technische universität
dortmund

Multivariate Statistical Process Control using Dynamic Ensemble Methods

Dhouha Mejri

Doctoral Thesis for the Degree of

”Doktor der Naturwissenschaften”
in the Department of Statistics
TU Dortmund University

Dortmund, December 2015

Acknowledgement

Foremost, I would like to express my honest gratitude to my supervisors Professor Claus Weihs and Professor Mohamed Limam who have supported me with their guidance, encouragement and advice. Without them, this thesis would not have been completed. One could not wish for better supervisors like them. I'm greatly thankful to the Deutscher Akademisher Austausch Dienst (DAAD) for their financial support and for giving me the opportunity to finish my thesis in such high quality environment. I would like also to thank the University of Dortmund and the department of Statistics for providing me with a scholarship funds. I'm also thankful to the Institut Suprieur de Gestion (ISG) de Tunis and the University of Tunis for their support and cooperation. I'm especially grateful for Mrs Sabine Bell for her big support and admirable encouragement during my stay at the university. I also thank my colleague Nadja Bauer for her consistent advice during the beginning of my studies at the University of Dortmund. Finally, I would like to express my heartfelt gratitude to my family. A special thank to my husband and my daughter for their love, help and concern during these years of thesis. I'm enormously thankful and indebted to my parents for being my constant source of love and strength during all my life.

Abstract

One important challenge with some applications such as credit card fraud detection, intrusion detection and network traffic monitoring is that data arrive in streams over time and leads to changes in concepts which are known in data mining as concept drift. Thus, models analyzing such data become obsolete and efficient learning should be able to identify these changes and quickly update the system to them. The objective of this dissertation is to investigate the effectiveness of ensemble methods and Statistical Process Control (SPC) techniques in detecting changes in processes in order to improve the robustness of tracking concept drift and coping with the dynamics of online data stream processes. For reaching this objective, different heuristics were proposed. First, an improved dynamic weighted majority Winnow algorithm based on ensemble methods is proposed. Furthermore, parameters optimization based on genetic algorithm of the proposed method as well as an analysis of its robustness are investigated. Second, in order to handle the problem of concept drift while monitoring nonstationary environment using SPC tools, a time adjusting control chart based on a recursive adaptive formulas of the charting statistics is proposed. Results show that the updating charts cope much better with the nonstationarity of the environment. Also, two new heuristics are proposed based on both ensemble methods and adaptive control charts. The first is an offline learning chart model while the second is an online batch learning algorithm. Results show that quick adaptation of the system and accurate shift point identification are achieved when using both heuristics together. Also, the new adaptive ensemble charts have better performance in learning concept drifts along with a good suitability to nonlinearity and noise issues.

Contents

List of Tables

List of Figures

Abbreviations

ACE	Adaptive Classifiers Ensemble
ADWIN	ADaptive WINdoing
ANOVA	ANalysis Of VAriance
AR	Autoregressive model
AWE	Accuracy Weighted Ensemble
BDDE	Batch Drift Detection
BWE	Batch Weighted Ensemble
CART	Classification and Regression Trees
CC	Control Charts
CUSUM	Cumulative Sum Control Chart
DWM	Dynamic Weighted Majority
DWMC	Dynamic Weighted Majority Control
$DWM-WIN$	Dynamic Weighted Majority Winnow
EWMA	Exponentially Weighted Moving Average
GA	Genetic Algorithm
HIG	Hybrid Interval Genetic algorithm
IFCS	Incremental Fuzzy Classification Systems
kknn	Weighted k-nearest Neighbors
LCCS	Learning Control Chart System
LCS	Learning Classifier System
LCL	Lower Control Limit
MA	Moving Average
MEWMA	Multivariate Exponentially Weighted Moving Average
MLP	Multilayer Perceptron
MOA	Massive Online Analysis
naive.Bayes	Naive Bayes
PM	Partial Memory

PSO	Particle Swarm Optimization
$REDLL$	REcurring concept Drift and Limited Labeled data
$rpart$	Recursive Partitioning And Regression Tree
$SD-EWMA$	Shift Detection Exponentially Weighted Moving Average
$SFEWMA-X$	Single Featured EWMA-XBAR
$SNCS$	Supervised Neural Constructivist System
SPC	Statistical Process Control
$TACL$	Time Adjusting Control Limits chart
$TS-TACL$	Two Stage Time Adjusting Control Limits chart
$TSSD-EWMA$	Two Stage Shift Detection Exponentially Weighted Moving Average
UCL	Upper Control Limit
VSI	Variable Sampling Interval
VSR	Variable Sampling Rate
XCS	X Classifier System

Chapter 1

Introduction

One major task in machine learning is to construct a model which copes with changing environments. The process of building models that can adapt to the environment based on new arrived data is called adaptive learning. Dynamic modeling algorithms, which learn from online data streams with concept drift, have recently received a lot of attention due to their importance in handling real world problems. Most often, learning from continuously arriving data, where concepts change over time, presents new challenges to traditional machine learning methods which are not designed to handle data where concepts are time-changing.

