The Islamic University of Gaza

Deanery of Post Graduate Studies

Faculty of Information Technology

Learning Concept Drift Using Adaptive Training Set Formation Strategy

By:

Sarah N. Kohail (220093066)

Supervised by:

Prof. Nabil M. Hewahi

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology

2011 م – 1432هـ

ABSTRACT

We live in a dynamic world, where changes are a part of everyday ‘s life. When there is a shift in data, the classification or prediction models need to be adaptive to the changes. In data mining the phenomenon of change in data distribution over time is known as concept drift. In this research, we propose an adaptive supervised learning with delayed labeling methodology. As a part of this methodology, we introduce an adaptive training set formation algorithm called SFDL, which is based on selective training set formation. Our proposed solution considered as the first systematic training set formation approach that take into account delayed labeling problem. It can be used with any base classifier without the need to change the implementation or setting of this classifier.

We test our algorithm implementation using synthetic and real dataset from various domains which might have different drift types (sudden, gradual, incremental recurrences) with different speed of change. The experimental results confirm improvement in classification accuracy as compared to ordinary classifier for all drift types. Our approach is able to increase the classifications accuracy with 20% in average and 56% in the best cases of our experimentations and it has not been worse than the ordinary classifiers in any case. Finally a comparison study with other four related methods to deal with changing in user interest over time and handle recurrence drift is performed. Results indicate the effectiveness of the proposed method over other methods in terms of classification accuracy.

Keywords: Concept Drift, Adaptive Learning, Training Set Formation, Delayed Labeling, Machine Learning.

عنوان البحث:

تعليم الآلة على التغيرات الغير متوقعة عن طريق إعادة تشكيل مجموعة التدريب

ملخص:

لقد شغلت محاكاة عقل الإنسان في تعلمه واستنتاجه حيزاً كبيراً من اهتمام الباحثين منذ زمن بعيد. ومع تقدم العلوم وتطور أجهزة الحاسوب تم إنشاء العديد من الأبحاث والخوارزميات والأنظمة الناجحة في مجالات متعددة كالطب والأرصاد الجوية وأمن المعلومات ومعالجة اللغ ة وغيرها. فقامت المؤسسات بالاستفادة من البيانات المخزنة (مصنفة سابقاً إلى تنبؤاتها الصحيحة) كمجموعات تدريب لخوارزميات تعليم الآلة وذلك لاستخراج مجموعة المعارف منها وبناء أنظمة للتنبؤ بالمستقبل وبالتالي اتخاذ قرارات صحيحة.

قد تعمل هذه الطرق التقليدية بشكل ممتاز ما لم تطرأ تغيرات تخل بصلاحية النظام وهذا يطبق على قليل من الأنظمة. فالتغير جزء لا يتجزأ من حياتنا اليومية. وعند حدوث تغيرات في بيئة النظام، يصبح النظام غير دقيق أو غير مجدي للعمل في البيئة الحالية. لقد صنفت هذه المشكلة كواحدة من أكبر عشر مشكلات تواجه الباحثين في مجال تعليم الآلة وتنقيب البيانات.

في هذا البحث سنقوم بعرض نموذج حل وخوارزمية ونظام للمساعدة في إعادة تشكيل بيانات التدريب والاستفادة منها تلقائياً بما يتناسب مع البيانات الحديثة الحالية (الغير مصنفة إلى تنبؤاتها بسبب تأخر التصنيف). الخوارزمية المقترحة تعمل مع جميع خوارزميات تعليم الآلة (للتصنيف أو التنبؤ) بدون الحاجة لتعديلها الأصل أو الإضافة عليها. كما وتعتبر هذه الخوارزمية هي أول طريقة منتظمة ومنهجية لحل المشكلة مع الأخذ بالحسبان مشكلة تأخر التنبؤات الصحيحة.

تم اختبار النظام المقترح على ست أنظمة متعددة المجالات، كما وتم الاختبار على جميع أنوع التغيرات المفاجئ منها والمتدرجة والمتراكمة وأخيراً المتكررة. اختبار النموذج المقترح اظهر تفوق النظام في جميع التجارب على الطريقة التقليدية، كما وأظهرت التجارب قدرة النظام على زيادة نسبة دقة النظام التي انخفضت بسبب عامل الزمن بمعدل 20% في كل التجارب وإلى ما يقارب الـ 56 % في أحسن حالة، ولم يكن النظام في أي حال من الأحوال أسوء من الطرق التقليدية في التصنيف.

في النهاية تم عقد دراسة مقارنة بين الطريقة المقترحة وأربع طرق أخرى في إمكانية التعرف على تغير اهتمام المستخدم في المجموعات الإخبارية المختلفة (طب، رياضة، الصحة) وتصنيفها بشكل صحيح وبالتالي بناء نظام لاقتراح الأخبار يتأقلم مع اهتمام المستخدم عبر الزمن. وقد أظهرت المقارنة تفوق الطريقة المقترحة على الطرق الأخرى.

الكلمات المفتاحية: التنبؤ بالتغيرات، خوارزميات التأقلم، مشكلة تأخر التصنيف، تشكيل مجموعات التدريب، تعليم الآلة.

To The Memory of My Father:

Nabil Kohail

ACKNOWLEDGEMENT

The journey has been challenging and exciting. My warm gratitude goes first to ALLAH who guide me to achieve my thesis and accept my prayers. Second to people who inspired me and helped in many ways.

I kindly thank my supervisor **Prof. Nabil M. Hewahi** for his constant guide, challenging discussions and advices. I am grateful to him for working with me. I learned so much, it has been an honor.

I would like to express my appreciation to the academic staff of information technology program at the Islamic University-Gaza.

Sarah N. Kohail
October, 2011

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION

F functional mapping

D training set

n number of instances in a dataset

η is the number of classes in a dataset

q dimensionality or feature space

$D^{(j)}$ a subset of D that include instances belong to class j

$\{ D^{(j)} \}$ is the number of instances corresponding to class j

x_i the ith instance

$x_z^{(i)}$ is the ith feature of the instance x_z

y_i the label of x_i

c_i class label (the ith class)

υ_i is the center of class c_i

$\{c_i\}$ is the number of instances belong to class c_i

k size of the neighborhood (number of the nearest neighbors)

$d(x_z , x_i)$ is the Euclidean distance between x_z and x_i

D^H historical labeled data

υ_j^H is a center of class j in historical data D^H

D^B new batch (unlabeled data)

m is the number of classes in D^B ($m \leq \eta$)

ACRONYMS

AI	Artificial intelligence
CCP	Conceptual Clustering and Prediction Framework
DSTC	Data stream based traffic classification method
IBL	Instance-based learning algorithms
JDK	Java Development Kit
KDD	Knowledge Discovery in Databases
kNN	the k-nearest neighbor classifier
ML	Machine learning
MLP	Multilayer Perceptron Neural Network
SFDL	Adaptive Training Set Formation Algorithm for Delayed Labeling
SIC	Simple Incremental Classifiers
SVM	Support Vector Machine
TW	Time Window method
VFDT	Very Fast Decision Tree
WAH	Window Adjustment Heuristic
WE	Weighted Examples Methods

CHAPTER 1
INTRODUCTION

The world is full of data. The evolution of information science and technology has so explosively increased the amount of data that there is too much data for humans to analyze themselves. Therefore, humans have invented machine learning. Machine learning (ML) is a branch of artificial intelligence (AI) that concern with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. As computing field, ML has become steadily more successful in applications over the past 20 years. Learning approaches such as data clustering, neural network classifiers and nonlinear regression have found surprisingly wide application in the practice of engineering, business, and science [24].

A research field closely related to machine learning is that of knowledge discovery and data mining. With the advent of high-throughput experimental technologies and of high speed internet connections, generation and transmission of large volumes of data has been automated in the period [1998-2007]. As a result, science, industry, and even individuals have to face the challenge of analyzing and dealing with large datasets which are too big for manual analysis. While these large "mountains" of data are easily produced nowadays, it remains difficult to automatically "mine" for valuable information within them [43]. "Data Mining", often also referred to as "Knowledge Discovery in Databases" (KDD) [4], is a young sub-discipline of computer science aiming at the automatic interpretation of large datasets.

Yang and Wu [40] identify ten challenging problems in data mining research by consulting some of the most active researchers in the field. One of the important and hot problem listed as tenth problem of the challenging problem in data mining research is dealing with non-static data. We live in a dynamic world, where changes are a part of everyday life. When there is a shift in data, the classification or prediction models need to be adaptive to the changes. In data mining the phenomenon of change in data distribution over time is known as concept drift [36]. To show the importance of this problem, assume a data mining application for spam filtering that is developed using dataset generated in current year. As this filter adapted to contend with today's types of spam emails, the spammers alter and confuse filters by disguising their emails to look more like legitimate email. So new spam will be generated and in this case current application will go toward approximation to classify these strange patterns and this will lead to less accurate, poor performance and incorrect knowledge. This dynamic nature of spam email raises a requirement for update in any filter that is to be successful over time in identifying spam [6].

The main difficulty in mining non-stationary data like spam, intrusion, stock marketing, weather and customer preferences is to cope with the changing of data concept. The fundamental processes generating most real-time data may change over years, months and even seconds, at times drastically. Effective learning in environments with hidden contexts and concept drift requires a learning algorithm that can detect context changes without being explicitly informed about them, can quickly recover from a context change and adjust its hypotheses to a new context, and can make use of previous experience in situations where old contexts and corresponding concepts reappear [27].

In this thesis we consider the problem of concept drift in supervised learning where the true classification for each instance (label) is delayed. In particular, we are interested in the training set formation strategy which is able to reform the training sets after considering each concept drift , this will lead to achieving adaptivity to concept drift.

In the rest of this chapter, we give an introductory background to the main topic of this thesis, namely concept drift problem and detectability of concept drift when labeled is delayed. We present the existing general concept drift learning strategy and concentrate on training set formation strategy. Later, we define and narrow down our research problem, formulate the general objectives, summarize the main contributions of the thesis and present its significance. We then state the general used strategy to accomplish the research. Finally, we present the structure of the thesis.

1.1 Learning under Concept Drift

Supervised learning is the task of inferring a function ***F*** from supervised training data. The training data consist of a set of training examples. In supervised learning, each example is a pair of objects input vectors ***x*** and output labels ***y***. The task is to predict the output labels $\boldsymbol{y'}$, having input vectors of a testing data $\boldsymbol{x'}$. By default it is assumed in supervised learning that the training and the testing data (or operational data) come from the same distribution. If distributions change over time, what will happen to prediction accuracy if the same $\boldsymbol{F}$ is still applicable. This problem, known as concept drift [13, 45].

First present of concept drift causes was by Kelly et al. [13]. They claim that change in outcome distribution (concept drift) may occur in three ways: Firstly, and most simply, the prior probability for the class, $\boldsymbol{p(y)}$ may change over time. Secondly, the distributions of the classes may change; that is, the $\boldsymbol{p(xly)}$, may alter over time. Thirdly, the posterior distributions of class memberships, the $\boldsymbol{p(y|x)}$ may alter. Where $\boldsymbol{x}$ is an instance in q-dimensional feature space and $\boldsymbol{y \in \{ c_1, \dots , c_m \}}$, the set of class labels.

To simplify the meaning, concept drift is an unforseen substitution of one data source S1 (with an underlying probability distribution Π_{S1}), with another source S2 (with distribution Π_{S2}). As concept drift is assumed to be unpredictable, periodic seasonality is usually not considered as a concept drift problem. As an exception, if seasonality is not known with certainty, it might be regarded as a concept drift problem. The core assumption, when dealing with the concept drift problem, is uncertainty about the future we assume that the source of the target instance is not known with certainty. It can be assumed, estimated, or predicted, but there is no certainty [45].

Figure 1.1 shows four main types of changes that may occur in a single variable along time assuming one dimensional data. We depict only the data from one class. By change types we mean the patterns the data sources take over time. The types of concept drift are defined based on those patterns.

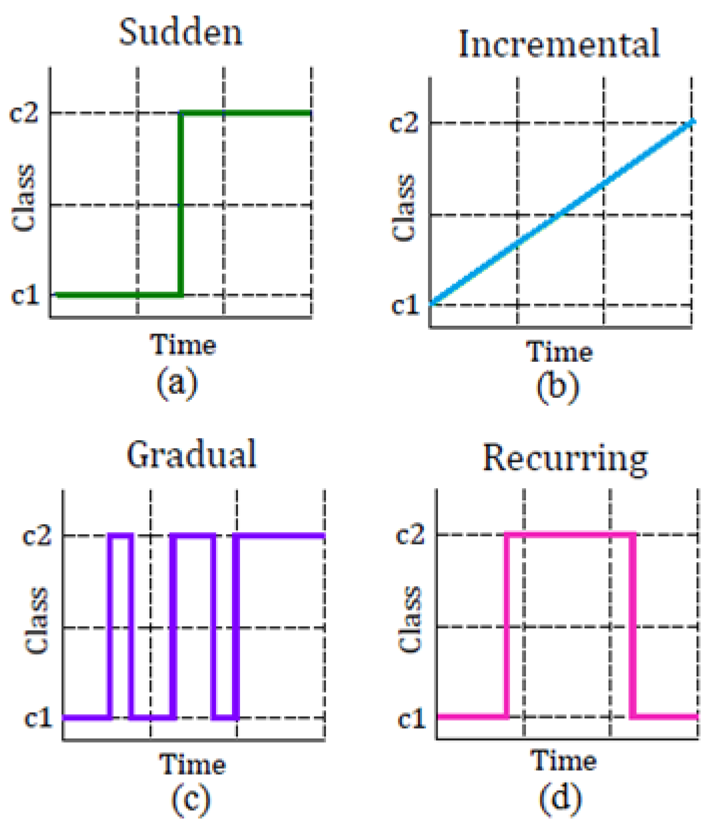


Figure 1.1: Illustration of the four structural types of concept drift [2].

