the conclusion given by the explanation.

The survey is provided to two group of users. Group one consists of seven peer data mining researchers who is experts on data stream mining. Group two consists of ten users who have little to no background knowledge to data mining. The first group's

feedback will reflect how accurate the provided written explanation is and how well the visualization is presented. The second group's feedback will be compared to the first group of users to see whether the ratings are statistically different. The average score and the standard deviation of the survey results will be used to construct 90% confidence interval using Student's t-distribution. To show users agree with the explanation, the confidence interval should be greater than 3. A total of four surveys were conducted.

8.3.2 Survey Result

Group one's average scores, standard deviations and 90% confidence interval are given in Table 8.3. The confidence interval of user group one most fell between the rating 3 and 4 range. The highest interval of agreement score was achieved at Survey 4, between 3.212 and 4.388. The lowest interval was in survey 3, between 2.886 to 3.914 Overall, based on score meaning in Table 8.2, the result shows that the group one users agree with the explanations but think there are problems within the visualization that need to be addressed.

Table 8.3. Average score, standard deviation, and confidence interval of group one user ratings

	Average score	Standard Deviation	Confidence Interval
Survey 1	3.6	0.6	[3.159, 4.041]
Survey 2	3.3	0.3	[3.080, 3.520]
Survey 3	3.4	0.7	[2.886, 3.914]
Survey 4	3.8	0.8	[3.212, 4.388]

Further interview with the users from group one revealed several factors that limit the effectiveness of the visualization according. First, it is difficult to interpret when the SVM model is too complex in the 2D projected space. Although in higher kernel dimension SVM models are always linear, after projecting to lower dimension they can take on various shapes. A complex looking model reduces clearness in the explanation.

For example, in Figure 8.4, the SVM model presented is very complex in the 2D space. Since the two classes of data were also not separated well, it is almost impossible to explain if the current model is indeed suitable for the current data distribution.

Second, the shape difference between new model and old model highly impacts perceived correctness on the explanation. The explanations produced within the DSE relies heavily on differences that can be identified visually to show changes in the model. When the differences cannot be visualized clearly, the explanation of changes in model can be unconvincing. An example is shown in Figure 8.5, which shows both the old and new models due to concept drift. In this case, the new model improved the margin density, reducing the number of samples within the margin from 132 to 107, an 18.9% drop. In the visualization however, the new model does not visually change in shape, and thus during the survey this visualization was not able to convince users that the new model has an improved performance. Upon closer inspection, the majority of the gain in reducing margin density occurred within the circled area in Figure 8.5. The new support vector is able to exclude many blue class samples from the margin area. Such details are

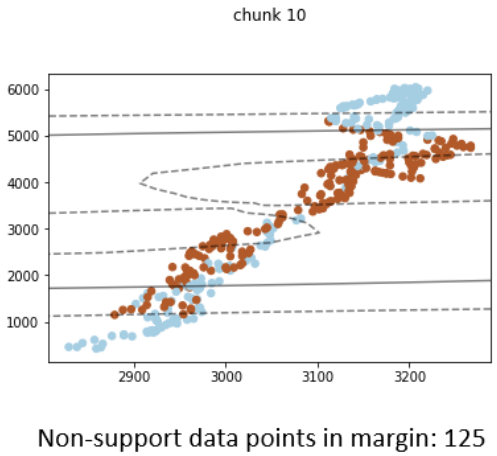


Figure 8.4. An overly complex SVM model, due to projecting high dimensional kernel space model to 2D space

sometimes overshadowed by the overall similarities between the new model and the old model.

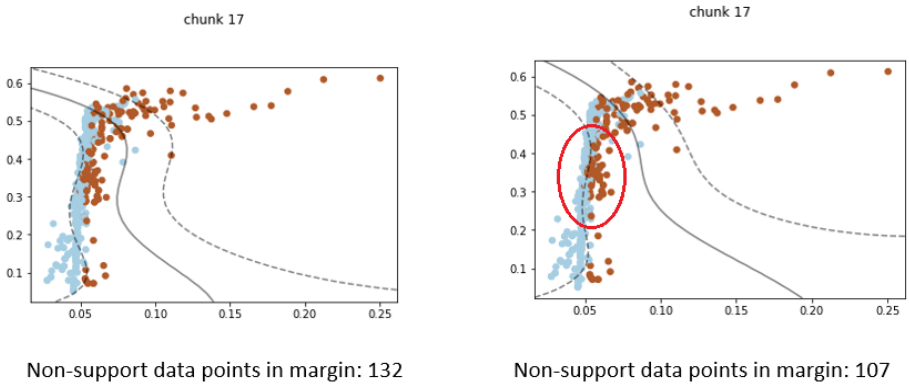


Figure 8.5. Explanation is less convincing when the new model does not change significantly in shape compared to the old model. Circled area shows where the margin density

Third, changes in data distribution shape affect perceived corrected of the explanation, regardless of if the change impacts model performance. While concept drift can be defined as data distribution change, not all distribution changes result in concept drift. Distribution changes with no impact on learning model performance do not need training a new model to replace the existing model. However, since the plotting of data samples makes most of the elements within the visualizations, changes in data distribution has a large impact on the visual representation of concept drift, or lack

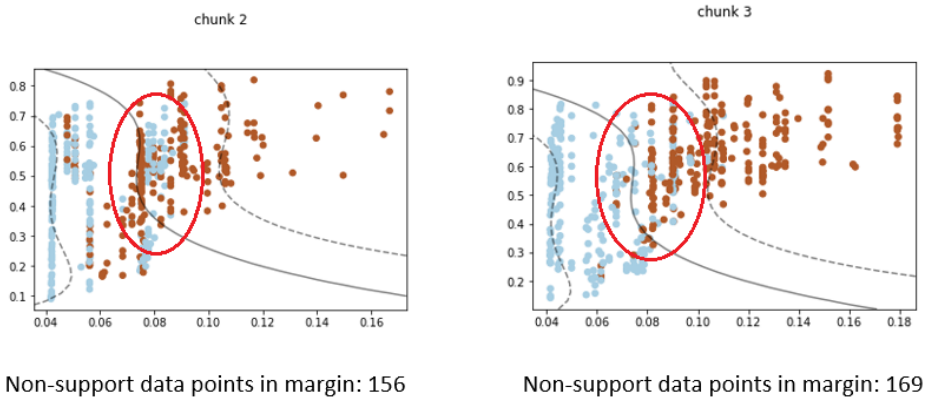


Figure 8.6. Changes in data distribution might not be concept drift, but it impacts perceived correctness in the explanation. Circled area shows the change in data distribution but does not result in worsening of model performance.

thereof. An example is provided in Figure 8.6, which shows two chunks of data with no concept drift. The learning model does not perform significantly worse between the two chunks of data. The visualization, however, shows significant changes of data distribution both within and outside of the margin. The circled area shows why the distribution change did not result in worse performance. In chunk 2, the majority of data samples are overlapped in a small area, which makes it difficult to visually estimate accurately how many of sample there were. In chunk 3, those samples are more spread out. This change does not affect the model performance, but visually it looks as though some significant change has occurred within the data.