In recent years, credit card fraud detection, network traffic monitoring, industrial process control and intrusion detection have contributed to different new applications of continuously arriving data known as data streams. In general, existing solutions construct stream data mining under the assumption that data are stationary and hence classification systems are used with balanced class distribution. Consequently, the problem of learning in a nonstationary environment is of great importance and it is known as "concept drift". Specifically, the stationarity assumption contradicts the concept drift reality which requires prior knowledge of when and where concepts may change. That is why old representations of real-world data streams become inadequate. Then, constructing a learner who is able to learn incremental data from a nonstationary environment becomes essential. The problem of concept drift was handled by classification methods such as ensemble methods as well as techniques from Statistical Process Control (SPC). Control Charts (CCs) are the most important tools used for monitoring and control in SPC domain. These tools are used to distinguish between process variations resulting from common causes and those from special causes. CCs allow maintaining the process stability and detecting changes of the process. However, although many methods were proposed to improve the change detection ability, there are still many problems. One of the issues in SPC is that methods are constructed under several assumptions and do not allow the distinction between concept drift situations and outliers due to the nonstationarity of the process. Also, another problem in SPC is that each CC has specific ad-

vantages in detecting small, moderate or large shift. That's why it is usually difficult to choose the best CC to apply in data streams processes. To overcome these shortcomings, integration of data mining algorithm into SPC domain could improve drift detection ability. Furthermore, the use of adaptive CC with time changing modeling allows a self adjustment system which is updated after each shift detection and copes very well with the changing environments. Moreover, combining CC decisions based on classification algorithms provides better process monitoring and improves CC ability to detect all shift ranges. One of the most intelligent heuristics in online classification method for concept drift is Dynamic Weighted Majority (DWM) of **[Kolter and Maloof, 2007]** and **[Mejri et al., 2013]**.

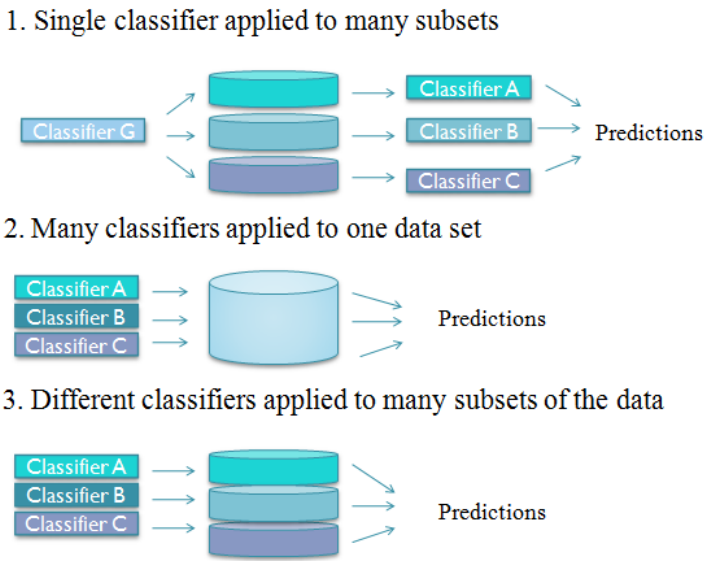

Figure 1.1: Illustration of different methods to generate different classifiers.

1.1 Dynamic ensemble methods

A group of humans can usually make better decisions than individuals, mainly when each element of the group contributes with his own adjudication. This reasoning is identical in machine learning. Better performance is achieved with an ensemble of learning models by combining the decisions of different learners.

[Zhou, 2012] defined ensemble methods as the task of training multiple learners to solve the same problem. Ensemble methods can be classified in two groups: *homogeneous ensembles* that are based on learners of the same type and *heterogeneous ensembles* which use different learning

algorithms. Base learners generated from the training data can be Weighted k-Nearest Neighbors (kknn), naive Bayes (naiveBayes), decision tree (rpart) or any other classifier.

The construction of different classifiers' model can be obtained through 3 ways: (a) to train a single classifier on different subsets of the dataset, (b) to train different classifiers on the same dataset or (c) to train different classifiers on different subsets of the dataset. An illustration of these different ways to construct an ensemble algorithm is illustrated in Figure (1.1).