The simplest pattern of a change is a sudden drift illustrated in Figure 1.1 (a). Sudden drift shows abrupt changes that instantly and irreversibly change the variables class assignment. Real life examples of such changes include change of person interest, customer preferences, e-commerce environment and stock prices. The next two plots Figure 1.1 (b) and (c) illustrate changes that happen slowly over time thus the drift is noticed only when looking at a longer time period. Incremental drift occurs when variables slowly change their values over time, we can see it as a sequence of small sudden drifts. Gradual drift occurs when the change involves the class distribution of variables. Some researchers do not distinguish these two types of drift and use the terms gradual and incremental as synonyms. A typical example of incremental drift is price growth due to inflation, whilst gradual changes are exemplified by slowly changing definitions of spam or user-interesting news feeds [2].

The forth type of drift illustrated in Figure 1.1 (d) is referred as reoccurring concepts. It happens when several data generating sources are expected to switch over time at irregular time intervals. Thus previously active concepts reappear after some time. This drift is not certainly periodic, it is not clear when the source might reappear, that is the main difference from seasonality concept used in statics. This type of change is regarded by some researchers as local drift [35]. An example of reoccurring drift is changing in food sales.

1.2 Existing Strategies for Concept Drift Learning

In order to overcome the concept drift issue, there are three strategies [5]: (1) every certain period -which depends on the particular application- a new system is developed using all the available data. This strategy is time, finance, and computation cost. In addition, when a certain data-mining algorithm considers all past training examples with new one, the induced patterns may not be valid and relevant to the new data. (2) build the system from early beginning to be adaptable with changes by adding new inputs may be better at explaining the causes of the concept drift, this strategy can be applied using many current methods CART, ID3, C4.5, IFN and multilayer perception (e.g. for sales prediction application adding information “features” about the season can reduce concept drift). This strategy is not suitable for many applications that the changes in environment are unpredictable. (3) discard an old model and train a new one using the new data. This strategy computationally more efficient than “learning from scratch” and provide further insights into the changes of the respective environment. But there are several problems associated with this.

- We can’t predict the exact time of change thus it is not known with certainty, when to discard and retrain.
- The changes might not be sudden but gradual, the contexts might reoccur, thus the exact point of change is not identifiable.
- The new data after the change is scarce. Thus the data after the change might not be enough to train the new learner accurately.

To deal with these problems Žliobaitė and Pechenizkiy [44] identify four main adaptivity areas that can be incorporated into all parts of the learning process; the first is base learners [3] (e.g. add/delete decision tree nodes dynamically). The second is parameterization of the learners can be adaptively manipulated [23] (e.g. dynamically change the neural networks weights). The third approach is adaptive training set formation [17, 18, 20, 45] (e.g. Training set selection, training set weighting and training set manipulation) which is the scope and focus of our thesis explained in the next section. The fourth adaptivity area is classifier ensemble [10, 37, 41] where classification outputs of several models are combined or selected to get the final output. The combination or selection rules are often referred as fusion rules.

1.3 Training Set formation Strategy

In the previous section we present how adaptivity for concept drift problem could be achieved. In this section we will focus our attention on training set formation strategy that is the subject of the thesis work.

Training set formation can be decomposed into:

1. **training set selection**: used to select the most relevant examples to current concept. The relevancy here related to how representative or important older examples are for predicting new instances of the possibly changed concept. For example, instead of taking all the training history, a number of the instances that is strongly related to the current distribution are considered. Training set selection can applied in two ways [36, 45]:
    a. Sequential instance selection (training windows strategies): training window strategies select the nearest neighbors in time to form a training set. Training window strategies are preferred when sudden drift expected. See Figure 1.2 (a) for visualization.

    b. Selective sampling (instance selection): In this case closest instances in the feature space to the target instance are selected to form a training set. Selective sampling in space is particularly beneficial when reoccurring or gradual concepts are expected. See Figure 1.2 (b) for visualization.

2. **training set weighting:** in this case instances can be weighted according to their age, and their competence with regard to the current concept. Klinkenberg [18] shows in his experiments that instance weighting techniques handle concept drift worse than analogous instance selection techniques, which is probably due to overfitting the data.

3. **training set manipulation:** a concept drift may lead to a different relevance pattern of the features describing the observations. Features or even combinations of attribute values that were relevant in the past may no longer be enough discriminatory. Training set manipulation include feature reselection (use dynamic feature space by time), adding new labels that appear with time and delete labels that disappear with time.

At the end of this section we can say that training set formation methods have an advantages over other adaptivity methods since they do not require complicated parameterization and they can be used for online learning plugging in different types of base classifiers.

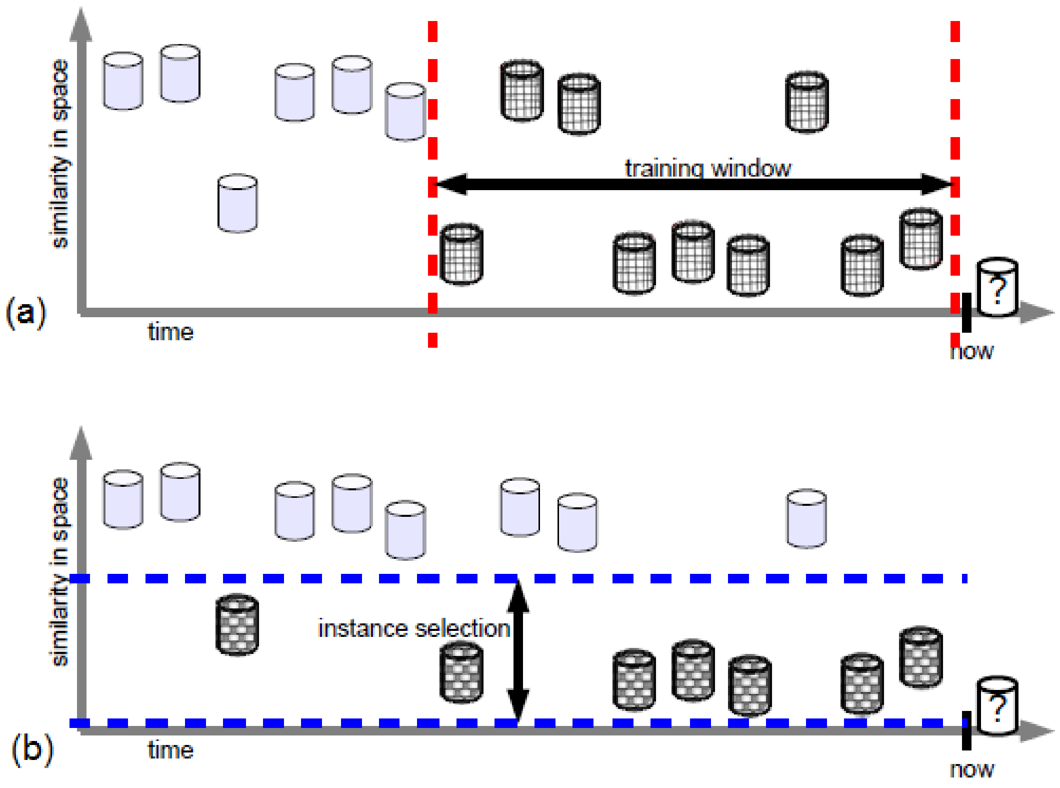


Figure 1.2: Training set selection (a) based only on similarity in time (training window), (b) based only on similarity in the feature space (selective sampling) [47].

1.4 Concept Drift under Delayed Labeling

Learning concept drift includes two tasks: change detection and learner adaptivity respectively [20]. Learner adaptivity might be achieved using one of the four approaches discussed in the previous section. According to change detection task, drift learner can be either trigger based or evolving [44]. Trigger based means that there is a signal which indicates a need for model change. The trigger directly influences how the new model should be constructed. Most often change detectors are employed as triggers. The evolving methods on the contrary do not maintain an explicit link between the data progress and model construction and usually do not detect changes. They aim to build the most accurate classifier by maintaining the ensemble weights. They usually keep a set of alternative models, and the models for a particular time point are selected based on their performance estimation.

Change detection is not just a task to decide whether or not the distribution change, but also it must analyze and give exact reasons about the change. This is important to choose the suitable adaptivity strategy. Most of the work to date on both drift detection and drift handling assumes that the true class of all instances in the data stream will be known shortly after classification [6, 21, 37]. Under such assumption the incoming new data can be regularly used to update the model. Some works like Lindstrom et al. [20] use active learning technique which is used to build classifiers from large collections of unlabeled examples with the assistance of a human expert. The human expert is asked to label only those examples that are deemed to be most informative to the training process. In this way the accuracy of the model examined periodically and real error could be computed.

In real sequential classification tasks, it is not realistic to require labeling every time step, since in many domains collecting labeled training objects may be costly (e.g. require sensors and hardware systems), time-consuming (e.g. require manual human inputs), dangerous or destructive, while it is relatively easy to obtain unlabelled objects [21]. Examples of tasks where delayed labeling exist are sales prediction, bankruptcy prediction, outcome of patient treatment, intrusion or fraud detection and spam categorization tasks. So, a main question arise, how could we benefit from the unlabelled data until the labels become available and how could we extract changes from new data and update our classifier to be consistent with incoming changes?

Most of the research is devoted to solve change detection with delayed labeling problem inspired by statistics. Researcher use methods like posterior probabilities estimation [46], statistical distance function [14], univariate statistical tests and decision tree [8] or nearest neighbor based statistics [30] to detect changes between labeled set and new arriving unlabeled set.

1.5 Research Problem Statement

We formulate the following problem statement:

How to build, develop and implement adaptive supervised learning model with delayed labeling that is able to handle concept drift using training set formation strategy in order to improve the classification and prediction accuracy that dropped by time?

1.6 Research Objectives

1.6.1 Main Objective

The main objective of this research is to build, develop and implement an efficient adaptive training set formation approach to learn and deal with concept drift problem in supervised learning when labels of new arrived data is delyaed. We shall try to increase the classification accuracy of ordinary classifier (old classifer) that is dropped over time due to change.

1.6.2 Specific Objectives

- Build an effective model to classify new arriving data correctly in the absence of its true class labels and reform the old data according to changes detected in new data.
- Implement the proposed model.
- Apply our proposed model on various domains with different drift types and evaluate the results.
- Compare our proposed method with other existing methods.

1.7 Research Scope and Limitation

This research proposes a concept drift learner where adaptivity to changes in data over time is achieved by selective training set formation. The work is applied with some limitations and assumption such as:

1) To select most representative training set, we will integrate the time similarity (sequential selection) and feature space similarity (selective sampling).
2) Our work is limited for supervised learning with single class label.
3) We assume that we receive a set of instances (batch learning) where we can decide if the system change or not and we do not consider real time classification.

1.8 Significance of the Thesis

1) Add a significant contribution to scientific research in solving concept drift research problem.
2) Helping concerned people working in various domains that have concept drift to obtain a better prediction for classification.

1.9 Research Methodology

In our research, we devote our study on automatic classification based on timely fashioned unlabeled instances. In our process we shall use adaptive supervised learning technique with delayed labeling. This is done to change and update the training set by what is called formation methods. We follow a research methodology that consists of the following:

1) **Literature survey:** this include reviewing the recent literature closely related to the thesis problem statement and the research question. After analyzing the existing methods, identifying the drawbacks or the lack of existing approaches, we formulate the strategies and solutions how to overcome the drawbacks.
2) **Develop the algorithm:** to solve the research problem we build a new algorithm to solve concept drift problem using training set selection strategy with respect to a particular focus area. Chapter 3 depict our proposed algorithm.

3) **Implement the algorithm:** using Java programming language we will implement our algorithm.
4) **Design experimental scenarios and apply it to various domains**: to verify the developed algorithm we try various suitable real problems and artificial drift with corresponding datasets that are commonly used in concept drift research.
5) **Evaluate the obtained results:** in this stage we will analyze the obtained results and justify our model feasibility by comparing it with other approaches.

1.10 Outline of the Thesis

The thesis is organized as follows. Chapter 2 present some related works. Chapter 3 includes the methodology and model architecture. In Chapter 4, we present and analyze our experimental results. Chapter 5 will draw the conclusion and summarize the research achievement and future directions.

CHAPTER 2
RELATED WORKS

This chapter intends to give an overview to approaches related to the main topics of this thesis. The problem of concept drift has draw much attention recent years. The researches in this field differ from one another according to how the author look to the problem. This creates a new topics, ideas and challenges. On the other hand the problem receives new aspects and names which makes it difficult to follow. For example some researchers have treated the drift problem as noise or outlier analysis which creates some of mispresentation and misunderstanding for the real problem.

To simplify our literature review presentation we use the taxonomy proposed by Žliobaitė [48]. This taxonomy clearly visualize the main contribution on adaptive supervised learning techniques. The taxonomy is graphically presented in Figure 2.1. From the taxonomy, we can divide the contribution on adaptive supervised learning into two parts: works that concentrate on building a concept drift detection based algorithms and others which are interested in finding ways to keep the base learner updated with every change happen.

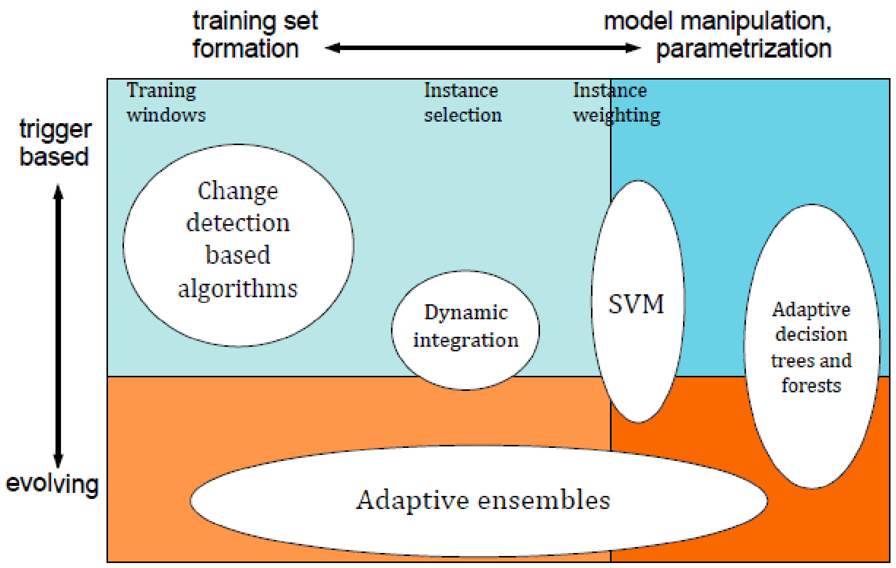


Figure 2.1: A taxonomy of adaptive supervised learning techniques [48].