Group two user rating averages are shown in Table 8.4. The group two users give lower average ratings with higher standard deviations compared with group one user ratings. While group one's average ranged between 3.3 to 3.8, group two's average ranged 3.0 to 3.2. This result is to be expected. Without data mining background, it is more likely that the group two users have less understanding of the meaning behind the visualization and the written explanation. The confidence intervals show that the ratings 3 more or less fall in the middle of most of the intervals, which means that the user believe there are both merits and problems to the visualizations and written explanations.

Table 8.4. Average score, standard deviation, and confidence interval of group two user ratings

	Average score	Standard Deviation	Confidence Interval
Survey 1	3.1	1.1	[2.462, 3.738]
Survey 2	3.2	1.0	[2.620, 3.780]
Survey 3	3.0	1.1	[2.360, 3.640]
Survey 4	3.1	1.0	[2.360, 3.640]

Figure 8.7 shows the comparison of confidence interval between expert (blue) and non-expert user (orange). Based on the Student's T distribution, survey 2 and 3 show expert and non-expert users overlapping in most of the confidence intervals. Survey 1 and 4 have less overlapping, which means that there are more differences between the expert and non-expert opinion. Overall, all confidence intervals between the two group overlaps, which means that group two users do not have statistically significant difference in ratings compared to group one at the 90% confidence level under Student's T distribution.

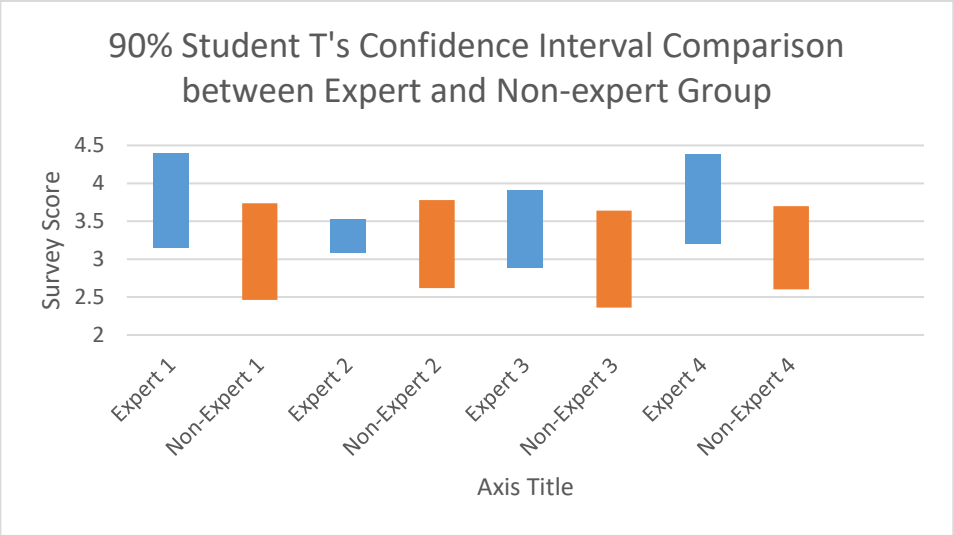


Figure 8.7. 90% Student T's Confidence Interval Comparison between Expert and Non-expert Group

Table 8.5. Common improvement needed for DSE framework according to non-expert user feedback when the user rated less than 3 on a survey question

	Improvement Needed
1	The class boundary in the visualizations did not clearly separate the two classes
2	Visualization does not show clear change in data or change in model
3	Having more visual cue as to what component of the visualization means what
4	The margins in some visualizations are confusing. It is difficult to tell where is inside or outside of the margin.

A follow up questionnaire is provided for the group two users to identify what can be improved in the visualization. 4 out of the 10 group two users provided feedback. The questionnaire asks them what can be improved on the questions where they rated low and what is good on questions that they rated high. The common elements in their feed backs

Table 8.6. Common merits for DSE framework according to non-expert user feedback when the user rated more than 3 on a survey question

	Improvement Needed
1	The visualizations show clear change and improvement between chunks of data
2	The model shown in the visualization is simple and easy to understand
3	Multiple graphs were provided for comparison

are compiled and sorted from the most common problem to the lest. The summary of improvement is shown in Table 8.5 and the summary of merit is shown in Table 8.6. The questionnaire also asks the user's opinion where the user rated 3. 3 out of 4 responses said the rating is because the graphs did not clearly show the explanation is correct or wrong. 1 out of 4 responses said the explanation is not understandable. The questionnaire reveal that the non-expert users face the same difficulty as expert users when using the DSE framework. In both groups, the need to clearer representation of change and model performance is needed for understanding the explanation.

8.4. Conclusion

Current machine learning explanation framework aims to explain and interpret the decision-making process of complex machine learning algorithms. In the case of data stream mining, explaining individual learning model is not sufficient because of the existence of concept drift. Typical stream mining framework trains new model when concept drift is detected because the existing model no longer can predict the current data

distribution. It is therefore necessary to explain the difference between the new model and old model, to justify the change in the decision process. This paper proposed DSE, a visualization framework that shows how new model is better suited for the new data distribution when concept drifts occur, and also shows how current model is still sufficient when concept drift does not occur. The visualization uses the margin density method to compare classification confidence of the SVM learning model through the data stream. To measure the effectiveness of DSE, a survey was conducted between a group of data mining experts and non-expert users. The survey result showed that DSE is able to reduce the gap of understand concept drift between experts and non-expert users. The survey also identified problems of the visualization and points to the following improvement:

- Better representation of high dimension data and learning model.
- Increase visual cues to highlight changes in model before and after concept drift.
- Increase visual cues to show data distribution change that does not affect model performance.

Future works include making visualization more generalized for different learning models other than SVM, novel representation of high dimensional models in 2D or 3D space, and expand the explanation to ensemble frameworks, where models were not replaced but added to the ensemble.

* Human subject study is approved by IRB on 5/4/2022. IRB number: 22.0369.

Reference number: 744438

CHAPTER 9. CONCLUSION

This proposal introduces the differences between data stream classification and traditional data mining tasks in three major area: fixed size vs unknown size of data, static vs dynamic data and one-time processing vs repeated processing of data. These differences bring unique challenges to data stream classification frameworks. The frameworks need to be able to adapt to changes in data stream, called concept drift, and at the same time balance performance and costs associated with model training. The proposal presented four frameworks that tries to maximize overall classification accuracy and minimizing labeling cost in the preprocessing, change detection and model training steps of stream classification.