1.2 Adaptive statistical process CCs

[Lavretsky and Wise, 2013] defined Adaptive charts as a controller that performs the online estimation of uncertain processes by learning and / or remembering some specific patterns based on the prior knowledge obtained from a feedback loop of previous data. Figure (1.2) presents an example of an Adaptive CC called Shift-Detection EWMA chart for non stationary environment of **[Raza et al., 2013]** and **[Raza et al., 2015]**.

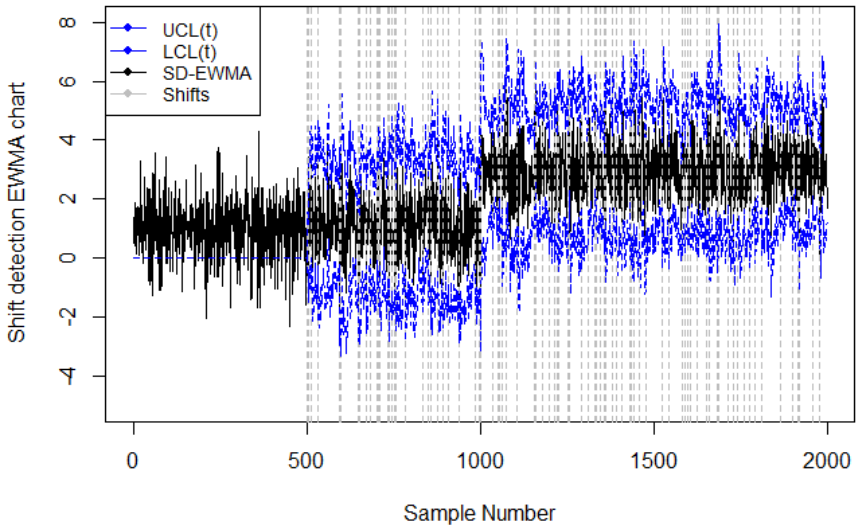

Figure 1.2: Example of a time updating CC for a covariate shift detection.

1.3 Dissertation outline

The objective of this research thesis is to improve the performance of dynamic ensemble methods in detecting concept drift under the nonstationarity assumption of the processes and to enhance the application of data mining methods into SPC. Part I is devoted to optimize the DWM algorithm by including learner's age as a new criteria and optimizing the key parameters of DWM with a Genetic Algorithm (GA).

This part will also present an assessment of the proposed enhanced method in datasets with concept drift by studying the impact of the permutation sequences of the concepts, the non linearity as well as the noise effect on the robustness of the studied ensemble method. Part II presents a proposal of two new adaptive and robust time adjusting CCs for detecting concept drift under the worst case condition assumption. An evaluation and a comparative study based on different datasets is presented and analyzed. Part III discusses two different methods for combining CCs. One is to be applied for offline data which consists of a CC learning procedure based on applying DWM at the end of the monitored process. The second proposal consists of online combined charts which dynamically treat CCs as classifiers when applying the DWM to combine the individual charts. This part involves the evaluation and the analysis of these two new methods in different non-stationary processes.

Part I

Dynamic Weighted Majority Method Optimization for Concept Drift Detection

Classification methods which learn from data streams have been recently challenged. One of the reasons of this is that real life problems change over time. As a result, the assumption that data usually follows the same distribution is often violated in real world applications. Thus, many sophisticated algorithms were designed to improve learning in non-stationary environments. **[Gama et al., 2004]** classify these methods into two categories: i) methods adapting classifiers without taking into account past history in detecting the changes occurring at a regular time interval, ii) methods that adapt classifiers just after considering all detected changes. According to **[Díaz et al., 2014]**, ensemble methods are usually applied to the first category because they are able to adapt to changes by updating learners in the ensemble, removing some classifiers and adding new ones, etc, …. However, the second strategy requires storing classifiers' history of detecting concepts according to their age and utility. The second category is often defined as a system for recurring concepts.

[Street and Kim, 2001] proposed Streaming Ensemble Algorithms (SEA), one of the first algorithms for streaming data, by building different classifiers on different consecutive chunks of the training dataset. Then, once the ensemble is constructed, new classifiers are added based on their ability to improve the ensemble accuracy while old classifiers are removed to maintain the same ensemble size. Although this method outperforms other methods using all training data in a single model, it has drawbacks. It removes old classifiers without considering their contributions. Also, despite it is good in adapting to gradual changes it is not for abrupt changes. Also, SEA is an unweighted majority voting technique.

[Wang et al., 2003] used the same training data chunks division of **[Street and Kim, 2001]** but with weighted classifier ensembles to handle streaming data with concept drift. The method called Accuracy Weighted Ensemble (AWE) gives a weight to each classifier based on its expected prediction accuracy obtained when using the example from the current test chunks. This method can adapt to gradual drift but as with SEA it has a difficulty coping with sudden concepts. Indeed, it is not able to select classifiers in a dynamic way without a decrease in the accuracy performance.