Change detection can be based on monitoring the raw data [14, 36], the parameters of the learners [33] or the outputs (error) of the learners [17, 20]. Dries and Dries and Ruckert [7] developed change detection methods in each of the three categories. Unfortunately the majority of existing change detection algorithms typically requires a large number of labeled data in order to achieve performances at satisfactory levels; and these algorithms generally assume the availability of such labeled data [39]. An interesting analytical study conducted by Žliobaitė [46] to formulate and address the problem of concept drift under delayed labeling. She studied the types of changes and showed what types of concept drift are detectable from changes in the data distribution and what types would require labels or additional external features for detection detected. In chapter 1 we said that most of researches devoted to solve change detection when labels are delayed are inspired by statistics or depend on active learning where data is manually labeled by experts on demand. Another research try to reduce the amount of manual labeling required by using classification confidence criteria done by by Lanquillon [19].

A work in [29] try to make changes easy to be visualized and tracked using visualization technique. The technique uses parallel histograms to aid in understanding concept drift in multidimensional problem spaces and illustrates the relationship between changes in distributions of multiple antecedent feature values and the outcome distribution.

After change have been detected the designer of the learner must choose the mechanisms which will make the learner adaptive. The pure adaptive learning methods take into account every new instance that arrives. Probably the first systems capable of handling concept drift were STAGGER [31] and FLORA [38].

STAGGER [31] is an incremental learning system that dynamically tracks changes of concepts. STAGGER uses a connectionist representation scheme employing nodes to represent attributes and Bayesian-weighted connections to associate attribute nodes to a concept node. STAGGER learns and tracks changing concepts by adding new attribute nodes or adjusting the connection weights for the concept's connections.

FLORA [38] is a large series of pure drift learning algorithms proposed by Widmer and Kubat. It flexibly react to concept drift and can take advantage of situations where a context repeats (reoccurring concepts) itself. The idea behind their algorithms is that the learner trusts only the latest examples, these examples are referred to as the window. Examples are added to the window as they arrive, and the oldest examples are deleted from it. Both of these actions (addition and deletion) trigger modifications to the current concept hypothesis to keep it consistent with the examples in the window. In the simplest case FLORA1 (the first version of FLORA), the window will be of fixed size, and the oldest example will be dropped whenever a new one comes in. Note that FLORA algorithms considered as training set selection strategy that depend on Time Window (TW) method which classifies incoming instances based on the knowledge of the latest N examples.

The FLORA family pass through many stages of development; the first development is FLORA2, which maintains a dynamically adjustable window during the learning process. The heuristic for adjusting the size of the window is known as WAH (Window Adjustment Heuristic). WAH shrinks the window and forgets old instances when a concept drift seems to occur (a drop in accuracy) and keeps the window size fixed as long as the concept seems to be stable. Otherwise, the window keeps growing until the concept seems to be stable. FLORA3 version stores concepts in stable situations and reuses them whenever a similar context re-appears. In environments with small number of contexts, the process of relearning speeds up due to storage of past concepts. FLORA4 is designed to be exceptionally robust with respect to noise in the training data [38].

Although FLORA family applied successfully in many domains and provides high accuracy, it suffers from two problems lie in the WAH. First problem is WAH dependents on many parameters that require many tuning cycles to reach adequate performance. Widmer and Kubat [25] claim that using the idea to predict parameter depending on stored experience and behavior can solve this problem. Also optimization algorithms can play a good role in this problem. The second problem is that windows adjustment depends mainly in examples age factor leading to significant loss of useful knowledge lies in old data [37]. Instead of discarding data using the criteria based solely on their arrival time, decisions must be made based on their relation to current concept. However it is difficult to decide what are the examples that represent outdated concepts (window adjustment), and hence their effects should be excluded from the model. Adaptive window adjustment heuristic draw much of researchers attention [15, 16, 38]. A commonly used approach is to 'forget' examples at a constant rate or use instance weighting strategy which is also called Weighted Examples (WE) method [18, 36]. A comparative review of forgetting mechanisms for partial memory learning can be found in [22]. Instance weighting strategies use the ability of some learning algorithms such as Support Vector Machines (SVMs) to process weighted instances [18]. An alternative solution to forgetting mechanism and instance weighting is using distance function which select relevant instances to current concept based on space and time similarity. This solution is used by Žliobaitė [47] to build adaptive training set selection when the drift is gradual.

Effective adaptive approach must maintain the relationship between new and old knowledge. This is what most ensemble learning suffer from like what is proposed in [41]. Ensample learning is a learning paradigm where a collection of a finite number of different models is trained for the same task. Ensemble learning can maintain a set of concept descriptions, predictions of which are combined using voting or weighted voting, or the most relevant description is selected. In incremental based ensemble learning for each new examples (that represent new concept), a new model is trained on the new concept, then this model added to the ensemble to work. In some applications that deal with recurring drift, they define previously a fixed set of classifiers each corresponds to one concept and incrementally updates its knowledge by time. This method called Simple Incremental Classifiers (SIC) [11, 36]. Some ensemble learning wait for many examples to be generated to update its existing classifiers (in the case of SIC) or add new one, and the others uses instance-based learning (IBL) algorithms (e.g. IB3) [1], that generates and add models using only specific instances. Sometimes the instance based learning fail to distinguish between true concept changes and noise. Systems that are designed primarily to respond quickly to concept change (e.g., instance based learning) may overreact to noise; on the other hand systems that are designed primarily to be highly robust against noise may not adapt to real changes. Concept drifting learner should combine robustness to noise and sensitivity to concept change.

In the context of ensemble learning, Nishida and Yamauchi [27] propose a system that include multiple online and offline classifiers. Online classifier are used for learning changing concepts, which continue to learn examples in order to adapt to gradual changes, and offline classifiers, which are not updated to handle recurring concepts. The class prediction is determined by selecting one classifier, which adapts well to the current concept, from all online and offline classifiers. The system can detect the concept change by monitoring the classification errors. When change is detected the system adds a new online classifier to respond quickly to this change. The system also clusters classifiers in order to understand the relationship between knowledge and explore hidden contexts of past concepts to predict the next concept, therefore it will respond quickly to sudden changes. Although all this advantages, the system didn't provide accepted efficiency in terms of computation and memory cost. It misses the procedure that removes redundant classifiers (i.e., additional memory) without decreasing the ability to handle recurring concepts. Also it fails in building a good relationship between the knowledge in each classifier, and the system can't determine the reason of the concept drift precisely. Katakis et al [11] try to deal with these problems. They propose a conceptual clustering and prediction framework (CCP) for classifying data streams by exploiting incremental clustering in order to dynamically build and update an ensemble of incremental classifiers. Figure 2.2 illustrate the main components of the CCP framework.

CCP framework consist of three components:

1. Mapping Function: that maps batches of examples into a new conceptual feature space is proposed. this procedure tries to ensure that the more similar two batches will be conceptually, the closer in distance their corresponding conceptual vectors will be.
2. The Clustering Algorithm: it work in order to group different concepts and identify recurring contexts.
3. Incremental Classifiers: ensemble is produced by building or updating (if previously exist) an classifier for every concept discovered.

The authors carried out some experiments on spam filtering and news recommendation datasets. The datasets include recurring drift where the user interest change and recur over time. Their experiments evaluation shows the ability of CCP framework to manage and switch between concepts much faster from the drift. Also they provide good classification accuracy in comparison to simple incremental method.

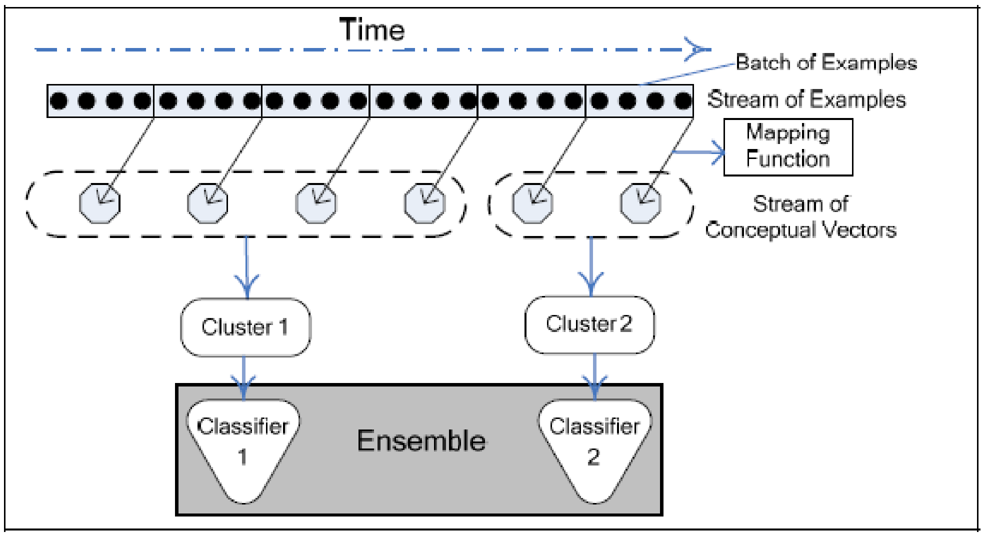


Figure 2.2: Conceptual Clustering and Prediction framework (CCP) [11]

Most of the proposed drift handling methodologies restrict the reason of change to change in time where in many data mining application like intrusion detection and spam filtering, the change is not related to time only but also in feature space. Integration between time and feature space improve generalization performance and drift handling as compared to using only time or only space criterion. One of the proposed methods that consider feature reselection is the one by Tian et. al [34]. Authors present data stream based traffic classification method (DSTC) framework and a data stream mining algorithm, called VFDT (Very Fast Decision Tree) that can achieve online dynamic classification for all kinds of traffic, e.g. encrypted traffic and peer-to-peer traffic, without interpreting packet content. The methodology goes in three steps: traffic model building, online traffic classification and change detection. When a change has been detected, traffic model will be triggered to update model accordingly. This traffic model is responsible for new data preparation, feature reselection and model rebuild. But this model can't work if the size of data is very large.

From the previous work analysis and discussion, we can conclude that the optimal concept drift learner should be able to:

- Respond to sudden, gradual changes and recurring concepts.
- Detect concept drift quickly and recognize the source of drift. Also it should differentiate between noise and concept drift.

- High ability to deal with new data and learn fast from a large amount of data especially high-speed data streams.
- Dynamically create new modules that provide consistent results with existing models results (in case of ensemble learning).
- Memories information and experience, therefore it can predict concept drift early, with high self-optimization and self-healing.
- Adapt only when there is strong evidence that concept drift has occurred, and so reduce the amount of manual labeling required.
- Keep the learning algorithm as effective, efficient, and with as little parameterization as possible.

It is to be noted that most of the used methods for concept drift are using supervised learning as initial training method for the system [42]. This method is found to be an essential procedure for preparing the proposed systems to deal with concept drift.

CHAPTER 3
METHODOLOGY AND PROPOSED MODEL

In this chapter, we present a proposed model for adaptive training set formation that is able to handle drift when label is delayed. We organize this chapter into three sections. Section 3.1 contains the basic fundamentals used in our work. In Section 3.2, we present a general view of our proposed algorithm Training **S**et **F**ormation for **D**elayed **L**abeling Algorithm (SFDL) before we provide the details of each of its steps in Section 3.3.

3.1 Fundamentals

Before going into the details of the proposed approach, we shall present some important fundamentals and basic terminology that we used in our research.

3.1.1 Distance Function (Euclidean distance)

The distance function is used to determine similarity. For numeric attributes distance similarity is usually based on standard Euclidean distance. The Euclidean distance between two points x_z and x_l where each point is a q-dimensional real feature vector is computed as follows [51]:

$$d(x_z, x_l) = \sqrt{\sum_{i=1}^{q} |x_z{}^{(i)} - x_l{}^{(i)}|^2} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.1)$$

here $x_z{}^{(i)}$ is the ith feature of the instance x_z and q is the dimensionality.

3.1.2 k - Nearest Neighbor Algorithm

The k-Nearest Neighbors (k-NN) algorithm is the most basic instance-based method [21, 28]. k-NN is also a lazy learning method where it does not decide how to generalize beyond the training examples until each new input is encountered. The algorithm classifies objects based on closest training examples in the feature space. It is considered as the simplest of all algorithms for predicting the class of a test example. The training phase consists of simply storing every training example with its label. To make a classification for a new example, first compute its distance to every training example. For numeric attributes, the distance is usually defined in terms of the standard Euclidean distance. For Boolean and discrete attributes, the distance is usually defined in terms of the number of attributes that two instances do not have in common. k-NN then keep the k closest training examples in distance, where $k \geq 1$ is a fixed integer. The new example is classified by a majority vote of its neighbors. Figure 3.1 show the pseudo code of k-NN algorithm.

k-NN is robust to noisy training examples and quite effective when it is provided a sufficiently large set of training examples, but storing all of the training examples significantly increases the computational cost to find k nearest neighbors. However this is not a big problem in the existence of current memory chips and physical devices development. Also many memory indexing methods introduced in order to decrease searching and sorting time [28].