Chapter 3 tries to reduce labeling cost of extremely imbalanced data stream by presenting several alternative approaches to random selection for sampling and labeling. The goal was to reduce the total number of samples needed for labeling because labeling is time consuming and expensive. The approaches utilized a grid density algorithm to search for minority class clusters in order to retrieve these instances with fewer overall samples than the default random selection approach. A synthetic data set and two real world data sets, Yeast and Satimag from UCI machine learning repository, were used for experimentation. The results from the synthetic data set showed that the efficiency of the approaches varies when different grid sizes were applied. Specifically, the results for our approach were similar to the default random selection approach when either the grid was

too large or too small. In one of the extreme cases where each grid contains 1 sample, the results had no significant difference from the default approach. It is also clear at the other extreme where one grid contains all samples then our approach degrades into random selection. However, when the grid size was between the two extremes, the results from our approach showed improvement over the default random selection approach. The real-world data results confirmed that observation. At optimal grid configuration, our proposed approach Grid Search had an average improvement of 19.4% for Yeast and 5.3% for Satimag. The experimentation also showed that dimension reduction is useful for reducing the number of grids in high dimensional data space and thus increases sampling efficiency. The Projected Grid that reduced both data sets to 2 dimensions had an average improvement of 18.2% in Yease and 27.4% in Satimag. On a higher dimension data set, results from Projected Grid improved significantly from the non-reduced Satimag results. Between 4 to 16 dimensions, we saw an average 28.9% improvement over random selection.

Chapter 4 explores the advantage of having a dynamic preprocessing strategy in data stream classification. The framework SPSD is proposed, which is a smart preprocessing approach that separates preprocessing from modeling in a stream mining framework. SPSD monitors the min-max range of each chunk of data and calculates two metrics to avoid unnecessary re-normalization in noisy data with outliers. Metric 1 is the percentage of samples that fall outside of previous min-max range. Metrics 2 is the percentage of difference between sample values and the referenced min-max value. If these two metrics reach above their respective threshold values, SPSD triggers a re-normalization. In our real-world experiment, we compared SPSD with two stream data

classification strategies, one that does not adapt to change ("no-change") and one that constantly retrains a new model at each chunk of data("all-change"). SPSD is shown to perform better than the "no-change" and in 50% of all chunks better than "all-change". The experiment demonstrated that 50% of retraining in all-change strategy can be avoided by simply preprocess the data using new parameters after concept drifts. We also compared SPSD with traditional streaming mining frameworks without SPSD: SEA, AWE, ACE and MAE. The comparison showed that among all four frameworks 34% to 48% of all chunks of data can benefit from SPSD so that no model retraining is need in those chunks. Our comparison showed that SPSD has the potential to reduce the cost associated with new model generation. SPSD demonstrates that for a streaming mining framework, one should not assume any component of the framework to be stationary. As demonstrated in our experiments, traditional frameworks can obtain better prediction results by not assuming that preprocessing step remains the same between model retraining. Because of the changing nature of non-stationary data streams, all components of a learning framework might benefit from an adaptive approach.

Chapter 5 presents the SRADL framework, which solves the delayed labeling problem where labeled samples needed for training new models are not immediately available after concept drift. In this chapter we described the delayed labeling problem in streaming data classification. SRADL contains three components: Concept Drift Detection, Semi-supervised Learning and Labeled Sample Reservoir. Concept Drift Detection monitors the data stream and signals Semi-supervised Learning component to update its learning model. Semi-supervised learning then requests labels to be made and trains a new semi-supervised model using available labels in the Labeled Sample Reservoir. The

reservoir is updated whenever latest samples are labeled. Our experiments involved two scenarios of the labeling process. The first scenario assumes that labels will arrive all together after a certain delay. The second scenario assumes that labels arrive continuously. We compared SRADL with three approaches: static (no training of new model occur), no-delay (all labeled samples are immediately available) and wait-and-train (wait for availability of labeled samples then train new model). In scenario 1, SRADL scored similarly compared to wait-and-train in some cases, and in some cases worse than wait-and-train. For scenario 2, however, SRADL performed much better both in synthetic and real-word data set experiments in most cases. The most improvement occurred when labeling delay time were long.

Chapter 6 demonstrated the complexity of real-world application requires an approach to cope with different types of concept drift to be detected in streams without labeling samples. Existing detection algorithms, which focus on only one type of concept drift, might not be able to detect drift in real-world streaming data where multiple types of concept drift occur. Framework HEFDD is proposed, which employs an ensemble of state-of-the-art concept drift detection algorithms with heuristic voting mechanism. Concept drift is detected as long as majority voting is reached for a specific type of concept drift. After each type of concept drifts is voted, the union of detection decisions for all types of drift is produced as the final detection decision. HEFDD was implemented using Margin Density, Clustering, Kolmogorov-Smirnov (KS) Test and Grid-Based drift detection algorithms as components of the ensemble. Stationary drifts are covered by Margin Density approach while non-stationary drifts are covered by Clustering, KS test and Grid-Based approach. HEFDD was tested with synthetic and real-word dataset,

including EM, Covtype and SPAM. For synthetic data set experiments, it is verified that different algorithms detect different types of drift, confirming our heuristic analysis. In real world experiment, HEFDD shows significant improvement at $p < 0.05$ using z-score test when compared to individual drift detection algorithm. In EM and Covtype HEFDD outperforms Margin Density by as much as 10% accuracy because it is able to detect more concept drifts in the stream.

Chapter 7 improved default SRADL framework by combining it with HEFDD to produce the SRADL-HEFDD framework. The HEFDD framework enabled SRADL to react to different kind of concept drift using different synthetic labeling strategies. To handle fixed space drift, SRADL-HEFDD uses grid within SVM margins to assign synthetic labels to unlabeled samples. To handle non-fixed space drift, SRADL-HEFDD assumes samples from the same class will form clusters in the data space and assign synthetic labels using KNN. SRADL-HEFDD was compared to SRADL. The results showed SRADL-HEFDD can utilize small number of labeled samples more effectively than the default SRADL.

Chapter 8 introduced Model Explainability and formalized the DSE framework. Existing model explainability frameworks focus on static model in static data mining. However, concept drifts in data streams require models to adapt to dynamic data environment. The changes between models before and after concept drifts were not explained using traditional model explainability framework. DSE is a visualization framework aims to explain changes in data stream mining models to data analyst experts and non-experts alike. DSE uses SVM and SVM related algorithms for model building and concept drift detection but can be extended to other algorithms. When concept drift

was detected. multiple visualizations of models before and after concept drift were created. They were compared to show how the new model fit the new data distribution after concept drift better. When concept drift was not detected, visualizations were also created to validate that this is not a false-negative detection. Since DSE aims to explain changes in models to non-experts, a survey was set up to measure DSE's effectiveness. Visualizations and explanations were presented to an expert user group and a non-expert user group, and how much each group agreed with the explanation were recorded. Although the average agreement score in the non-expert group is lower than the expert group, the 90% confidence interval of t-distribution overlaps between the two group. This means that the agreement scores were not statistically significantly different between the two groups. The survey results showed that DSE is able to reduce the gap between expert and non-expert users in understanding changes in data stream mining.