[Deckert, 2011] improved AWE algorithm by incorporating a Batch Drift Detection Method (BDDM). His method is called Batch Weighted Ensemble (BWE) and it is based on a simple linear regression model that estimates the change in the data. Thus, the method is suitable to data with gradual as well as sudden drift. Moreover, the idea of this method is to add a new classifier only when the concepts in the batches are not stable. Else, the ensemble is not modified when the concept is stable. By this way, BWE outperforms AWE in detecting abrupt change.

Although these methods present intelligent heuristics for mining data streams with concept drift using static and incremental classifiers, they are not constructed in a way that allow handling online data streams. Other methods update the ensemble of classifiers with each new instance or batch coming over time. **[Maloof and Michalski, 2004]** proposed an online method called AQ-PM based on the AQ system of **[Michalski, 1969]** with a partial memory (PM) strategy by

involving a forgetting mechanism when applying the algorithm to each example one by one as they arrive.

Another online ensemble algorithm is proposed by **[Kolter and Maloof, 2005b]** called Addexp. The method presents an online ensemble method based on adding a new *expert*, which is referred to us as *base learner* or *classifier*, to the ensemble during the online learning process and reducing the weight of bad classifiers. However, Addexp updates the ensemble of classifiers at each time step which enables it to handle recurring concepts.

[Kolter and Maloof, 2007] proposed a new method improving all the previous ones. It investigates Addexp for dynamic processes and adds many other steps. Whereas Addexp initializes a new classifier's weight as the total weight of the ensemble times some factor γ between 0 and 1, DWM algorithm sets the new weight of classifiers to 1. Then, the weight of classifiers is updated with each new instance or batch coming over time, classifiers with many incorrect predictions are removed while new classifiers are added based on their global prediction in the ensemble. **[Kolter and Maloof, 2007]** demonstrated that DWM has a competitive performance on different real and artificial datasets for concept drift. However, despite its effectiveness, DWM has some shortcomings. It penalizes classifiers that missclassify instances or batches during the learning process by decreasing their weight. However, it does not reward good classifiers with good predictions which makes the weights of classifiers fall quickly. This limitation makes it inappropriate for recurring concepts since only few memory for good classifiers is considered.

There are systems of detecting recurrent concepts under nonstationary processes. **[Nishida et al., 2005]** proposed an online learning system called Adaptive Classifiers ensemble (ACE) which is a weighting procedure based on different mechanisms to handle recurring concepts. In fact, the method used a batch classifier ensemble method, an online learner, a drift detection technique based on the predictive accuracy of each learner in the current data stream, a sliding window and a long term buffer to store classifier's prediction history and to store recent training examples respectively. Moreover, the pruning method used to adjust learner's weight during the process enables the system to hold efficient classifiers.

REccuring concept Drift and Limited Labeled data (REDLL) is another algorithm for data streams proposed by **[Li et al., 2010]** not as an ensemble method but as a method designed to cope with recurrent concept drifts with limited labeled data. It uses a semi-supervised learning model. It builds a decision tree and built a concept of clusters to label unlabeled data based on K-Means clustering. The authors prove that the method is efficient in handling recurrent concepts even in the presence of unlabeled data.

While many ensemble methods were proposed to mine data streams with concept drift and many systems were proposed to handle recurrent concepts, there are only very few works where ensemble methods deal with recurrent concepts since the classifiers' updating weight mechanism in ensemble methods ignores good classifiers' contribution during the learning process.

In this thesis, we focus on improving DWM as one of the most accurate ensemble methods coping with concept drift. It adjusts expert's weights according to the environment. More precisely, it treats concept drift as a prediction issue where the aim is to detect when it happens and then updates the algorithm accordingly. It learns incrementally with no dependence on a specific classifier model. An ensemble of experts is added and others are deleted according to global and local prediction in order to cope with nonstationary environment.

While DWM outperforms single classifiers and other ensemble methods for concept drift, it does not take into account expert's good history and their contributions during the learning process. Indeed, it doesn't consider the age of the expert in the ensemble when removing some bad classifiers and adding new ones. Another limitation of DWM is that it is based on two key parameters: one by θ denoting the threshold for removing experts from the ensemble and the second β a parameter to reduce the weight of bad classifiers after making a wrong prediction. These parameters are fixed by the users without any optimization technique. However, values of these parameters greatly impact the algorithm's performance as well as the number of classifiers in the ensemble and their optimal choice may change while applying DWM in different data sets.