#	**Pseudo-code for the basic *k*-NN classifier**
1	**Input:** Training set $D = \{(\boldsymbol{x_1}, \boldsymbol{y_1}), \ldots, (\boldsymbol{x_n}, \boldsymbol{y_n})\}$
2	$\boldsymbol{x'}$ new instance to be classified
3	**Output:** predicted class label $\boldsymbol{y'}$ for $\boldsymbol{x'}$
4	**ALGORITHM**
5	FOR each labeled instance $(\boldsymbol{x_i}, \boldsymbol{y_i})$ calculate $d(\boldsymbol{x_i}, \boldsymbol{x'})$ from equation (3.1)
6	Order $d(\boldsymbol{x_i}, \boldsymbol{x'})$ from lowest to highest, $(i = 1, \ldots, n)$
7	Select the k nearest instances to $\boldsymbol{x'}$: $D_{\boldsymbol{x'}}$
8	Output $\boldsymbol{y'}$ that is the most frequent class in $D_{\boldsymbol{x'}}$

Figure 3.1: *k*-NN algorithm [28]

In addition to the class label outputted by k-NN classifier, we modified the Figure 3.1 so it can output two additional class labels, $\boldsymbol{y''}$ and $\boldsymbol{y'''}$ for the same example. The basic idea of the algorithm does not change, but we add two more computations, one for $\boldsymbol{y''}$ and the other $\boldsymbol{y'''}$. The purpose of doing this computing is to decide later what class label should be assigned to the given drift example. The details of this process and how the values of $\boldsymbol{y''}$ and $\boldsymbol{y'''}$ are used will be explained in the next section. Computation of $\boldsymbol{y''}$ and $\boldsymbol{y'''}$ are illustrated in Figure 3.2.

# Pseudo-code for the modified k-NN classifier

```
1   Input:  Training set D = {( $x_1$ , $y_1$), . . . , ($x_n$, $y_n$)}
2           $x'$ new instance to be classified
3   Output: predicted three different class labels $y'$, $y''$, $y'''$ for $x'$
4   ALGORITHM
5   FOR each labeled instance ($x_i$, $y_i$ ) calculate d($x_i$ , $x'$) from equation (3.1)
6   Order d($x_i$ , $x'$) from lowest to highest, (i = 1, . . . , n).
7   Select k nearest instances to $x'$ that belong to class j, (j = 1,…., ɳ ):$D^{(j)}$, ɳ is the
8   number of classes.
9   Select the $k$ nearest instances to $x'$: $D_{x'}$
10  Output $y'$ that is the most frequent class in $D_{x'}$
11
12  FOR each class j
13    $Summ_j = \sum_{z=1}^{\{D^{(j)}\}} d(x_z, x')$, (where { $D^{(j)}$} is the number of instances corresponding to class j)
14  END FOR
15  $y''$ = class with minimum $Summ_j$
16
17  FOR each class label j
18    Get all instances from $D_{x'}$ that belong to class j : $D_{x'}^{(j)}$
19    IF $\{D_{x'}^{(j)}\} \neq 0$ ({$D_{x'}^{(j)}$} is the number of instances corresponding to class j from the whole set
20    $D_{x'}$)
21      $S_j = \sum_{z=1}^{\{D_{x'}^{(j)}\}} d(x_z, x') / \{D_{x'}^{(j)}\}$
22      END IF
23  END FOR
24  $y'''$ = class with minimum $S_j$
```

Figure 3.2: Modified *k*-NN algorithm

<u>Computing y'':</u>

After ordering the examples according to its distance from x' (line 6), we select the nearest k instances from each available class j, we represent the set of selected instances for class j by$D^{(j)}$. Where j = 1,…., ɳ and ɳ is the number of available classes. Then y'' is assigned to class which the summation of its distances ($Summ_j$) from x' is the minimum.

Computing ***y‴***:

After selecting the k nearest instances (line 9) we sum distances of instances that belong to different class labels and then dividing it by the number of nearest neighbor instances belong to that class label from the total k.

3.1.3 Closest Class

We develop this computation as a heuristic to help us to get the nearest class to current available classes. Many other methods calculate the distance between centers directly to get how much one class is far from the others. These methods may not work well when the distribution of the instance points belong to one class label is scattered and non-intensive. This heuristic guides the algorithm to decide how to change the class label when there is a drift especially when the drift is gradual.

```
#   Pseudo-code for computing the closest class to each available class

1   Input:  Training set D
2   Output: closest class label to each available class $y^{closest}$
3   ALGORITHM
4   Separate instances that belong to each class label in different set: ($D^{(1)}$,..., $D^{(\eta)}$),
5   $\eta$ is the number of classes
6   FOR j=1 to $\eta$-1
7       FOR i=j+1 to $\eta$
8           FOR z=1 to {$D^{(j)}$} (where { $D^{(j)}$} is the number of instances corresponding to class j)
9               pick one instance from $D^{(j)}$ : $\boldsymbol{x_z}$
10              pick random instance from $D^{(i)}$: $\boldsymbol{x_r}$
11              $\boldsymbol{S}$ +=Euclidean distance d($\boldsymbol{x_z}$ , $\boldsymbol{x_r}$) (equation 3.1)
12          END FOR
12      $Average_{j,i} = \frac{\boldsymbol{S}}{\min(\{D^{(j)}\},\{D^{(i)}\})}$
13      END FOR
14  END FOR
15  FOR each class c , (c=1,…., $\eta$)
16      $y_c^{closest} = min$ [ $Average_{j,i}$], (find minimum $Average_{j,i}$ ) where (c=j or
17      c=i)
```

Figure 3.3: Algorithm for computing the closest class to each available class

Figure 3.3 illustrate the Pseudo code for computing the closest class for each existing class. After applying this algorithm all classes will have a closest class.

The input for this algorithm is training data and the output is closest class label for each class available in the training set. It is to be mentioned that if class **X** is the closest class to class **O** it is not necessary that class **O** is the closest class to **X**. To compute the closest class for a particular classc_i, (i= 1,...., ɳ), first the algorithm compute the average between every two classes. The average is computed as follows:

1. The algorithm will separate instances according to their class label.
2. For each two different classes i and j:
    a. For each instance belong to first class i. another random instance will picked from the second class j.
    b. Euclidean distance between the two instances will computed and added to summation S.
3. Summation S will be divided on the number of instances of class with minimum number of instances (either i or j).
4. Now we have a single average for each pair of classes. The number of averages is equal to Binomial Coefficient $\binom{\eta}{2}$ with no repetition and order doesn't matter. This means we have ɳ classes, and we want to choose two (pair) of them each time.
5. The closest class for some class c will be the class which have minimum average with class c.

3.2 Proposed Approach – General View

Before going into details about our proposed approach, it is important to present a general view for this proposed approach to provide a general idea about methodology flow and major steps. Figure 3.4 provides a global view for concept drift learning scenario that we build. In our work, adaptivity to changes (drift) in data over time is achieved using training set formation strategy.

To make the flow clear and complete, we illustrate the followed scenario for the arrival of two consequent batches below and in Figure 3.4 (a) and (b).

Step I: Like most of the proposed methods for concept drift learning, we use supervised learning as initial training method for the system. As usual, in order to solve a given problem of supervised learning (in our case classification) two processes must be performed, training and testing. The goal here is to learn a model from the data that can be used to predict the classes of new (future, or test) cases/instances accurately. After training and testing a classifier, L_t is produced. Classifier L_t is considered as the best and accurate classifier at time t.

Step II: When the system receives new instances (a batch with drifts), the new instances will be classified using L_t classifier. This will continue until instances of window[1] size *w* arrived. This set is considered as a complete batch [x_{t+1} to x_{t+N}]. Window size value is fixed for single system depending on the system designer knowledge of context.

Step III: Apply our proposed algorithm named Adaptive Training Set Formation for Delayed Labeling Algorithm (SFDL) to old historical data (training data used to build L_t classifier) and new incoming batch. The work of this algorithm is summarized as follows:

- Select the most relevant instances to current concept (Instance Selection).
- Reclassifying the new arrived batch using the selected instances.
- Reform the old set according to the changes detected.

Step IV: The output of the previous step is a new formed training set that carry out the changes occurred during the period [$t+1$ to $t+N$]. This set will be used to retrain the model and produce L_{t+N} classifier as illustrated at Figure 3.4 (b).

Step V: When receiving another new batch, the process will be repeated from step II and so on.

[1] Window is a set of instances taken from a fixed time interval.

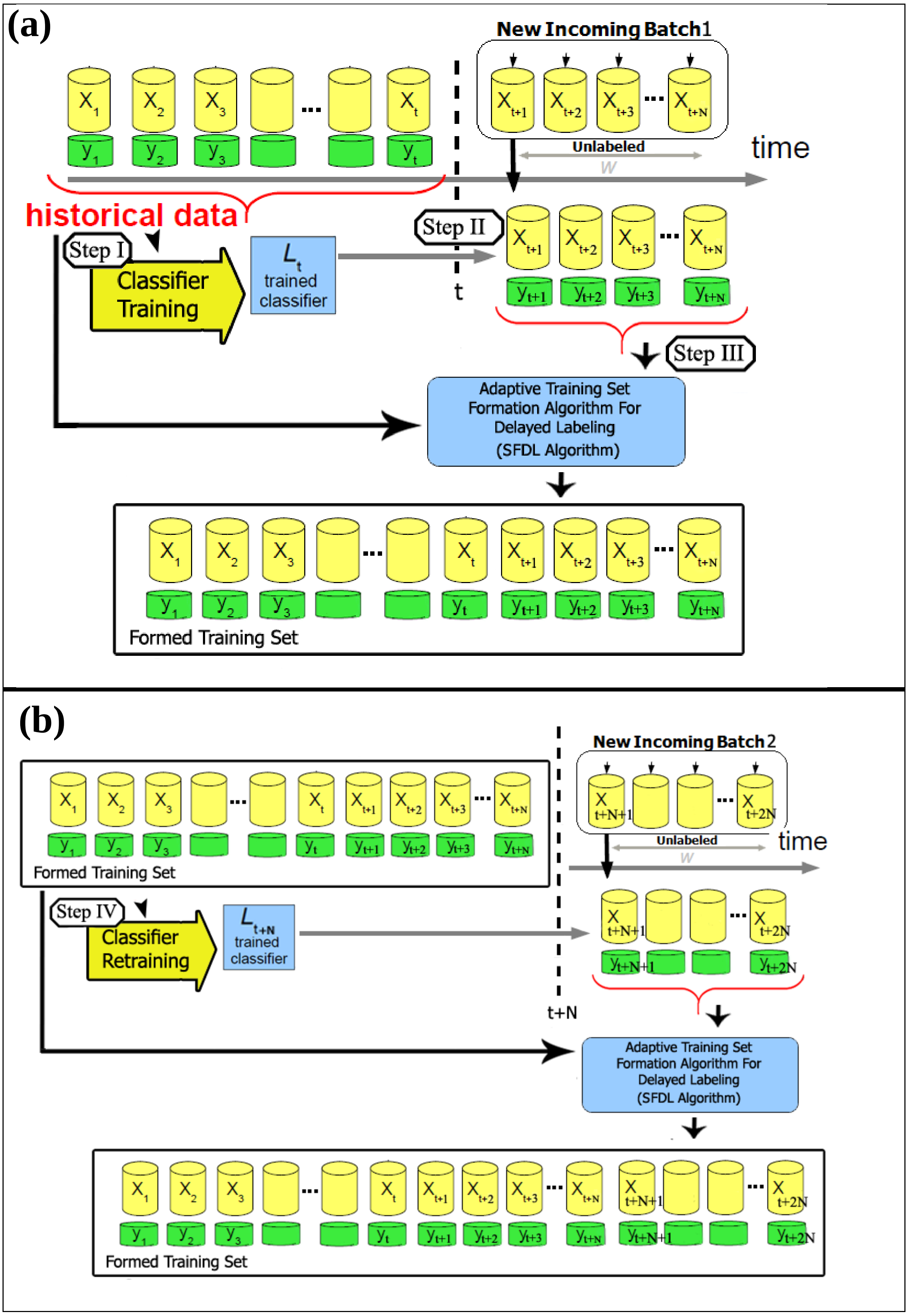


Figure 3.4: Global view for concept drift learning scenario using the proposed approach.

3.3 The Proposed Approach – Detailed Description

From the previous section it is notable that our proposed SFDL Algorithm typically can be used plugging in various base classifiers. No matter what is the type of learner. Our solution concentrate on training set formation strategy. That is to continually update the training data and form it according to changes in the new data. Before explaining our algorithm we should present this equation that explains how parameter alpha (α) is computed. α is a threshold value for each class label that would be used for selecting certain instances close to the given instance example. How α is going to be used is explained in the next section. For the ith class c_i with center v_i, alpha (α) is computed by the following equation :

$$\propto_i = \frac{max_{z=1}^{\{c_i\}}\left[d(v_i, x_z)\right] - min_{z=1}^{\{c_i\}}\left[d(v_i, x_z)\right]}{2} \quad \dots\dots\dots\dots (3.2)$$

Where:

$\{c_i\}$ is the number of instances belong to c_i. i = 1,……, m ; m is the number of classes

$max_{z=1}^{\{c_i\}}[d(v_i, x_z)]$ is the maximum distance between v_i and any instance belong to c_i.

$min_{z=1}^{\{c_i\}}[d(v_i, x_z)]$ is the minimum distance between v_i and any instance belong to c_i.

Note: class center computed as follows:

$$v_i = \frac{\sum_{z=1}^{\{c_i\}} x_z}{\{c_i\}} \dots\dots\dots\dots\dots\dots\dots (3.3)$$

To simplify the algorithm presentation, we divide it to three algorithms according to its role in the whole training set formation algorithm:

- Instance selection (Figure 3.6)
- Reclassifying the new incoming batch (Figure 3.7)
- Training set formation (Figure 3.8)

SFDL Algorithm is illustrated at Figure 3.5.