For future research, there are three main areas where this dissertation can be continued. First, semi-supervised learning with limited label availability can be further improved from SRADL and SRADL-HEFDD. More realistic assumptions of data distribution in data streams can be made, so that the process of synthetic labeling can be improved. Second, more types of concept drifts should be explored for HEFDD. Also, algorithms in reducing false negative and false positive concept drift detection should be proposed. Third, better visualization algorithms for DSE should be explored. The new algorithm should work better than Principal Component Analysis in visualizing high dimensional data and model. More learning models other than SVM should be integrated with DSE framework.

REFERENCES

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, *6*, 52138-52160.

Agrahari, S., & Singh, A. K. (2021). Concept Drift Detection in Data Stream Mining: A literature review. *Journal of King Saud University-Computer and Information Sciences*.

Ahmadi, Z., & Beigy, H. (2012, March). Semi-supervised ensemble learning of data streams in the presence of concept drift. In *International Conference on Hybrid Artificial Intelligence Systems* (pp. 526-537). Springer, Berlin, Heidelberg.

Ahmadi, Z., & Kramer, S. (2018). Modeling recurring concepts in data streams: a graph-based framework. *Knowledge and Information Systems*, 55(1), 15-44.

Al-Khateeb, T., Masud, M. M., Al-Naami, K. M., Seker, S. E., Mustafa, A. M., Khan, L., Trabelsi, Z., Aggarwal, C. & Han, J. (2016). Recurring and novel class detection using class-based ensemble for evolving data stream. *IEEE Transactions on Knowledge and Data Engineering*, 28(10), 2752-2764.

Anderson, R., Koh, Y. S., & Dobbie, G. (2016, December). CPF: Concept Profiling Framework for recurring drifts in data streams. In *Australasian Joint Conference on Artificial Intelligence* (pp. 203-214). Springer, Cham.

Ángel, A. M., Bártolo, G. J., & Ernestina, M. (2016). Predicting recurring concepts on data-streams by means of a meta-model and a fuzzy similarity function. *Expert Systems with Applications*, 46, 87-105.

Astudillo, C. A., González, J. I., Oommen, B. J., & Yazidi, A. (2016, December). Concept Drift Detection Using Online Histogram-Based Bayesian Classifiers. In *Australasian Joint Conference on Artificial Intelligence* (pp. 175-182). Springer, Cham.

Babu, D. K., Ramadevi, Y., & Ramana, K. V. (2017). PGNBC: Pearson Gaussian Naïve Bayes classifier for data stream classification with recurring concept drift. *Intelligent Data Analysis*, *21*(5), 1173-1191.

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., Morales-Bueno, R. (2006). Early drift detection method. In *Proc. ECML/PKDD 2006, Work. Knowledge Discovery from Data Streams*, pp. 77–86.

Bennett, K. P., & Demiriz, A. (1999). Semi-supervised support vector machines. In *Advances in Neural Information processing systems* (pp. 368-374).

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009, June). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 139-148). ACM.

Blackard, J. A., & Dean, D. J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, *24*(3), 131-151.

Bock, R.K., Chilingarian, A., Gaug, M., Hakl, F., Hengstebeck, T., Jirina, M., Klaschka, J., Kotrc, E., Savicky, P., Towers, S., Vaicilius, A., Wittek W. (2004). Methods for multidimensional event classification: a case study using images from a Cherenkov gamma-ray telescope. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 516, pp. 511-528.

Brzeziński, D., & Stefanowski, J. (2011, May). Accuracy updated ensemble for data streams with concept drift. In *International conference on hybrid artificial intelligence systems* (pp. 155-163). Springer, Berlin, Heidelberg.

Brzezinski, D., & Stefanowski, J. (2013). Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 81-94.

Burnap, P., & Williams, M. L. (2016). Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science*, *5*(1), 11.

Cabanes, G., & Bennani, Y. (2012, June). Change detection in data streams through unsupervised learning. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, (pp. 1-6). IEEE.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015, August). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1721-1730).

Cattral , R.& Oppacher, F. (2017). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/Poker+Hand]. Irvine, CA: University of California, School of Information and Computer Science.

Chen, Y., & Tu, L. (2007, August). Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 133-142). ACM.

Crone, S. F., Lessmann, S., & Stahlbock, R. (2006). The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research*, *173*(3), 781-800.

da Costa, F. G., Duarte, F. S., Vallim, R. M., & de Mello, R. F. (2017). Multidimensional surrogate stability to detect data stream concept drift. *Expert Systems with Applications*, 87, 15-29.

de Francisci Morales, G., Bifet, A., Khan, L., Gama, J., & Fan, W. (2016, August). Iot big data stream mining. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 2119-2120).

de Lima Cabral, D. R., & de Barros, R. S. M. (2018). Concept drift detection based on Fisher's Exact test. *Information Sciences*, 442, 220-234.

Dehghan, M., Beigy, H., & ZareMoodi, P. (2016). A novel concept drift detection method in data streams using ensemble classifiers. *Intelligent Data Analysis*, 20(6), 1329-1350.

Demšar, J., & Bosnić, Z. (2018). Detecting concept drift in data streams using model explanation. *Expert Systems with Applications*, 92, 546-559.

Ditzler, G., & Polikar, R. (2011, July). Semi-supervised learning in nonstationary environments. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, (pp. 2741-2748). IEEE.

Ditzler, G., & Polikar, R. (2012). Incremental learning of concept drift from streaming imbalanced data. *IEEE transactions on knowledge and data engineering*, *25*(10), 2283-2301.

dos Reis, D. M., Flach, P., Matwin, S., & Batista, G. (2016, August). Fast unsupervised online drift detection using incremental Kolmogorov-Smirnov test. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1545-1554). ACM.

Dries, A., & Rückert, U. (2009). Adaptive concept drift detection. Statistical Analysis and Data Mining: *The ASA Data Science Journal*, 2(5-6), 311-327.

Du, L., Song, Q., & Jia, X. (2014). Detecting concept drift: An information entropy based method using an adaptive sliding window. *Intelligent Data Analysis*, 18(3), 337-364.

Duda, P., Jaworski, M., & Rutkowski, L. (2017, November). On ensemble components selection in data streams scenario with reoccurring concept-drift. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on* (pp. 1-7). IEEE.

Faithfull, W. J., Rodríguez, J. J., & Kuncheva, L. I. (2019). Combining univariate approaches for ensemble change detection in multivariate data. *Information Fusion*, 45, 202-214.

Fan, W. (2004, August). Systematic data selection to mine concept-drifting data streams. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 128-137). ACM.