Therefore, to overcome these issues, we propose to reward the contribution of successful classifiers in the learning process by increasing their weight during the classifiers' weight update. Good classifiers contain the old concept history, hence encouraging successful classifiers to remain in the ensemble, instead of quickly removing them and add new ones without any history, is one efficient way to handle concept drift as well as sudden and gradual shifts. Accordingly, we consider the age of the learner in the ensemble as a new criteria for best classifier selection which makes the DWM suitable for the treatment of any type of concept drift. Furthermore, we propose an efficient optimization technique based on a Genetic Algorithm (GA) to automate the choice of DWM parameters.

Finally a comparative study is conducted to evaluate the effectiveness of our proposals in improving the DWM algorithm based on a benchmark dataset where we highlight the effect of noise and nonlinearity in the concept drift detection.

Part I of the thesis is outlined as follows. Chapter 2 presents an overview of the DWM algorithm, explains the proposed DWM-WIN algorithm including the two new criteria and presents the experimental results of the optimized algorithm. Chapter 3 introduces the problem of parameter optimization and details the proposed method of integrating GA in DWM algorithm and evaluates the performance of the optimized DWM-WIN-GA. Chapter 4 evaluates the proposed method using a dataset with concept drift and analyses the different circumstances of the data construction.

Chapter 2

DWM-WIN: An Improved Version of Dynamic Weighted Majority Algorithm

In real world data applications, handling data streams arriving over time is a challenging domain because of the volume of incoming data and the particularity of the dynamics that may happen to its underlying structure. Thus, many machine learning methods such as classification and clustering are considered as the most challenging techniques for this context. Ensemble methods have been effectively used for mining data stream processes. This chapter proposes an extension of DWM ensemble method to improve the classification performance when a new batch of data arrives over time and provides an overview of the basics about data stream and concept drift. It is organized as follows: Section 1 presents definitions and the issue of learning from data stream. In Section 2, the concept drift is defined, Section 3 presents an overview of DWM. In Section 4, the proposed enhancement to DWM is detailed. Section 5 summarizes the different steps of the new method. Finally, Section 6 concludes this chapter.

2.1 Data stream

Data stream classification is a major challenge in data mining. There are two key problems when handling such data. First, it is impractical to store and use all historical data for training, since it would require infinite storage and running time. Second, there could be a changing in the concept to be learned called "concept drift". Solutions to these two problems are related. In fact, if a change occurs in the data, we need to refine our hypothesis to accommodate the new concept. Formally, a data stream is defined as an ordered sequence of data points:

$$D_{stream} = (t_1, t_2, t_3, \ldots, t_n, \ldots, t_{n+m}, \ldots), \tag{2.1}$$

where the indexes indicate the sequence of time in which the data is arriving over time in form

of instances or batches and t_i is the i^{th} arrived data. Therefore, the subsequence between the times t_i and t_j is called window and is denoted as follows:

$$W[i,j] = (t_i, t_{i+1}, \ldots, t_j), \tag{2.2}$$

where $i \leq j$. There are different types of window modeling. In the following we present the most well known window models.

Landmark window: This method consists of choosing all data points starting from a fixed point in time. In a more general case, this method takes the entire data stream as a basic for estimation. However, in real time processes, recent time points are more interesting to be considered than the whole data stream. For this case, other window modeling methods were proposed.

Sliding window: This method only considers the recent data of the stream. Therefore, the interest is in the window $W[t-m+1,t]$. Correspondingly, when the data arrive over time, the size of the window will be the same but it will change to the current window.

Damped window model: This is a decay rate based method. In fact, for each new data arriving over time, more weight is assigned and the previously arrived data weight is updated by multiplying it with the new rate. Therefore, more weight is given to the recently arrived data.

A data stream presents a very interesting real domain which offers the opportunity to develop machine learning techniques for high dimensions in both number of examples and variables. In spite of that, learning a data stream in a non-stationary environment presents a difficult task since many issues can be discussed. In the following, we mention some relevant issues of learning from data streams.

2.1.1 Issues of learning from data stream

Incremental and decremental issues: The ability to continuously adapt the decision model when a new information is available is an important task. Sliding window models require forgetting the old data. Correspondingly, methods for forgetting data are required and this is a challenge.

High speed nature of data: One of the most consistent properties of data streams is the high rate of arriving data. Furthermore, the speed of the classification method has to be higher than the speed of the data arriving over time.

Necessity of unlimited memory: In classification, we need methods which store the data in memory to construct one model. Therefore, challenges of memory problems are very interesting and several methods like sampling and aggregation were proposed. In fact, handling unbounded memory is one motivation for developing specific updating algorithms based on data streams.