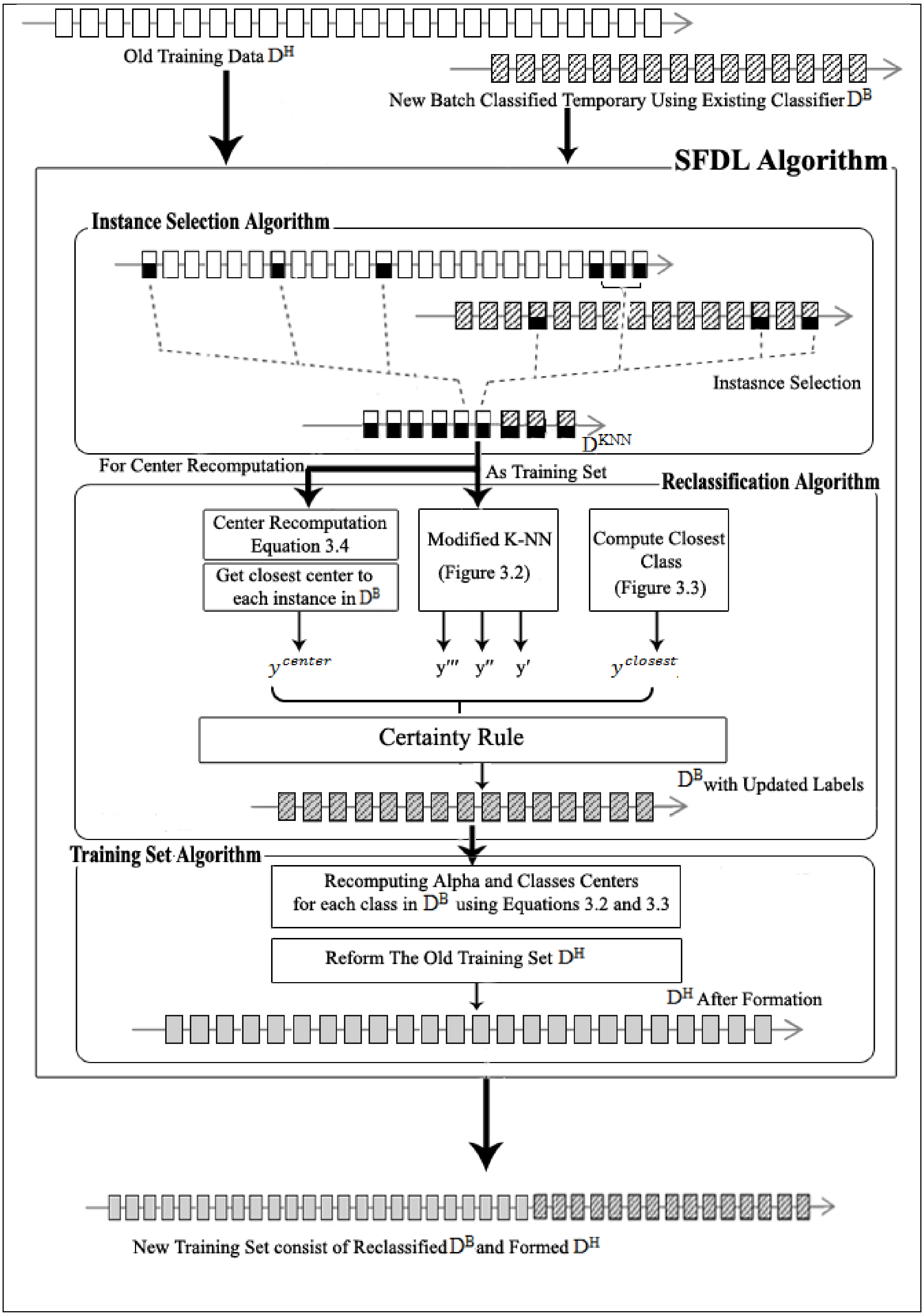

Old Training Data D^H
New Batch Classified Temporary Using Existing Classifier D^B
SFDL Algorithm
Instance Selection Algorithm
Instasnce Selection
D^{KNN}
For Center Recomputation
As Training Set
Reclassification Algorithm
Center Recomputation Equation 3.4
Get closest center to each instance in D^B
Modified K-NN (Figure 3.2)
Compute Closest Class (Figure 3.3)
y^{center}
y‴ y″ y′
$y^{closest}$
Certainty Rule
D^B with Updated Labels
Training Set Algorithm
Recomputing Alpha and Classes Centers for each class in D^B using Equations 3.2 and 3.3
Reform The Old Training Set D^H
D^H After Formation
New Training Set consist of Reclassified D^B and Formed D^H

Figure 3.5: SFDL algorithm

3.3.1 Instance Selection Algorithm

The pseudo code of the algorithm is presented at Figure 3.6. The algorithm used to select the most relevant examples to current concept. The relevancy here is related to how importance older examples are for predicting new instances in term of time similarity and feature space similarity. The algorithm takes five inputs:

- Historical data D^H which is used to build the existing classifier L_t .

- New batch D^B which is arrived during the period [t to t+N] and labeled using classifier L_t .

```
Adaptive Training Set Formation for Delayed Labeling Algorithm (SFDL) – Instance Selection

Input:
List of historical data $D^H$ =( $x_1, . . . , x_t$) with labels ( $y_1, . . . , y_t$).
New unlabeled batch $D^B$ = ($x_{t+1}, . . . , x_{t+N}$) is labeled using classifier $L_t$: ( $y_{t+1}, . . . , y_{t+N}$).
Computed $\propto$ for each class in $D^B$ (equation 3.2): ( $\propto_1 ......\ \propto_m$), m is the number of classes in new batch $D^B$.
Computed centers for each class in $D^B$ (equation 3.3): ( $\upsilon_1^B ......\ \upsilon_m^B$).
Integer window $w_{resent}$ .
Output:
Selected training set $D^{KNN}$ that is very close to current concept and set of far instances $D^{FAR}$
ALGORITHM
FOR i= 1 to m
    FOR j=1 to t+N
        IF d($\upsilon_i^B, x_j$) $\le \propto_i$
            Add $x_j$ to $D^{KNN}$
        ELSE
            Add $x_j$ to $D^{FAR}$
        END IF
    END FOR
END FOR

IF $w_{resent}$ > 0
    select most recent $w_{recent}$ instances and add them to $D^{KNN}$
END IF
```

Figure 3.6: Instance selection algorithm

- Computed centers υ and alpha $\propto$ for each class in the new batch using equations 3.2 and 3.3. It is not necessary that the new batch instances are classified to all possible classes, so the number of classes at new batch could be less than the possible classes ($m \leq \eta$).

- Integer w_{resent}. This parameter represents how many respective recent instances will be selected before time t. In some application where the drift is sudden, the time factor is not important, therefore selecting instances according to its age is ineffective. So the designer of the application can set w_{resent} to zero.

The algorithm output is a set of relevant instances to current concept called D^{KNN}. To select instances according to distance similarity, for each class available in the new batch, the algorithm will go through all instances (old and new one) from x_1 to x_{t+N} ignoring its class label and select instances in which the Euclidean distance between the center of this class and the instance is less than its computed $\propto$.

Relevant instances in term of time are selected according to w_{recent} value. The value selection depends on the domain at hand, as well as on the expectations of a designer regarding the drift type.

3.3.2 Reclassifying the New Incoming Batch

Depending on the selected set D^{KNN}, the algorithm will reclassify the new instances, which were initially classified using the available classifier. We reclassify them because we assume that the current classifier maybe become outdated and useless for classifying new instances.

The algorithm illustrated at Figure 3.7. The main inputs for this algorithm are D^H, D^B, D^{KNN} and *k* value that is the size of neighborhood (number of nearest neighbor). The work of the algorithm is summarized in the following points:

- Applying modified k-NN (Figure 3.2) with *k* as a size of neighborhood and D^{KNN} as training set to classify each instance in new batch. Modified k-NN algorithm will return three different classes as explained in section 3.1.2. the original k-NN label $\boldsymbol{y'}$ and two additional labels $\boldsymbol{y''}$ and $\boldsymbol{y'''}$.

- The next step is to update the position of the existing class centers. To do that we compute new center $\upsilon_{\varkappa}$ using this formula:

$$\upsilon_{\varkappa} = \frac{\upsilon_{\varkappa}^{B} + \upsilon_{\varkappa}^{H} + \upsilon_{\varkappa}^{KNN} + (\max_{z=1}^{\{c_{\varkappa}\}}[\, d(\upsilon_{\varkappa}^{B}, x_z)] - \min_{z=1}^{\{c_{\varkappa}\}}[\, d(\upsilon_{\varkappa}^{B}, x_z)])}{4} \quad \ldots (3.4)$$

Where:

$\upsilon_{\varkappa}^{B}, \upsilon_{\varkappa}^{H}, \upsilon_{\varkappa}^{KNN}$: are the centers of class $\varkappa$ in D^{B}, D^{H} and D^{KNN} datasets respectively, $\varkappa = 1 \ldots\ldots \eta$, where η is the number of available classes.

Note: in some cases D^{B} and D^{KNN} do not include all possible classes available in D^{H}, in this case the associated centers ($\upsilon_{\varkappa}^{B}$ or $\upsilon_{\varkappa}^{KNN}$) for missing classes will not be known, so it will set to zero.

Combining "centers of the new objects" with the previous centers help in moving centers smoothly and gradually forget old concept and switch to new one. It is notable that centers of classes which are not changed (not included at new batch) will not be affected by this formula because the new centers will be set to zero.

- After updating the classes position, the algorithm will compute the Euclidean distance between new centers and every instance in new arriving batch D^{B}. The class with closest center to the instance will be set as fourth label y^{center} (in addition to y''', y'' and y').

- Another class label $y^{closest}$ will be computed using Figure 3.3. Unlike y^{center} which represent the closest class to a specific instance, $y^{closest}$ represent the closest class distribution to other class distribution as a whole.

- Now each instance in D^{B} has five different class labels (y^{center}, $y^{closest}$, y''', y'', y'). The five labels are used to decide if some instance will stay with its current class or it must be assigned to other possible closest class. Reclassification of any instance in D^{B} depends on a heuristic certainty rule. If certainty rule is satisfied, the instance will be reclassified to most frequent class label of all five classifications. Otherwise, it will be reclassified to y^{center} .

# Adaptive Training Set Formation for Delayed Labeling Algorithm (SFDL) – Reclassifying New Batch Instances

```
1   Input:
2   List of historical data instances $D^H$ =( $x_1$, . . . , $x_t$) with labels ( $y_1$, . . . , $y_t$)
3   New batch $D^B$ = ($x_{t+1}$, . . . , $x_{t+N}$) labeled using classifier $L_t$ ( $y_{t+1}$, . . . , $y_{t+N}$)
4   Selected instances $D^{KNN}$ and far instances $D^{FAR}$ (Figure 3.6)
5   Nearest neighbor value : $k$
6   Output:
7   New batch $D^B$ with new class labels
8
9   ALGORITHM
10
11   Apply modified k-NN (Figure 3.2) to reclassify each instance in $D^B$ using $D^{KNN}$
12   as training set. (three labels $y'''$, $y''$, $y'$ Returned)
13
14   Recompute classes center using Equation 3.4.
15   $\varkappa$ = 1….. $\eta$ , $\eta$ is the number of possible available classes.
16
17  For each instance in $D^{B,}$ get the closest center (using Euclidean distance) from all
18  new computed centers in line 14 and assign it to $y^{center}$
19
20  For each class in $D^{B,}$ compute closest class (Figure 3.3) ( $y^{closest}$ )
21
22  (Now there is Five different class label ($y^{center}$ , $y^{closest}$ , $y'''$, $y''$, $y'$ ) for each
23  instance in $D^B$) where $y^{center}$ The class with closest center to the instance.
24
25  FOR each instance in $D^B$:
26       IF (Certainty Rule return true)
27           reclassify this instance to most frequent class label
28       ELSE
29           classify the instance to $y^{center}$
30       END IF
31  END FOR
```

Figure 3.7: Reclassifying New Batch Instances algorithm

Certainty Rule dictates the following:

1. The instance is not included at D^{FAR}.
2. There is no uncertainty in classification (between the five labels) of the instance to a specific class. This means that majority in the classification must be clear. For example if two of five classes have been classified to label **X** and two for class **O** and one for class **Z** (2:2:1) in this case we said that there is no certainty, because the voting is very close. The same example if three of five classes have been classified to label **X** and two for class **O** (3: 2). Cases like (3:1:1) and (4:1) reflects a good majority .

By certainty rule we want to determine those instances that are not classified well by the existing classifier, far away from the current classes and have fuzzy membership.

We choose y^{center} to be a label for those instances that do not satisfy certainty rule.

3.3.3 Training Set Formation Algorithm

This algorithm (Figure 3.8) work to reform the old set according to the changes made on reclassification step because the old data needs to be adapted to fit to the new data. The algorithm is very simple. The main functions of this algorithm are:

- Recomputing the centers and $\propto$ for each class in D^B (after reclassifying its instances) using equations 3.2 and 3.3.
- Reform the old set. For each instance in D^H if its distance from any class center υ_i^B is less than $\propto_i$ of the same class, then this instance will be reclassified to its close class according to distance from the center.

The output from SFDL Algorithm is a new training set consists of reclassified new batch (output of algorithm at Figure 3.7) and reformed old set (output of algorithm at Figure 3.8).

Because the used dataset for training and testing new instances will increase gradually after each set formation, a certain criteria could be used to eliminate certain number of instances to have always a dataset that does not exceed a predefined size.

This can be performed proportional to the number of instances related to each class label in the dataset.