Fan, W. (2004, August). Streamminer: A classifier ensemble-based engine to mine concept-drifting data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30* (pp. 1257-1260). VLDB Endowment.

Farid, D. M., Zhang, L., Hossain, A., Rahman, C. M., Strachan, R., Sexton, G., & Dahal, K. (2013). An adaptive ensemble classifier for mining concept drifting data streams. *Expert Systems with Applications*, *40*(15), 5895-5906.

Fisher, A., Rudin, C., & Dominici, F. (2018). All models are wrong but many are useful: Variable importance for black-box, proprietary, or misspecified prediction models, using model class reliance. *arXiv preprint arXiv:1801.01489*.

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

Friedman, J. H., & Rafsky, L. C. (1979). Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. *The Annals of Statistics*, 697-717.

Fritsch, A., & Ickstadt, K. (2009). Improved criteria for clustering based on the posterior similarity matrix. Bayesian analysis, 4(2), 367-391.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004, September). Learning with drift detection. In *Brazilian symposium on artificial intelligence* (pp. 286-295). Springer, Berlin, Heidelberg.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4), 44.

Gao, J., Fan, W., & Han, J. (2007, October). On appropriate assumptions to mine data streams: Analysis and practice. In *Seventh IEEE International Conference on Data Mining, 2007. ICDM 2007*. (pp. 143-152). IEEE.

Gemaque, R. N., Costa, A. F. J., Giusti, R., & Dos Santos, E. M. (2020). An overview of unsupervised drift detection methods. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(6), e1381.

Ghosal, S., Blystone, D., Singh, A. K., Ganapathysubramanian, B., Singh, A., & Sarkar, S. (2018). An explainable deep machine vision framework for plant stress phenotyping. *Proceedings of the National Academy of Sciences*, *115*(18), 4613-4618.

Glazer, A., Lindenbaum, M., & Markovitch, S. (2012). Learning high-density regions for a generalized kolmogorov-smirnov test in high-dimensional data. In *Advances in Neural Information Processing Systems* (pp. 728-736).

GonçAlves Jr, P. M., & De Barros, R. S. M. (2013). RCD: A recurring concept drift framework. *Pattern Recognition Letters*, 34(9), 1018-1025.

Haque, A., Khan, L., & Baron, M. (2016, February). SAND: Semi-Supervised Adaptive Novel Class Detection and Classification over Data Stream. In *Association for the Advancement of Artificial Intelligence* (pp. 1652-1658).

Harel, M., Mannor, S., El-Yaniv, R., & Crammer, K. (2014, June). Concept drift detection through resampling. In International conference on machine learning (pp. 1009-1017). PMLR.

Harries, M., & Wales, N. S. (1999). Splice-2 comparative evaluation: Electricity pricing.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100-108.

Hidalgo, J. I., Santos, S. G., & Barros, R. S. (2021). Dynamically adjusting diversity in ensembles for the classification of data streams with concept drift. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *16*(2), 1-20.

Ho, Y. C., & Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, *115*(3), 549-570.

Hoens, T. R., Polikar, R., & Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, *1*(1), 89-101.

Holzinger, A. (2018, August). From machine learning to explainable AI. In *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)* (pp. 55-66). IEEE.

Hopkins, M., Reeber, E., Forman, G., and Suermondt, J. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/spambase]. Irvine, CA: University of California, School of Information and Computer Science.

Hosseini, M. J., Gholipour, A., & Beigy, H. (2016). An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams. *Knowledge and Information Systems*, 46(3), 567-597.

Howard, A., Zhang, C., & Horvitz, E. (2017, March). Addressing bias in machine learning algorithms: A pilot study on emotion recognition for intelligent systems. In *2017 IEEE Workshop on Advanced Robotics and Its Social Impacts (ARSO)* (pp. 1-7). IEEE.

Hu, H., & Kantardzic, M. (2016, January). Smart preprocessing improves data stream mining. In 2016 49th Hawaii International Conference on System Sciences (HICSS) (pp. 1749-1757). IEEE.

Hu, H., & Kantardzic, M. (2017, January). Sliding reservoir approach for delayed labeling in streaming data classification. In Proceedings of the 50th Hawaii International Conference on System Sciences.

Hu, H., & Kantardzic, (2022) M. Heuristic ensemble for unsupervised detection of multiple types of concept drift in data stream classification. Intelligent Decision Technologies, (Preprint), 1-14.

Hu, H., Kantardzic, M., & Lyu, L. (2018, December). Detecting Different Types of Concept Drifts with Ensemble Framework. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 344-350). IEEE.

Hu, H., Kantardzic, M. M., & Sethi, T. S. (2013, October). Selecting samples for labeling in unbalanced streaming data environments. In 2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT) (pp. 1-7). IEEE.

Hu, H., Kantardzic, M., & Sethi, T. S. (2020). No Free Lunch Theorem for concept drift detection in streaming data classification: A review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(2), e1327.

Hu, R., Andreas, J., Darrell, T., & Saenko, K. (2018). Explainable neural computation via stack neural module networks. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 53-69).

Ikonomovska, E. (2011) Airline Dataset. http://kt.ijs.si/elena_ikonomovska/data.html

Jaworski, M., Duda, P., & Rutkowski, L. (2017, November). On applying the Restricted Boltzmann Machine to active concept drift detection. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI),* (pp. 1-8). IEEE.

Jiang, Y., Zhao, Q., & Lu, Y. (2014, August). Ensemble based data stream mining with recalling and forgetting mechanisms. In *2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)* (pp. 430-435). IEEE.

John, G. H., & Langley, P. (1995, August). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence* (pp. 338-345). Morgan Kaufmann Publishers Inc.

Kang, F., Jin, R., & Sukthankar, R. (2006, June). Correlated label propagation with application to multi-label learning. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) (Vol. 2, pp. 1719-1726). IEEE.

Kantardzic, M., Ryu, J. W., & Walgampaya, C. (2010, June). Building a new classifier in an ensemble using streaming unlabeled data. *In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* (pp. 77-86). Springer, Berlin, Heidelberg.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2015). Self-adaptive windowing approach for handling complex concept drift. *Cognitive Computation*, *7*(6), 772-790.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2019). A New Combination of Diversity Techniques in Ensemble Classifiers for Handling Complex Concept Drift. In *Learning from Data Streams in Evolving Environments* (pp. 39-61). Springer, Cham.

Kifer, D., Ben-David, S., & Gehrke, J. (2004, August). Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30* (pp. 180-191). VLDB Endowment.

Kim, H., Madhvanath, S., & Sun, T. (2015, October). Hybrid active learning for non-stationary streaming data with asynchronous labeling. In *2015 IEEE International Conference on Big Data (Big Data)* (pp. 287-292). IEEE.