Feature selection and pre-processing: In machine learning and data mining, finding the appropriate features and detecting the noise are necessary tasks. When data is streaming over time, the way we deal with these tasks changes accordingly and is bounded by a time limit. For example,

one feature which is considered as relevant in the beginning of the process may become irrelevant after a window of time. However, in static data if a feature is considered as irrelevant, it is ignored during all the process. Accordingly, the dynamic criteria of the data stream has to be considered as a big issue because the data has to be monitored over time. One proposed solution to this issue is "fractal dimension" proposed by **[Daniel and Ping, 2000]**.

Changing concepts: The main difficulty to mine incremental data happens when the distribution changes over time causing the phenomenon of concept drift (**[Schlimmer and Granger, 1986]**, **[Widmer and Kubat., 1996]**). The main issue is to detect these changes denoted as concept drift during the learning process and to adapt the classification method to these changes. One interesting challenge to this issue is to propose a dynamic classification method that self adjusts the ensemble of classifiers in the ensemble after each concept drift detection. Another proposal is to monitor classification methods and detect these changes in the process as soon as possible.

2.1.2 Types of dataset shift

Covariate shift: It occurs when the input distribution $P(X)$ changes between training and testing, $P_{training}(X) \neq P_{testing}(X)$, whereas the conditional probability in training and testing is not affected: $P_{training}(y/X) = P_{testing}(y/X)$.

Prior probability shift: This type of shift occurs when there is a change in the class variable y: $P_{training}(y) \neq P_{testing}(y)$.

Concept drift: Concept shift represents the hardest challenge among the different types of shift in datasets. It corresponds to the case when $P_{training}(y/X) \neq P_{testing}(y/X)$ whereas $P(X)$ remains the same in testing and training. The problem of concept drift is found in many real world applications such as traffic incident detection systems **[Hauskrecht, 2010]**, sensor network **[Nikovski and Jain, 2009]** and intrusion detection. In this thesis, our main focus is on concept drift, hence more details about concept drift are given in the next section.

2.2 Concept drift

Officially, the word "concept" denoted the distribution of a joint probability $P(X,y)$ in a certain point of time, where the input variables are designed by X and the class label by y. **[Tsymbal, 2004]** pointed out that a concept drift can be real or virtual. In real concept drift, the posterior probability $P(y = y_i/x)$ is affected which implies that the target concept to be learned of identical values of input variables changes, where $y_i \in Y_c$ where c is the ensemble of several class labels 1 $\leq i \leq c$.

To model concept drift problem, let $Z = (X,y)$ be a feature vector, where $X \in R^p$, and a label

$y \in [-1,1]$ representing its classification. The data arrive over time in batches. Without loss of generality, these batches are of equal size, each including m examples from the data as follows:

$$Z_{(1,1)},\ldots,Z_{(1,m)},Z_{(2,1)}\ldots,Z_{(2,m)},\ldots Z_{(t,1)}\ldots,Z_{(t,m)},Z_{(t+1,1)},\ldots,Z_{(t+1,m)}, \tag{2.3}$$

where $Z_{(i,j)}$ represents the j^{th} example of batch i. For each batch, $Z_{(i,j)}$ is independently and identically distributed with respect to a distribution $Pr_{i(X,y)}$. The latter and $Pr_{i+1(X,y)}$ might differ depending on the amount and the type of concept drift. In this context, the aim of a classifier A is to successively predict labels of the new batch and to minimize the cumulative number of prediction errors. In fact, a subset of the training example from batch 1 to t is used to predict a new batch $t+1$.

According to **[Gao et al., 2007]** one of the main issues in mining concept drifting data streams is to choose the appropriate training instances to learn evolving concepts. Consequently, one possible solution is to use an incremental learning with ensemble method technique by finding the best combination of learners each time a concept drift is detected.

Therefore, the basic properties of concept drift are that they are unexpected and unpredictable. Let $X \in R^p$ be the input features. We assume that a target variable $y \in R^p$ used in a classification task has to be predicted. In the training step, X and y which are representative of the data are assumed to be known. However, when a new instance or batch X arrives, y becomes not known during the time of the prediction. In Bayesian decision theory **[Duda et al., 2000]**, the prior probability of the classes is denoted $p(y)$ whereas $p(X/y)$ presents the probability density functions for the classes for $y = 1,\ldots, c$ where c is the number of classes. The posterior probability of the decision in a classification task is defined as:

$$p(y/X) = \frac{p(y)p(X/y)}{p(X)}. \tag{2.4}$$

In real world classification domain, the data arrive continuously in form of streaming flows and the underlying distribution changes accordingly. Hence, the concept drift between $[t_0,t_1]$ is such that:

$$p_{t_0}(X,y) \neq p_{t_1}(X,y), \tag{2.5}$$

where p_{t_0} is the joint distribution at time t_0 between the variable X and the target variable y. A concept drift occurs if one or more of these 2 components change and affect the prediction. Our interest is to know the consequence of each change on the distribution components. For this we distinguish between 3 possible situations: a change in $p(y)$, a change in the probability $p(X/y)$, and as a consequence, a change in the posterior $P(y/X)$.