Adaptive Training Set Formation for Delayed Labeling Algorithm (SFDL) – Training Formation

```
Input:
List of historical data instances $D^H$ =( $x_1, . . . , x_t$) with labels ( $y_1, . . . , y_t$)
New batch  $D^B$ with its new labels (after applying Figure 3.7)
Output:
Re-formed old training set of $D^H$

ALGORITHM

Recompute $\propto$ for each class in $D^B$ (equation 3.2): ( $\propto_1 \ldots\ldots \propto_m$), m is the
number of classes in new batch $D^B$.
Recompute centers for each class in $D^B$ (equation 3.3): ( $v_1^B \ldots\ldots v_m^B$).

FOR j=1 to t
    FOR i= 1 to m
        IF $d(x_j, v_i^B) \leq \propto_i$
            $y_j$ = the class with closest center to $x_j$ .
        END IF
    END FOR
END FOR
```

Figure 3.8 Training set formation algorithm

CHAPTER 4
Experimental Results and Evaluation

This chapter discusses the experiments carried out to evaluate our proposed model. The chapter includes three sections: Section 4.1 presents all the datasets used in our experimentation and gives insight into the main characteristics of each data set. Section 4.2 briefly describes the experimental environment and states the programming language and tools used to develop the proposed system. Finally, in Sections 4.3 we present and discuss experimental results.

4.1 Datasets

For the purposes of research related to concept drift learning there is no standard concept drift benchmark dataset. Instead there are popularly datasets that were used by most of the existing researches. Unfortunately most of the real word datasets are not suitable for evaluating drift learning because there is a little concept drift in them. So researchers turn to introduce artificial drift in real datasets or create synthetic (fabricated) datasets with artificial drift.

In our experiments we use six data sets with concept drift, all of which are publicly available. The datasets are chosen from various domains that might have different drift types with different speed of change. They include no missing or noise. Table 4.1 illustrates the characteristics of each set. A short description of each data set is given below.

Table 4.1: Characteristics of the used datasets

Name		Size	Dimensionality	Classes	Type of data	Source
STAGGER		120	9	2	Artificial	[26]
SEA		800	3	2	Artificial	[32]
Elec		2973	6	2	Real	[9]
Chess		533	6	3	Real	[49]
Credit		1000	23	2	Real	[52]
Usenet	Usenet1	1500	99	2	Real	[50]
	Usenet2	1500				

STAGGER Dataset:

Each data point is described by 3 features, each with three possible categories: size ϵ {small, medium, large}, color ϵ {red, green, blue} and shape ϵ {square, circular, triangular}. The numerical representation of each data point consists of 9 bits, 3 for each feature. For example, a large, red, square object is encoded as the vector $[0, 0, 1, 1, 0, 0, 1, 0, 0]^T$. Three classification tasks were to be learned in a course of 120 points. From point 1 to point 40, the classes to be distinguished (with class label = 1) are [size = small AND color = red] versus all other values (with class label = 2); from 41 to 80, [color = green OR shape = circular] versus all other values; and from 81 to 120, [size = small OR size = large] versus all other values. The drift complexity in STAGGER dataset lies not only at changing in posterior probabilities but also the change in class balance [26].

SEA Dataset:

Each data point is described by three features, $x = [x1, x2, x3]^T$, where values of $\mathbf{x}$ are uniformly randomly generated from $[0, 10]^3$. Only the first two features are relevant. An instance belongs to class 1 if $x1 + x2 \le \theta$ and belongs to class 2 otherwise, where θ is a threshold value, different for each concept. There are four concepts $\theta = 7, 8, 8.5, 9$. We generate 200 instances for each concept (100 instances for each class label). We insure there is no label noise was added so the two classes are perfectly separated [32].

Electricity Market Dataset (Elec):

Electricity data characterizes electricity demand in Australia, the task is to predict electricity market price. We use the time period with no missing values comprised of 2973 instances collected along a period of 3 months from May 11 to July 11, 1997 from the Australian New South Wales Electricity Market. Class Label has two values 'up' or 'down' indicating the change of the price. In our experimentation each month represents one concept [9].

Chess Dataset:

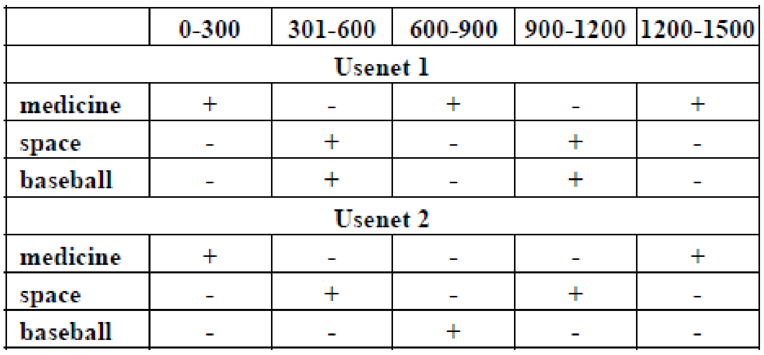
Constructed using the data obtained from chess.com portal. The data consists of game records of one player over a period from 2007 December to 2010 March. A player has a rating, which changes depending on his/her results achieved (the higher is the rating, the stronger is the player). A payer is developing skills over time, besides engages into different types of competitions (personal, tournament or champtionship). The rating and the type of game determine how the system selects an opponent. This is where the concept drift is expected. The task is to predict if the player will win or lose based on the setting. There is natural problem of delayed labeling, the winner is known only after the game is finished. In turn based chess one game might last even for several months [49].

German Credit Approval Dataset (German2):

Classifies customers as having good or bad credit risks. Following [45], a gradual concept change was introduced artificially as a hidden context. We sort the data using one of the features (feature ‘age’ was chosen) and then eliminate this feature from the dataset. Delayed labeling is relevant for this task, since the true label (whether a person fails to repay the credit) is known after some time. However, the decision makers need to know indications of changes right away [52].

Usenet Dataset:

This dataset include two sets usenet1 and usenet2. The sets based on the 20 newsgroups collection. They simulate a stream of messages from different newsgroups that are sequentially presented to a user, who then labels them as interesting or junk, according to his/her personal interests. The difference between two sets lies on the change in one user interest in the various newsgroups over time. This dataset was used to build news recommender systems, document categorization and spam filtering applications [12]. Figure 4.1 shows the news interest change among the batches (we have five batches each contains 300 instances and depicted as the first row in Figure 4.1) and illustrate which newsgroups articles are considered interesting (+) or uninteresting (-) in each time period [50].



	0-300	301-600	600-900	900-1200	1200-1500
Usenet 1					
medicine	+	-	+	-	+
space	-	+	-	+	-
baseball	-	+	-	+	-
Usenet 2					
medicine	+	-	-	-	+
space	-	+	-	+	-
baseball	-	-	+	-	-

Figure 4.1: Dataset Usenet1 and Usenet2 [11]

4.2 Experiments Setup

This section describes the setting of experiments for evaluating our proposed approach. Its primary purpose is to empirically validate the advantages and to notice the shortcomings of our proposed model. Our approach is expected to enhance the classification accuracy which might drop down over time if we use an ordinary classier (a classifier that does not consider concept drift in its approach) because of concept drift. The following describes experimental environment and tools, and experiment procedures.

4.2.1 Experimental Environment and Tools

The experiments took place on a machine equipped with an Intel Pentium Core 2 Duo T8300 @ 2.40 GHz processor and 2.00 GB of RAM. To implement our algorithm we used Java programming language. To carry out our thesis (including the experimentation), special tools and programs were used:

- Microsoft Excel: we use excel to partition, organize and store datasets in tables, do some simple preprocessing and analyze the results.

- Narasimhamurthy and Kuncheva Framework [26]: this framework work to simulate changing environments.We use this framework to generate STAGGER dataset.
- Matlab: we use Matlab to use the implementation of Narasimhamurthy and Kuncheva Framework [26] and generate STAGGER dataset.

- NetBeans 7.0 with integrated JDK (Java Development Kit) 1.6 java environment: NetBeans program helps us to develop, build, compile, validate and execute our algorithm.

- RapidMiner: to preprocess the data and train and test the classifiers.

- Microsoft Word: the program used to write and document the results and experimentations.

4.2.2 Experiment Procedure

The goal of experiments is to observe the system performance as target concepts change from time to time. To achieve the goal we follow this experiment procedure:

1. We start by dividing the dataset into smaller sub sets we called each one as a “batch”. We benefit from previous researches in the way they partition the dataset and insure that every “batch” represent a change [11, 45].

2. We use one batch as a training set for the initial learner. Datasets where instances are ordered according to time, we use oldest batch to be the initial training set. Otherwise we pick a random batch as initial training set, because the drift type in such sets is mostly sudden. Table 4.2 shows dataset partitioning details. The table illustrate type of drift represented, dataset partitioning way, number of instances included in the initial dataset with class balance distribution in percentage, number of batches, number of instances in each batch, class balance for each batch (in percentage) and if dataset is time ordered (Y) or not (N).

3. As we do in traditional procedure, we build the initial learner using the initial training set. We use 10-cross validation with stratified sampling in order to estimate the performance of a learning classifier and ensure we get the best model in current time according to its classification accuracy. The accuracy of the model is calculated using the following equation:

$$Accuracy = \frac{number\ of\ instances\ correctly\ classified}{n} \times 100 \ldots \ldots \ldots \ldots (4.1)$$

Table 4.2: Dataset partitioning details

Name	Drift Type	# instances initial training set (class balance %)	#Batch	Batch Description (Batch, # instances, Class Balance %)	Time Order
Stagger	Sudden	40 (57: 45)	2	Batch1, 40, (10: 90) Batch2, 40, (65: 35)	N
SEA	Sudden	200 (50:50)	3	Each batch include 200 instances with (50:50) class balance	N
Elec	Gradual	1006 (64: 36)	2	Batch1, 1440, (49:51) Batch2, 527 , (62:38)	Y
Chess	Incremental	233(42:54: 4)	3	Batch1, 100, (35: 56: 9) Batch2, 100, (33: 62: 5) Batch3, 100, (32: 60: 8)	Y
Credit	Gradual	400 (63: 37)	6	Batch 1, 100, (70:30) Batch 2, 100, (76:24) Batch 3, 100, (79: 21) Batch 4, 100, (71:29) Batch 5, 100, (79: 21) Batch 6, 100, (71:29)	Y
Usenet	Reoccurring **[Figure 4.1]**	300 (50:50)	4	Usenet1: Batch 1, 300, (33:67) Batch 2, 300, (67:33) Batch 3, 300, (33:67) Batch 4, 300, (67:33) Usenet2: Each batch include 300 instances with (67:33) class balance	Y

4. Next, we pass the first batch, classifying it using current learner (ordinary classifier), apply SFDL training set formation algorithm and retrain the model using formed data. As we mention before SFDL algorithm needs two parameters: (1) number of neighborhood *k* (space similarity) and (2) number of the most recent instances w_{recent} (time similarity). The choice of these two parameters value (time and space combination) directly depends on the observed change types and the future expectations as well as designer knowledge of domain. Parameter setting is fixed for one application run.

5. We measure the accuracy at two points after passing the batch: (1) after its been classified by Algorithm at Figure 3.7. (2) after training set formation and retraining the model.

6. Then we pass the next batch, classify it using the most recent trained classifier after set formation and so on. The procedure explained previously in chapter 3. After each batch classification after set formation, we compare our results with ordinary classifier.

4.3 Experimental Results and Discussion

This section summarizes and discusses the results of numerous experiments that have been conducted.

4.3.1 Sudden Drift Experiments (STAGGER and SEA datasets)

Table 4.3 illustrates experimental results for both STAGGER and SEA datasets. STAGGER dataset partitioned into three parts each represent different concept: the first concept [size = small AND color = red] used to build the initial learner, the second concept [color = green OR shape = circular] as batch1 and third concept [size = small OR size = large] as batch2. For SEA dataset we use the first concept where θ = 7 as to build the initial learner and concepts where θ = 8, 8.5, 9 as batch1, batch2 and batch3 respectively.

Table 4.3: Results of STAGGER and SEA datasets.
Accuracy after the batch reclassification (Figure 3.7) is underlined.
Accuracy after training set formation and model retraining is in bold.

STAGGER	**Naïve Bayes Acc = 100% *k*=2 w_{recent} = 0**		batch1	batch2	
		Ordinary	57.50%	53.66%	
		Arrival of batch1	65.00% **90.00%**	46.34%	
		Arrival of batch2	-	43.90% **65.85%**	
SEA	**Decision Tree Acc = 100% k=10 w_{recent} = 0**		batch1	batch2	batch3
		Ordinary	59.00%	50.00%	50.00%
		Arrival of batch1	50.00% **86.00%**	53.00%	-
		Arrival of batch2	-	50.00% **89.50%**	73.50%
		Arrival of batch3	-	-	86.00% **89.00%**

For STAGGER dataset the best model for classifying the first concept Naïve Bayes with training accuracy = 100%. We use the same model to predict batch1 and batch2. The accuracy of classification was 57.50% and 53.66% for batch1 and batch2 respectively. This results confirm the existence of drift where the current Naïve Bayes model could not classify the other concepts correctly.