Kim, Y., & Park, C. H. (2017). An efficient concept drift detection method for streaming data under limited labeling. *IEICE Transactions on Information and systems*, *100*(10), 2537-2546.

Klauschen, F., Müller, K. R., Binder, A., Bockmayr, M., Hägele, M., Seegerer, P., ... & Michiels, S. (2018, October). Scoring of tumor-infiltrating lymphocytes: From visual estimation to machine learning. In *Seminars in cancer biology* (Vol. 52, pp. 151-157). Academic Press.

Kolter, J. Z., & Maloof, M. A. (2003, November). Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Third IEEE international conference on data mining* (pp. 123-130). IEEE.

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (2004). Multimedia mining. *WSEAS Transactions on Systems*, *3*(10), 3263-3268.

Krawczyk, B., & Cano, A. (2018). Online ensemble learning with abstaining classifiers for drifting and noisy data streams. *Applied Soft Computing*, 68, 677-692.

Krawczyk, B., & Woźniak, M. (2015). One-class classifiers with incremental learning and forgetting for data streams with concept drift. Soft Computing, 19(12), 3387-3400.

Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., ... & Stefanowski, J. (2014). Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, *16*(1), 1-10.

Kuo, Y. T., Lonie, A., Pearce, A. R., & Sonenberg, L. (2014). Mining surprising patterns and their explanations in clinical data. *Applied Artificial Intelligence*, *28*(2), 111-138.

Kuwajima, H., Tanaka, M., & Okutomi, M. (2019). Improving transparency of deep neural inference process. *Progress in Artificial Intelligence*, *8*(2), 273-285.

Lazarescu, M. M., Venkatesh, S., & Bui, H. H. (2004). Using multiple windows to track concept drift. *Intelligent data analysis*, 8(1), 29-59.

Lee, H., Yune, S., Mansouri, M., Kim, M., Tajmir, S. H., Guerrier, C. E., ... & Gonzalez, R. G. (2019). An explainable deep-learning algorithm for the detection of acute intracranial haemorrhage from small datasets. *Nature Biomedical Engineering*, *3*(3), 173.

Lee, J., & Magoules, F. (2012, June). Detection of concept drift for learning from stream data. In *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)*, (pp. 241-245). IEEE.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., & Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, *113*(523), 1094-1111.

Lee, K., Palsetia, D., Narayanan, R., Patwary, M. M. A., Agrawal, A., & Choudhary, A. (2011, December). Twitter trending topic classification. In 2011 IEEE 11th International Conference on Data Mining Workshops (pp. 251-258). IEEE.

Letham, B., Rudin, C., McCormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, *9*(3), 1350-1371.

Li, C., Zhang, Y., & Li, X. (2009, June). OcVFDT: one-class very fast decision tree for one-class classification of data streams. In *Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data* (pp. 79-86). ACM.

Li, P., Wu, M., He, J., & Hu, X. (2021). Recurring Drift Detection and Model Selection-Based Ensemble Classification for Data Streams with Unlabeled Data. *New Generation Computing*, *39*(2), 341-376..

Li, P., Wu, X., & Hu, X. (2010, October). Mining recurring concept drifts with limited labeled streaming data. In *Proceedings of 2nd Asian Conference on Machine Learning* (pp. 241-252).

Li, P., Wu, X., Hu, X., Liang, Q., & Gao, Y. (2010). A random decision tree ensemble for mining concept drifts from noisy data streams. *Applied Artificial Intelligence*, *24*(7), 680-710.

Li, Y. F., & Liang, D. M. (2019). Safe semi-supervised learning: a brief introduction. Frontiers of Computer Science, 13(4), 669-676.

Li, Y. F., Kwok, J. T., & Zhou, Z. H. (2009, June). Semi-supervised learning using label mean. In *Proceedings of the 26th Annual International Conference on Machine Learning*(pp. 633-640). ACM.

Linardatos, P., Papastefanopoulos, V., & Kotsiantis, S. (2020). Explainable ai: A review of machine learning interpretability methods. *Entropy*, *23*(1), 18.

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

Lobo, J. L., Del Ser, J., Laña, I., Bilbao, M. N., & Kasabov, N. (2018, October). Drift Detection over Non-stationary Data Streams Using Evolving Spiking Neural Networks. In *International Symposium on Intelligent and Distributed Computing* Springer, Cham, (pp. 82-94).

Loyola-Gonzalez, O. (2019). Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE Access*, *7*, 154096-154113.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346-2363.

Lughofer, E., Weigl, E., Heidl, W., Eitzinger, C., & Radauer, T. (2015). Integrating new classes on the fly in evolving fuzzy classifier designs and their application in visual inspection. *Applied Soft Computing*, *35*, 558-582.

Lughofer, E., Weigl, E., Heidl, W., Eitzinger, C., & Radauer, T. (2016). Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances. *Information Sciences*, *355*, 127-151.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 4765-4774.

Lundberg, S. M., Nair, B., Vavilala, M. S., Horibe, M., Eisses, M. J., Adams, T., ... & Lee, S. I. (2018). Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature biomedical engineering*, *2*(10), 749.

Mahdi, O. A., Pardede, E., & Cao, J. (2018, January). Combination of information entropy and ensemble classification for detecting concept drift in data stream. In *Proceedings of the Australasian Computer Science Week Multiconference* (p. 13). ACM.

Masud, M. M., Chen, Q., Khan, L., Aggarwal, C., Gao, J., Han, J., & Thuraisingham, B. (2010, December). Addressing concept-evolution in concept-drifting data streams. *In 2010 IEEE 10th International Conference on Data Mining (ICDM)*, (pp. 929-934). IEEE.

Masud, M. M., Chen, Q., Khan, L., Aggarwal, C. C., Gao, J., Han, J., Srivastava, A. & Oza, N. C. (2013). Classification and adaptive novel class detection of feature-evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, 25(7), 1484-1497.

Masud, M., Gao, J., Khan, L., Han, J., & Thuraisingham, B. (2009, September). Integrating novel class detection with classification for concept-drifting data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 79-94). Springer, Berlin, Heidelberg.

Masud, M., Gao, J., Khan, L., Han, J., & Thuraisingham, B. M. (2010). Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, *23*(6), 859-874.

McInerney, J., Lacker, B., Hansen, S., Higley, K., Bouchard, H., Gruson, A., & Mehrotra, R. (2018, September). Explore, exploit, and explain: personalizing explainable recommendations with bandits. In *Proceedings of the 12th ACM Conference on Recommender Systems* (pp. 31-39). ACM.

Mesterharm, C. (2005, October). On-line learning with delayed label feedback. In *International Conference on Algorithmic Learning Theory* (pp. 399-413). Springer, Berlin, Heidelberg.

Minku, L. L., & Yao, X. (2012). DDD: A new ensemble approach for dealing with concept drift. *IEEE transactions on knowledge and data engineering*, 24(4), 619-633.