Concept drift can be categorized in different ways. Sudden concept drift which happens when

the configuration patterns of the data sources changes over time. Let us suppose that we have only 2 possible sources over time that we denote by C_I and C_{II}. Sudden drift occurs when at time t_0, the source C_I is suddenly replaced by C_{II}. For example, while a man is reading a book sudden interest in the news about an interesting decrease in the price of petrol is defined as sudden drift.

The second type of drift is the gradual concept drift. It occurs when two sources C_I and C_{II} are mixed over time. However, the probability of using C_I decreases whereas the probability of using C_{II} increases over time. When the sampling begins, an observation from C_{II} could be considered as random noise. The incremental learning happens when the user is interested in a long term period with very small differences between the sources, it is also called "Step wise drift". Another type of structural drift is reoccurring context. We refer to this type of drift when an already used concept reoccurs after a certain period of time. This type of concept does not include periodic or seasonality events.

Some requirements for classification methods in non-stationary environments are: (a) to learn the drift occurred in the process in order to detect it as soon as possible, (b) to distinguish between drifting concepts and changes due to the non-stationarity of the process, (c) to adapt the process after a concept drift.

Different learning algorithms were proposed in the literature to handle concept drift problems. In the next section we discuss one of the most widely used algorithms for tracking concept drift.

2.3 An ensemble method for concept drift: DWM

DWM is an ensemble method for concept drift proposed by **[Kolter and Maloof, 2007]**. Initially, the algorithm requires a set of experts, all with a weight of 1. Hence, inputs of the algorithm are a pool constituted by a number of experts and n training examples containing feature vectors, a class label and several parameters which will be discussed later. First of all, the data is subdivided into K subsets (suppose $K = 5$), let each subset presents one batch. Then, the first batch presents the basic data to learn before the arrival of the first stream of data and the four others present the stream received over time in an incremental way. Secondly, for each batch a sampling with replacement is applied, inspired from Bagging methods of **[Breiman, 1996]**, each sub-sample leads to an expert prediction which is different from others. The technique used to create different classifiers is to apply the same classifier to different subsets of the data in order to obtain several learners.

Many methods are used to create different classifiers. As an example, different learners can be used such as decision tree, neural network, naive Bayes, etc. Another possibility to built different classifiers is to vary the parameters between classifiers, like modifying the split-min parameter in different decision trees or using different initialized weights in neural networks, or random forest. For DWM algorithm, we initially create different classifiers by applying one classifier to different subsets of the data. Other techniques are also applied in the next chapters.

Consequently, the algorithm trains a number of experts in the first batch, then after the arrival of a new stream, it memorizes first classifier's prediction on the algorithm and trains the ensemble

with the newly added expert, if it exists, on the new batch. Hence, classifiers in the pool of the second batch contain information acquired from the first one. Accordingly, this allows the algorithm to be incremental without need of the history of previous data **[Kolter and Maloof, 2005b]**. It is necessary to note that this technique gives more importance to new examples and it remembers those held in memory by using experts containing past information stored during the learning process. This step is essentially based on maintaining old classifiers in memory and training a new pool on the following batch, which makes the algorithm incremental.

2.3.1 Adjusting weight of experts

After training during the first batch, the algorithm updates the weight of its experts according to their decisions. As initially specified, experts have at the beginning 1 as weights. Then, each time an expert misclassifies an instance, the weight of this learner has to be reduced by multiplying it by β in order to give more importance to experts having made a correct prediction. As shown in Figure (2.1), e_1 and e_2 predict correctly the first instance. Hence their weights are maintained at 1. However, e_3 misclassifies the first instance x_1, then its weight becomes $w_3 = 0.5$. For the second instance, the weights used are those resulting from the first instance. For example, e_3 classifies x_2 correctly so its weight is maintained at $w_3 = 0.5$ whereas e_1 and e_2 misclassify x_2 hence, $w_1 = w_2 = 1 \cdot 0.5 = 0.5$. The same reasoning will continue until training of all instances and adjusting classifiers' weights depending on their performance. Once training and weight adjustment are achieved, the global prediction of the algorithm called Λ is computed through the function *argmax* which chooses the class predicted by the expert with the highest weight. Next page provides an illustrative example of this step. The function of weight update is called in line 11 of the Algorithm (2.1).