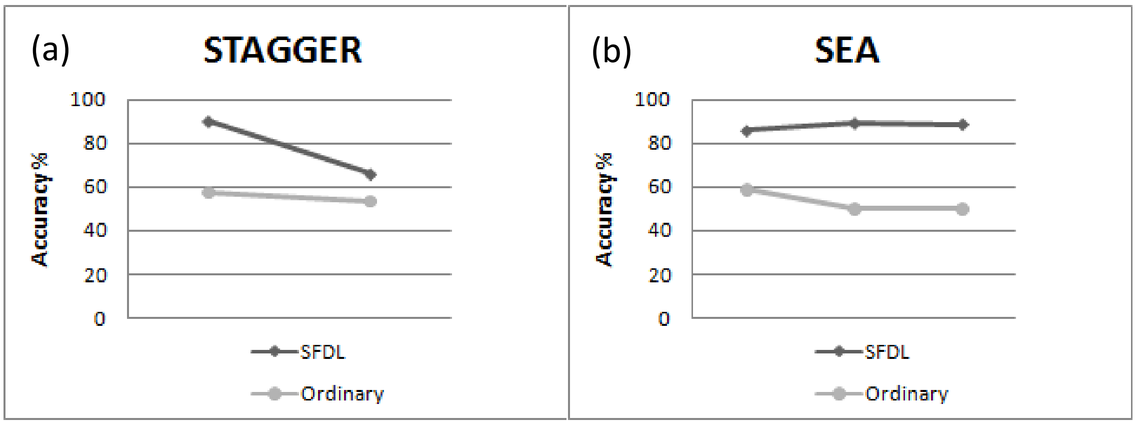


Figure 4.2: Accuracy over time for SFDL algorithm and ordinary classifier (a) STAGGER dataset (b) SEA dataset

Two accuracy observations recorded after passing batch1. First underlined observation with value = 65% represents the accuracy after reclassifying the batch using Algorithm at Figure 3.7. The second bold observation with value = 90% represent the accuracy after training set formation and model retraining. The SFDL algorithm applied with k = 2 and w_{recent} = 0. The size of batches is very small (40 instances) for this reason we chose a small number of neighborhood *k*. Also in sudden drift previous concept is not much trusted to classify the current batch. Choosing most recent historical examples to be selected at training set selection algorithm (w_{recent} > zero) is meaningless because we are dealing with sudden drift where source of drift is not related to time ordering. Although the problem of unbalancing with batch1 (10: 90) our algorithm enhance the accuracy by 32.5% (from 57.50% to 90.00%).

After the arrival of batch2 we classify it's instances using the most recent Naïve Bayes model (after retraining). Note that the accuracy decreased to 46.34%. This happened because retraining make the model adapted according to data in batch1 which is different from batch2. Also the problem of class imbalance at batch1 makes the updated model biased toward dominate label "2". This problem also affected the reclassification step where most selected instances belong to class"2", thus the accuracy decreased more to 43.90%. With perfect parameters setting (of k and w_{recent}) and role of certainty rule, the accuracy increased to 65.85% after applying SFDL algorithm.

For SEA dataset the most accurate classifier for classifying the first concept was Decision Tree with accuracy = 100%. The same model was used to classify the three incoming concepts. The accuracy of classification was 59.00%, 50.00%, 50.00% for batch1, batch2 and batch3 respectively. The classification accuracy of incoming concepts decreased compare to initial concept classification accuracy. Table 4.3 presents the accuracy after batch reclassification and model retraining after passing the three batches. SFDL algorithm applied with k = 10 and w_{recent}= 0 because we are dealing with sudden and medium-sized dataset. After model retraining, SFDL algorithm increases the accuracy of classification with at least 27%. Unlike the first two batches, reclassification accuracy of batch3 is higher than classification accuracy by initial model. This happened because of the concept sequencing, where θ graded from 7 to 9. Thus the classifier gains more knowledge after passing the two previous batches.

Figure 4.2 presents the curves of accuracy over time using SFDL algorithm and ordinary classifier for STAGGER and SEA datasets. It is notable that SFDL algorithm achieves better performance than the ordinary classifiers.

4.3.2 Gradual Drift Experiments (Electricity and Credit datasets)

Table 4.4 illustrates experimental results for both Electricity and Credit datasets. Apart from the other two datasets mentioned in the previous section, we partition electricity dataset into three different parts with different size where each represent different concept: the first part used to build the initial learner [May, 11 – May, 31], the second concept [June, 1 – June, 30] as batch1 and third concept [July, 1 – July, 11] as batch2. This exception in partitioning to different batch size comes due to the nature of dataset and concepts distribution (based on months).

For credit approval dataset, after sorting the instances from minimum to maximum according to age feature we chose instances [0 - 400] as training set to initial learner and divide the remaining 600 instances [400 - 900] into six consequent batches each with 100 instances.

Electricity and credit datasets are real world datasets with gradual drift. For both datasets we use Multilayer Perceptron Neural Network (MLP-NN) as training model.

Because we are dealing with time-related gradual drift, we take time similarity into consideration ($w_{recent} > 0$). For example, to predict electricity price or credit card approval, the most recent examples are more reliable to be used to classify new incoming instances than old historical data. The values of w_{recent} and *k* were chosen according to size of dataset and incoming batches.

From the results in Table 4.4, it is notable that average error in classifying incoming batches by ordinary classifier in gradual drift experiments is mostly less than it in gradual drift experiments. The reason is the two datasets (most real word datasets) include little concept drift.

Figure 4.3 presents the curves of accuracy over time using SFDL algorithm and ordinary classifier for Electricity and credit approval datasets. The figure shows that our algorithm achieves higher classification accuracy in comparison to ordinary classifier for both datasets.

Table 4.4: Results of Electricity and Credit datasets.
Accuracy after the batch reclassification (Figure 3.7) is underlined.
Accuracy after training set formation and model retraining is in bold.

			batch1	batch2				
Elec	**MLP-NN Acc=89.56% *k*=60 w_{recent} = 200**	Ordinary	69.44%	69.26%				
		Arrival of batch1	72.50% **87.43%**	66.66%				
		Arrival of batch2	-	67.17% **69.27%**				
Credit	**MLP-NN Acc =100% k=13 w_{recent} =100**		batch1	batch2	batch3	batch4	batch5	batch6
		Ordinary	74.00%	70.00%	65.00%	55.00%	72.00%	21.00%
		Arrival of batch1	64.00% **83.00%**	72.00%	-	-	-	-
		Arrival of batch2	-	64.00% **76.00%**	69.00%	-	-	-
		Arrival of batch3	-	-	69.00% **79.00%**	61.00%	-	-
		Arrival of batch4	-	-	-	66.00% **76.00%**	72.00%	-
		Arrival of batch5	-	-	-	-	73.00% **79.00%**	73.00%
		Arrival of batch6	-	-	-	-	-	75.00% **77.00%**

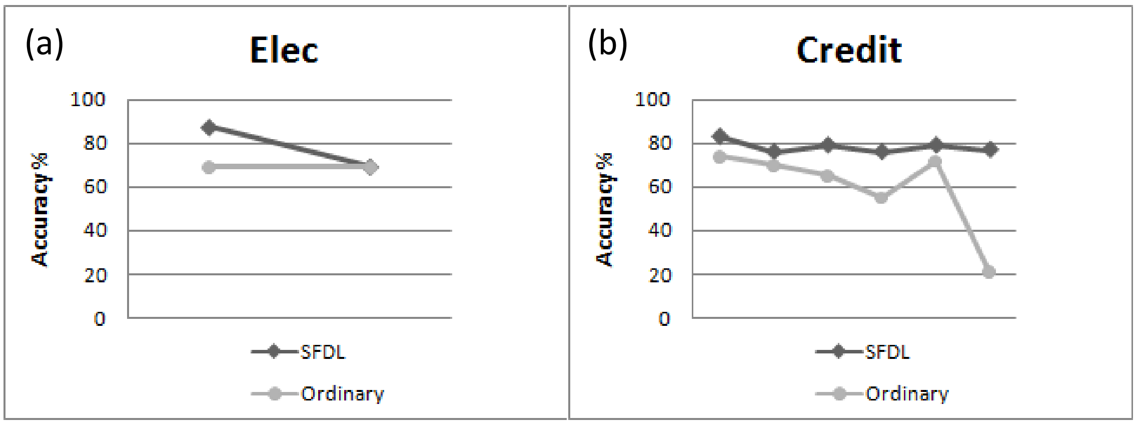


Figure 4.3: Accuracy over time for SFDL algorithm and ordinary classifier (a) Electricity dataset (b) Credit approval dataset

4.3.3 Incremental Drift Experiments (Chess dataset)

Incremental drift is a sequence of small sudden drifts. For this reason it's very difficult to predict and learn. The main difference between it and sudden drift is that incremental drift is related to time where sudden drift is not. Chess dataset is incremental real dataset. We partitioned chess dataset into four parts each represent different concept: the first concept includes playing records in a period [2007/12/07 to 2008/12/12] used to build the initial learner, the second concept in a period [2008/12/13 to 2009/03/24] used as first batch (batch1), third concept [2009/03/25 to 2009/06/30] as batch2 and fourth concept [2009/07/01 to 2010/03/09] as batch3. The results of chess experiments are presented in Table 4.5. The best model for predicting first concept was Rule-Based Classifier with accuracy = 92.06%. We choose a very small neighborhood *k* and w_{recent} value because it is suitable to the nature of data and change speed. From the table it is clear that our approach have better predictive performance than the classical ordinary classifier.

By the arrival of first and second batch, RFDL enhance the accuracy by at least 17% but it is not more than 3% for the last batch. We think the reason is the extensive sudden drifts during this period. Also in this period the user turns to play personal competitions (70% of total instances in batch3). This may add another hidden cause of drift related to player-opponent relationship.

Table 4.5: Results of Chess dataset.
Accuracy after the batch reclassification (Figure 3.7) is underlined.
Accuracy after training set formation and model retraining is in bold.

			batch1	batch2	batch3
Chess	**Rule-based Classifier Acc = 92.06% k=13 w_{recent} = 20**	Ordinary	51.00%	59.00%	64.00%
		Arrival of batch1	75.00% **74.00%**	74.00%	-
		Arrival of batch2	-	69.00% **76.00%**	67.00%
		Arrival of batch3	-	-	67.00% **67.00%**

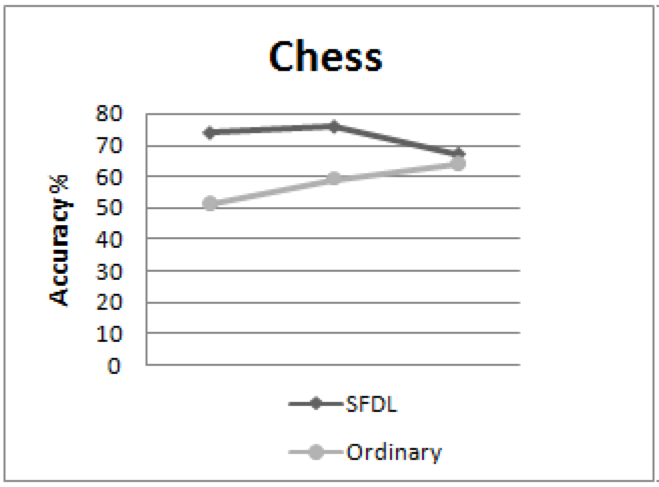


Figure 4.4: Accuracy over time for SFDL algorithm and ordinary classifier for chess dataset

Figure 4.4 presents the curves of accuracy for SFDL algorithm and ordinary classifier. SFDL shows superior accuracy over ordinary classifier.

4.3.4 Usenet Datasets Experiments

Changes in user interests over time are the main cause of concept drift in usenet dataset. It is obvious from Figure 4.1 that usenet datasets represent recurrence drift type. In fact this dataset is more much more complicated in reality due to unpredictable user interests.

We benefit from [11] to partition the data as illustrated in Figure 4.1. we use the first user interest (medicine articles) to build initial learner and other parts of interest to represent incoming batches. It is to be mentioned that batches with same interests are not identical.

Table 4.4 illustrates experimental results for usenet datasets. The best model for predicting first concept for both usenet datasets was MLP-NN with accuracy = 95.83% and 93.33% for usenet1 and usenet2 respectively. We benefit from Katakis et al. [11] experiments on Time Window methods to choose best w_{recent} value and other extensive experiments done to choose k neighborhood value.

Table 4.6: Results of Usenet datasets.
Accuracy after the batch reclassification (Figure 3.7) is underlined.
Accuracy after training set formation and model retraining is in bold.

			batch1	batch2	batch3	batch4
Usenet1	**MLP-NN Acc = 95.83% k=30 w_{recent} = 100**	Ordinary	24.33%	82.33%	20.67%	88.33%
		Arrival of batch1	52.66% **59.00%**	54.00%	-	-
		Arrival of batch2	-	69.33% **88.33%**	60.00%	-
		Arrival of batch3	-	-	54.33% **71.33%**	55.33%
		Arrival of batch4	-	-	-	66.33% **90.00%**
			batch1	batch2	batch3	batch4
Usenet2	**MLP-NN Acc = 93.33% k= 30 w_{recent} = 100**	Ordinary	60.67%	57.00%	60.67%	80.00%
		Arrival of batch1	49.00% **81.00%**	44.67%	-	-
		Arrival of batch2	-	42.00% **70.67%**	72.00%	-
		Arrival of batch3	-	-	64.00% **77.00%**	75.00%
		Arrival of batch4	-	-	-	72.00% **84.00%**

It is obvious that initial model for both usenet datasets can predict batches with medicine articles accurately than other batches.

The results show the ability of SFDL algorithm to switch between concepts (as user interests change) and how gradual forgetting by center combination (equation 3.4) improve classification accuracy. Figure 4.5 also shows the advantages of SFDL algorithm over ordinary classifier.

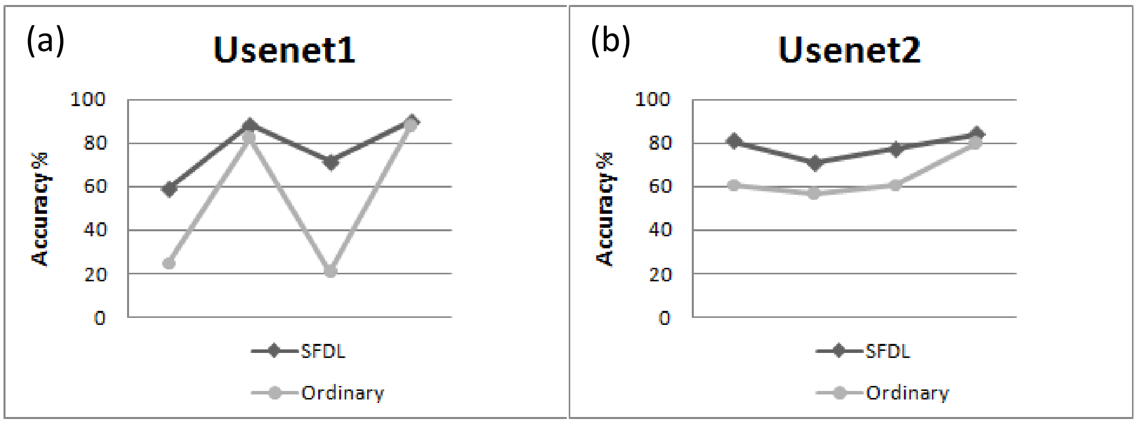


Figure 4.5: Accuracy over time for SFDL algorithm and ordinary classifier (a) Usenet1 dataset (b) Usenet2 dataset

Finally, Table 4.7 presents a comparative study between RFDL algorithm and four other stream classification methods for usenet datasets. These methods are considering concept drift in their approaches. A description for each of the four methods is provided at Chapter 2.