Minku, L. L., White, A. P., & Yao, X. (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5), 730-742.

Mohamad, S., Sayed-Mouchaweh, M., & Bouchachia, A. (2018). Active learning for classifying data streams with unknown number of classes. *Neural Networks*, 98, 1-15.

Mustafa, A. M., Ayoade, G., Al-Naami, K., Khan, L., Hamlen, K. W., Thuraisingham, B., & Araujo, F. (2017, December). Unsupervised deep embedding for novel class detection over data stream. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 1830-1839). IEEE.

Newman, D.J., Hettich, S., Blake, C.L., and Merz, C.J., UCI Reprository of ML databases [http://ics.uci.edu/~mlearn/MLRepository.html]. 1998, University of California, Department of Information and Computer Science: Irvine, CA.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., & Clune, J. (2016). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems* (pp. 3387-3395).

Nikzad-Langerodi, R., Lughofer, E., Cernuda, C., Reischer, T., Kantner, W., Pawliczek, M., & Brandstetter, M. (2018). Calibration model maintenance in melamine resin production: integrating drift detection, smart sample selection and model adaptation. *Analytica chimica acta*, *1013*, 1-12.

Nishida, K., Yamauchi, K., & Omori, T. (2005, June). ACE: Adaptive classifiers-ensemble system for concept-drifting environments. In *International Workshop on Multiple Classifier Systems* (pp. 176-185). Springer, Berlin, Heidelberg.

Nishida, K., & Yamauchi, K. (2007, October). Detecting concept drift using statistical testing. In *International conference on discovery science* (pp. 264-269). Springer, Berlin, Heidelberg.

Pesaranghader, A., & Viktor, H. L. (2016, September). Fast hoeffding drift detection method for evolving data streams. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 96-111). Springer, Cham.

Pietruczuk, L., Rutkowski, L., Jaworski, M., & Duda, P. (2016, July). A method for automatic adjustment of ensemble size in stream data mining. In *Neural Networks (IJCNN), 2016 International Joint Conference on* (pp. 9-15). IEEE.

Pinage, F. A., dos Santos, E. M., & da Gama, J. M. P. (2016). Classification systems in dynamic environments: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(5), 156-166. doi: 10.1002/widm.1184.

Polonik, W. (1999). Concentration and goodness-of-fit in higher dimensions:(Asymptotically) distribution-free methods. *The Annals of Statistics*, 27(4), 1210-1229.

Pratama, M., Lu, J., Lughofer, E., Zhang, G., & Anavatti, S. (2016). Scaffolding type-2 classifier for incremental learning under concept drifts. *Neurocomputing*, *191*, 304-329.

Pratama, M., Lu, J., Lughofer, E., Zhang, G., & Er, M. J. (2016). An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks. *IEEE Transactions on Fuzzy Systems*, *25*(5), 1175-1192.

Priya, S., & Uthra, R. A. (2021). Comprehensive analysis for class imbalance data with concept drift using ensemble based classification. *Journal of Ambient Intelligence and Humanized Computing*, *12*(5), 4943-4956.

Ramakrishna, B., & Rao, S. K. M. (2017). Concept Drift Detection in Data Stream Mining: The Review of Contemporary Literature. *Global Journal of Computer Science and Technology*.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., & Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. Neurocomputing, 239, 39-57.

Reddy, K. S., Reddy, M. K., & Sitaramulu, V. (2013, February). An effective data preprocessing method for Web Usage Mining. In *2013 International Conference on Information Communication and Embedded Systems (ICICES)* (pp. 7-10). IEEE.

Reddy, Y. C. A. P., Viswanath, P., & Reddy, B. E. (2018). Semi-supervised learning: A brief review. Int. J. Eng. Technol, 7(1.8), 81.

Ren, S., Liao, B., Zhu, W., & Li, K. (2018). Knowledge-maximized ensemble algorithm for different types of concept drift. *Information Sciences*, 430, 261-281.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135-1144). ACM.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018, April). Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Roscher, R., Bohn, B., Duarte, M. F., & Garcke, J. (2020). Explainable machine learning for scientific insights and discoveries. *Ieee Access*, *8*, 42200-42216.

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*, 33(2), 191-198.

Rutkowski, L., Jaworski, M., Pietruczuk, L., & Duda, P. (2014). A new method for data stream mining based on the misclassification error. *IEEE transactions on neural networks and learning systems*, 26(5), 1048-1059.

Ryu, J. W., Kantardzic, M. M., & Kim, M. W. (2012, August). Efficiently maintaining the performance of an ensemble classifier in streaming data. In *International Conference on Hybrid Information Technology* (pp. 533-540). Springer, Berlin, Heidelberg.

Saha, D., Haque, M. M., Sarkar, A., & Alam, F. (2018). Novel Class Detection in Concept Drifting Data Streams Using Decision Tree Leaves (Doctoral dissertation).

Sakthithasan, S., & Pears, R. (2016). Capturing recurring concepts using discrete Fourier transform. *Concurrency and computation: practice and experience*, *28*(15), 4013-4035.

Sethi, T. S., & Kantardzic, M. (2017). On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82, 77-99.

Sethi, T. S., Kantardzic, M., & Hu, H. (2016). A grid density based framework for classifying streaming data in the presence of concept drift. *Journal of Intelligent Information Systems*, 46(1), 179-211.

Sethi, T. S., Kantardzic, M., Arabmakki, E., & Hu, H. (2014, August). An ensemble classification approach for handling spatio-temporal drifts in partially labeled data streams. In *2014 IEEE 15th International Conference on Information Reuse and Integration (IRI)*, (pp. 725-732). IEEE.

Sethi, T. S., Kantardzic, M., Lyu, L., & Chen, J. (2018). A dynamic-adversarial mining approach to the security of machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(3), e1245.

Shaker, A., & Lughofer, E. (2014). Self-adaptive and local strategies for a smooth treatment of drifts in data streams. *Evolving Systems*, *5*(4), 239-257.

Siahroudi, S. K., Moodi, P. Z., & Beigy, H. (2018). Detection of evolving concepts in non-stationary data streams: A multiple kernel learning approach. *Expert Systems with Applications*, 91, 187-197.

Sidhu, P., & Bhatia, M. P. S. (2017). A two ensemble system to handle concept drifting data streams: recurring dynamic weighted majority. *International Journal of Machine Learning and Cybernetics*, 1-16.

Sidhu, P., & Bhatia, M. P. S. (2018). A novel online ensemble approach to handle concept drifting data streams: diversified dynamic weighted majority. *International Journal of Machine Learning and Cybernetics*, 9(1), 37-61.

Singh, A., Sengupta, S., & Lakshminarayanan, V. (2020). Explainable deep learning models in medical image analysis. Journal of Imaging, 6(6), 52.