It is notable that in general using all the four methods plus the SFDL algorithm, the average classification accuracies for usenet2 dataset are higher than of that of usenet1. This is because usenet1 includes more complicated drift where the same batch includes another drift which means that the user switch between two different interests. It is clear that SFDL approach outperforms all other methods and the approach with Time Window (N=100) is the worst.

Table 4.7: Average accuracy of the four methods in the Usenet datasets

Method	**Usenet1**	**Usenet2**
SFDL Algorithm	81.00%	81.20%
Simple Incremental [11]	59.00%	73.00%
TimeWindow (N=100) [38]	56.00%	60.00%
TimeWindow (N=150) [38]	59.00%	62.00%
TimeWindow (N=300) [38]	58.00%	70.00%
Weighted Examples [18]	67.00%	75.00%
CCP (batch size = 50) [11]	81.00%	80.00%

CHAPTER 5
Conclusion and Future Work

5.1 Conclusion

In this thesis, we addressed a problem of supervised learning over time when the data is changing (concept drift) and label of new instances is delayed. We introduce an adaptive training set formation algorithm called SFDL, which is based on selective training set formation.

SFDL algorithm includes three sub algorithms: instance selection algorithm that is used to select the most relevant examples to current concept in terms of time similarity and space similarity, reclassification algorithm to reclassify the new instances, which were initially classified using the available classifier and the third algorithm is training set formation algorithm which work to reform the old set according to the changes made on reclassification step.

We tested our approach using synthetic and real datasets. The datasets are chosen from various domains which might have different drift types (sudden, gradual, incremental reoccurrences) with different speed of change. Experimental evaluation confirms improvement in classification accuracy as compared to ordinary classifier for all drift types. Our approach is able to increase the classifications accuracy with 20% in average and 56% in the best cases of our experimentations and it has not been worse than the ordinary classifiers in any case

Finally, we conducted a comparative study between our proposed method and another four methods to identify recurrence drift and predict changes in user interest in news group over time. The results show the superiority of our solution over other methods in handling recurrence drift and fast respond to change.

Our proposed solution is considered as the first systematic training set formation approach that take into account delayed labeling problem.

5.2 Future Work

Future research will be directed in the following direction:

- For input setting parameters like number of neighborhood *k* and number of most recent instances w_{recent}, these parameters have been determined by application designer. It is better to automatically determine these parameters to preserve self-adaption.

- In our algorithm the training set will be increased gradually after each formation. A special sampling strategy could be developed to preserve new knowledge and remove insignificant instances.

- Extending our algorithm so it can add or remove classes. This is important where in some domains, there are classes that disappear by time and must be removed or vice versa.

- Develop a dynamic feature space formation. This is very useful when dealing with textual data, structured and unstructured documents, web content analysis changes might affect only a part of the feature space, related to the changed vocabulary.

- Build a strategy to form artificial instances from existing historical data. We expect such strategies could increase flexibility in adaptation to drifts in cases of small sample size.

- Exploring some ideas to enhance the proposed strategy to improve the results accuracy. A very high classification accuracy can be provided if we build a customized version to deal with each drift individually.

Finally, concept drift problems are heterogeneous from the application perspective. We believe that the future research on adaptivity to concept drift has prospects and demand to come closer to specializing in application groups.

REFERENCES

[1] Aha, D.; Kibler, D. and Albert, M.;"Instance-Based Learning Algorithms," Machine Learning, vol.6 no.1, pp.37-66, Jan 1991.

[2] Brzezinski, D.; "Mining data streams with concept drift ", Master thesis, Poznan University of Technology, 2010.

[3] Chao, S. and Wong, F.; "An Incremental Decision Tree Learning Methodology regarding Attributes in Medical Data Mining," Proceedings of International Conference on Machine Learning and Cybernetics, IEEE Computer Society, pp. 1694-1699, 2009.

[4] Cohen, L.; Avrahami, G.; Last, M. and Kandel, K.; "Info-fuzzy algorithms for mining dynamic data streams," Applied Soft Computing, vol.8 no.4, pp.1283-1294, September 2008.

[5] Crespo, F. and Weber, R.;" A methodology for dynamic data mining based on fuzzy clustering, Fuzzy Sets and Systems," pp.267-284. 2005.

[6] Delany, S. J.; Cunningham, P.; Tsymbal, A. and Coyle, L.; "A case-based technique for tracking concept drift in spam filtering," Knowledge-Based Systems, vol 18. 2005.

[7] Dries, A. and Ruckert, U.; "Adaptive concept drift detection," In SDM, pp. 233-244. 2009.

[8] Fan, W.; Huang, Y.; Wang, H.; and Yu, P. S. "Active mining of data streams," Proceedings of 4th SIAM ICDM. 2004.

[9] Harries, M.; "Splice-2 comparative evaluation: Electricity pricing," Technical report, The University of South Wales, 1999. URL http://www.liaad.up.pt/~jgama/ales/ales_5.html (last accessed Oct 8, 2011).

[10] Katakis I.; Tsoumakas G. and Vlahavas I.; "Tracking recurring contexts using ensemble classifiers: an application to email filtering," Knowlegde and information systems, vol.22 no.3, pp.371–391. 2010.

[11] Katakis, I.; Tsoumakas, G. and Vlahavas, I. "An Ensemble of Classifiers for Coping with Recurring Contexts in Data Streams". In Proceeding of 18th European Conference on Artificial Intelligence, Patras, Greece. 2008.

[12] Katakis, I.; Tsoumakas, G.; Banos, E,; Bassiliades, N. and Vlahavas, I.; "An adaptive personalized news dissemination system," Journal of Intelligent Information Systems. 2009.

[13] Kelly, M.; Hand, D. and Adams,N.; "The impact of changing populations on classifier performance," Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp.367-371, August 15-18, 1999.

[14] Kifer, D.; Ben-David, S.; and Gehrke, J.; "Detecting change in data streams," Proceedings of the Thirtieth Int. Conf. on VLDB, vol.30. 2004.

[15] Klinkenberg, R. and Joachims, T.; "Detecting Concept Drift with Support Vector Machine," Proceeding of 17th Int'l Conf. Machine Learning, pp. 487-494, 2000.

[16] Klinkenberg, R. and Renz, I.; "Adaptive Information Filtering: Learning in the Presence of Concept Drifts," Proceeding of AAAI Workshop Learning for Text Categorization, pp. 33-40, 1998.

[17] Klinkenberg, R. and Ruping,S.; "Concept drift and the importance of examples," in: Text Mining – Theoretical Aspects and Applications, Franke, J.; Nakhaeizadeh and Renz, I.; Springer, Berlin, Germany, 2003.

[18] Klinkenberg, R.; "Learning drifting concepts: Example selection vs. example Weighting," Intelligent Data Analysis, vol.8 no.3.pp.281–300. 2004.

[19] Lanquillon, C.; "Information filtering in changing domains," In Workshop on Machine Learning for Information Filtering, IJCAI'99, pp.41–48. 1999.

[20] Lindstrom, P.; Delany, S. and Namee, B.; "Handling Concept Drift in Text Data Streams Constrained by High Labelling Cost," Proceedings of the 23rd Florida Artificial Intelligence Research Society Conference (FLAIRS). 2010.

[21] Ludmila I.; Kuncheva, J. and Salvador S.; "Nearest Neighbour Classifiers for Streaming Data with Delayed Labelling," Eighth IEEE International Conference on Data Mining icdm, pp.869-874. 2008.

[22] Maloof M. and Michalski R.; "Selecting examples for partial memory learning," Machine Learning. pp. 27-52, 2000.

[23] Menezes, B.; Parma, G; Seleme, S. and Nied, A.; "On-line neural training algorithm with sliding mode control and adaptive learning rate," Neurocomputing, vol.70, pp. 2687-2691. 2007.

[24] Mjolsness, E. and DeCoste, D.;" Machine Learning for Science: State of the Art and Future Prospects," Science, pp.2051–2055. 2001.

[25] Moore, A.; "Fast, Robust Adaptive Control by Learning only Forward Models," Advances in Neural Information Processing Systems 4, pp.571-578, 1992.

[26] Narasimhamurthy, A.; and Kuncheva, L.; "A framework for generating data to simulate changing environments". In Proceeding of the 25th IASTED int. Multi-Conference, Artificial Intelligence and Applications (AIAP'07), pp. 384–389. ACTA Press, 2007.

[27] Nishida, K. and Yamauchi, K.; "Learning, detecting, understanding, and predicting concept changes," International Joint Conference on Neural Networks ijcnn, pp.2280-2287, 2009.

[28] Nishida, K.; "Learning and Detecting Concept Drift,". PhD thesis, Hokkaido University, Japan, 2008.

[29] Pratt, K. and Tschapek, G.; "Visualizing concept drift," Conference on Knowledge Discovery in Data, Proceedings of the Ninth ACM SIGKDD International Conference on Kowledge Discovery and Data Mining, pp. 412-419, 2003.

[30] Schilling, M.; "Multivariate two-sample tests based on nearest neighbors, " Journal of the American Statistical Association, vol. 81 no. 395, pp. 799–806, 1986.

[31] Schlimmer, J. and Granger, R.; "Incremental learning from noisy data," Machine Learning, vol.1 no.3, pp.317–354, 1986.

[32] Street, N. and Kim, Y.; "A streaming ensemble algorithm (SEA) for large-scale classification," In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, 2001. URL http://www.liaad.up.pt/~kdus/kdus_5.html (last accessed Oct 8, 2011).

[33] Su, B.; Shen, Y. and Xu, W.; "Modeling concept drift from the perspective of classifiers," In Proceeding of the conference on Cybernetics and Intelligent Systems, 2008 IEEE, pp.1055–1060, 2008.

[34] Tian, X.; Sun, Q.; Huang, X. and Ma, Y.; "Dynamic Online Traffic Classification Using Data Stream Mining," MultiMedia and Information Technology, 2008. MMIT '08. International Conference on, pp.104-107. 30-31 Dec. 2008.

[35] Tsymbal, A.; Pechenizkiy, M.; Cunningham, P. and Puuronen, S.; "Dynamic integration of classifiers for handling concept drift," Information Fusion, vol. 9 no. 1, pp.56–68, 2008.

[36] Tsymbal, A.; "The problem of concept drift: Definitions and related work. Technical Report," Department of Computer Science, Trinity College: Dublin, Ireland, 2004.

[37] Wang, H.; Fan, W.; Yu, P. and Han. J.; "Mining concept-drifting data streams using ensemble classifiers," Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03). ACM, New York, NY, USA, pp.226-235, 2003.

[38] Widmer, G. and Kubat, M.; "Learning in the presence of concept drift and hidden contexts," Machine Learning, vol. 23 no. 1, pp. 69-101, April 1996.

[39] Widyantoro, D.; "Concept Drift Learning and its Application to Adaptive Information Filtering". Ph.D. Dissertation. Texas A&M University. 2003.

[40] Yang, Q. and Wu, X.; "10 challenging problems in data mining research," International Journal of Information Technology & Decision Making, vol. 5, no. 4, pp. 597-604, 2006.

[41] Yang, X.; Yuan, B. and Liu, W.; "Dynamic Weighting Ensembles for Incremental Learning," Pattern Recognition, 2009. CCPR 2009. Chinese Conference on, pp.1-5, 4-6 Nov. 2009.

[42] Zhang, Z. and Zhou, J.; "Transfer estimation of evolving class priors in data stream classification," Pattern Recognition, vol. 43 no. pp. 3151-3161, 2010.

[43] Zimek, A.; Kriegel, H. ; Borgwardt, K.; Kroger, P.; Pryakhin, A. and Schubert, M.; "Future trends in data mining," Data Mining and Knowledge Discovery, vol. 15, no. 1, pp. 87–97, Aug. 2007.

[44] Žliobaitė, I. and Pechenizkiy, M.; "Reference Framework for Handling Concept Drift: An Application Perspective," Technical report, Eindhoven University of Technology, 2010.

[45] Žliobaitė, I.; "Adaptive training set formation". PhD thesis, Vilnius University, 2010.

[46] Žliobaitė, I.; "Change with Delayed Labeling: When is it Detectable?," In IEEE International Conference Data Mining Workshops (ICDMW), pp.843-850, 13-13 Dec. 2010.

[47] Žliobaitė, I.; "Combining time and space similarity for small size learning under concept drift,". In: Lecture Notes in Computer Science, Rauch, J.; Ras, Z.; Berka, P. and Elomaa, T.; ISMIS-Springer, vol. 5722, pp. 412–421, 2009.

[48] Žliobaitė, I.; "Learning under concept drift: an overview," Technical report, Vilnius University, Faculty of Mathematics and Informatics, 2009.

[49] Chess.com dataset, http://sites.google.com/site/zliobaite/resources-1 (last accessed Oct 8, 2011).

[50] http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html (last accessed Oct 8, 2011).

[51] http://statsoft.com/textbook/cluster-analysis/ (last accessed Oct 28, 2011).

[52] UCI Machine Learning Repository: Data Sets, http://archive.ics.uci.edu/ml/datasets.html (last accessed Oct 8, 2011).