Sobolewski, P., & Wozniak, M. (2013). Concept Drift Detection and Model Selection with Simulated Recurrence and Ensembles of Statistical Detectors. *Journal of Universal Computer Science*, 19(4), 462-483.

Song, X., Wu, M., Jermaine, C., & Ranka, S. (2007, August). Statistical change detection for multi-dimensional data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 667-676). ACM.

Spinner, T., Schlegel, U., Schäfer, H., & El-Assady, M. (2019). explAIner: A visual analytics framework for interactive and explainable machine learning. *IEEE transactions on visualization and computer graphics*, *26*(1), 1064-1074.

Spinosa, E. J., de Leon F de Carvalho, A. P., & Gama, J. (2007, March). Olindda: A cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM symposium on Applied computing* (pp. 448-452). ACM.

Street, W. N., & Kim, Y. (2001, August). A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 377-382). ACM.

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*, 28(6), 1532-1545.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, *106*(2), 58.

Tsymbal, A., Pechenizkiy, M., Cunningham, P., & Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. *Information fusion*, *9*(1), 56-68.

Tu, L., & Chen, Y. (2009). Stream data clustering based on grid density and attraction. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(3), 12.

Ustun, B., & Rudin, C. (2016). Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, *102*(3), 349-391.

Veale, M., Van Kleek, M., & Binns, R. (2018, April). Fairness and accountability design needs for algorithmic support in high-stakes public sector decision-making. In Proceedings of the 2018 chi conference on human factors in computing systems (pp. 1-14).

Vorburger, P., & Bernstein, A. (2006, December). Entropy-based concept shift detection. In *Sixth International Conference on Data Mining, ICDM'06*, (pp. 1113-1118). IEEE.

Wang, B., Tu, Z., & Tsotsos, J. K. (2013). Dynamic label propagation for semi-supervised multi-class multi-label classification. In Proceedings of the IEEE international conference on computer vision (pp. 425-432).

Wang, D., Wu, P., Zhao, P., Wu, Y., Miao, C., & Hoi, S. C. (2014, December). High-dimensional data stream classification via sparse online learning. In 2014 IEEE international conference on data mining (pp. 1007-1012). IEEE.

Wang, F., & Zhang, C. (2007). Label propagation through linear neighborhoods. IEEE Transactions on Knowledge and Data Engineering, 20(1), 55-67.

Wang, H., Fan, W., Yu, P. S., & Han, J. (2003, August). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 226-235). AcM.

Wang, P., Zhang, P., & Guo, L. (2012, April). Mining multi-label data streams using ensemble-based active learning. In *Proceedings of the 2012 SIAM international conference on data mining* (pp. 1131-1140). Society for Industrial and Applied Mathematics.

Wang, X., Kang, Q., Zhou, M., & Yao, S. (2018, August). A Multiscale Concept Drift Detection Method for Learning from Data Streams. In *2018 IEEE 14th International Conference on Automation Science and Engineering* (CASE) (pp. 786-790). IEEE.

Wang, Y., Xu, X., Zhao, H., & Hua, Z. (2010). Semi-supervised learning based on nearest neighbor rule and cut edges. *Knowledge-Based Systems*, *23*(6), 547-554.

Wares, S., Isaacs, J., & Elyan, E. (2019). Data stream mining: methods and challenges for handling concept drift. *SN Applied Sciences*, *1*(11), 1-19.

Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4), 964-994.

Webb, G. I., Lee, L. K., Goethals, B., & Petitjean, F. (2018). Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, *32*(5), 1179-1199.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, *23*(1), 69-101.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. & Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048-2057).

Yan, J., Zhang, B., Liu, N., Yan, S., Cheng, Q., Fan, W. & Chen, Z. (2006). Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing. *IEEE transactions on Knowledge and Data Engineering*, *18*(3), 320-333.

Yang, C., Rangarajan, A., & Ranka, S. (2018, June). Global model interpretation via recursive partitioning. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)* (pp. 1563-1570). IEEE.

Yu, S., & Abraham, Z. (2017, June). Concept drift detection with hierarchical hypothesis testing. In *Proceedings of the 2017 SIAM International Conference on Data Mining* (pp. 768-776). Society for Industrial and Applied Mathematics.

Zeng, Z., Miao, C., Leung, C., & Chin, J. J. (2018, April). Building more explainable artificial intelligence with argumentation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., & Ma, S. (2014, July). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (pp. 83-92). ACM.

Zhang, Y., Meratnia, N., & Havinga, P. (2009, May). Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks. In *Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on* (pp. 990-995). IEEE.

Žliobaite, I. (2010, December). Change with delayed labeling: When is it detectable?. In *2010 IEEE International Conference on Data Mining Workshops* (pp. 843-850). IEEE.

Zliobaite, I. (2013). How good is the electricity benchmark for evaluating concept drift adaptation. *arXiv preprint arXiv:1301.3524*.

Zliobaite, I., Bifet, A., Pfahringer, B., & Holmes, G. (2014). Active learning with drifting streaming data. *IEEE transactions on neural networks and learning systems*, 25(1), 27-39.

Zliobaite, I., & Gabrys, B. (2012). Adaptive preprocessing for streaming data. *IEEE transactions on knowledge and data Engineering*, *26*(2), 309-321.

Zupon, A., Alexeeva, M., Valenzuela-Escárcega, M., Nagesh, A., & Surdeanu, M. (2019, June). Lightly-supervised Representation Learning with Global Interpretability. In *Proceedings of the Third Workshop on Structured Prediction for NLP* (pp. 18-28).

CURRICULUM VITA

NAME: Hanqing Hu

ADDRESS: Computer Science and Engineering Department
University of Louisville, Louisville, KY, 40208

EDUCATION: B.S., Miami University, 2012
M.S., Miami University, 2012

PUBLICATION:

Hu, H., & Kantardzic, (2022) M. Heuristic ensemble for unsupervised detection of multiple types of concept drift in data stream classification. Intelligent Decision Technologies, (Preprint), 1-14.

Hu, H., Kantardzic, M., & Sethi, T. S. (2020). No Free Lunch Theorem for concept drift detection in streaming data classification: A review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(2), e1327.

Hu, H., Kantardzic, M., & Lyu, L. (2018, December). Detecting Different Types of Concept Drifts with Ensemble Framework. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 344-350). IEEE.

Hu, H., & Kantardzic, M. (2017, January). Sliding reservoir approach for delayed labeling in streaming data classification. In Proceedings of the 50th Hawaii International Conference on System Sciences.

Hu, H., & Kantardzic, M. (2016, January). Smart preprocessing improves data stream mining. In 2016 49th Hawaii International Conference on System Sciences (HICSS) (pp. 1749-1757). IEEE.

Hu, H., Kantardzic, M. M., & Sethi, T. S. (2013, October). Selecting samples for labeling in unbalanced streaming data environments. In 2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT) (pp. 1-7). IEEE.