2.3.2 Weight normalization

After achieving weight updating step, normalization would take place in order to prevent newly added experts from dominating the ensemble. Normalizing weights consists of adjusting the classifiers' weights so that the sum equals 1. The formula when the weight are normalized is:

$$w_j^{'} = \frac{w_j}{\sum_{j=1}^{m} w_j}, \tag{2.6}$$

where w_j is the weight of each classifier j for $j \in 1, \ldots, m$. Consequently the total sum of weights will be:

$$\sum_{j=1}^{m} w_i^{'} = 1, \tag{2.7}$$

When a new batch arrives and a new expert has to be added, weights resulting from the last batch are kept the same, whereas the new classifier's weight is initialized to 1. Adjusting and normalization continue each time there is a new stream. This step is used by **[Kolter and Maloof,**

2007] to prevent new added experts from dominating the ensemble. For each batch there would be removing and adding of experts. This step discussed next is described in the illustrative example of Figure (2.2) and the function of normalization is called in line 16 of Algorithm (2.1).

2.3.3 Expert removal

After adjusting the weight of classifiers, normalizing them and finding the local prediction, even good classifiers begin to loose steadily their weights until reaching the value of the threshold θ. If after many training instances, the weight of e_1 is under θ, for example fixed at 0.12 in the overview of Figure (2.2), this expert should be removed from the ensemble because it does not contribute to the performance of the prediction. This step function is called in line 17 and 18 of Algorithm (2.1) and detailed in Algorithms (2.5) and (2.4).

The value of θ will be discussed in Chapter 3 first by making cross validation with several values and second by an optimization algorithm in Chapter 4.

2.3.4 Adding expert

After many training instances, if the global prediction is different from the class label, the algorithm creates another expert in order to improve the global prediction denoted by Λ. This technique of adding and removing experts makes the algorithm dynamic and flexible whatever are the local and the global prediction, correct or wrong. The new expert added is initialized at one, and then, the new ensemble becomes $\{ e_1, e_2, \ldots, e_{m+1} \}$ as shown in Figure (2.3) and shown in line 23 of Algorithm (2.1). In Figure (2.4), the removing and adding of experts' steps are detailed.

This new ensemble presents the new pool of experts that will be the inputs of the algorithm for the next batch. Before the arrival of a new stream of data, this pool is trained while taking into account the impact of the new experts on the global prediction of the ensemble. This Λ presents the final output of the method.

Although DWM presents an intelligent heuristic for online classification method in non-stationary environment, it has some limitations. In fact, this method does not take into account the age of the expert in the ensemble nor the classifier's correct prediction. DWM reduces the weight of the classifiers with wrong predictions but does not consider those with correct predictions for many batches in the whole process. In the next section, we present our proposed method based on considering the age of the experts in the ensemble as well as their correct prediction during the learning process as a new criterion of classifier's selection. Our enhancement is explained in details and illustrated with an overview in Figure (2.2).

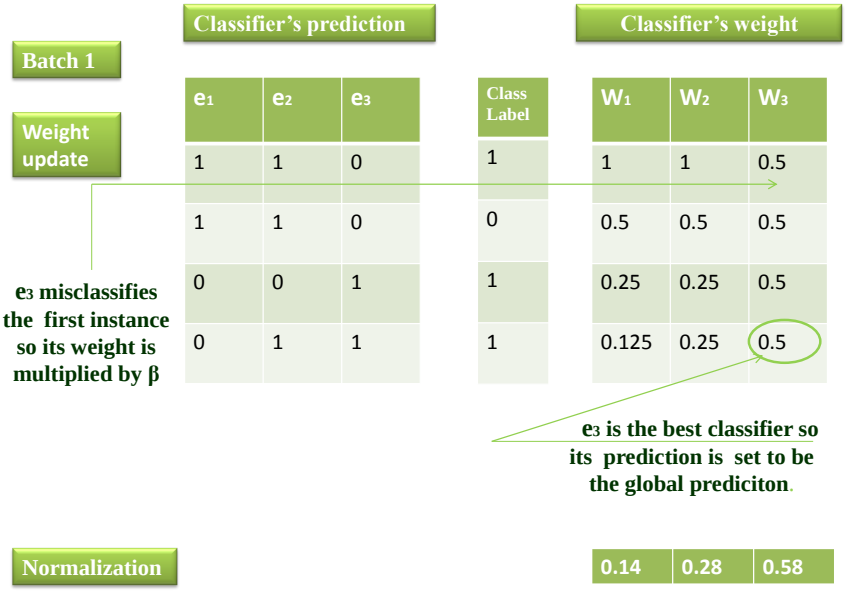

Figure 2.1: Overview of the steps of DWM in Batch 1: adjusting expert's weight and normalization.

2.4 DWM-WIN algorithm

In DWM algorithm classifiers are built from batches of training examples and their weights are adjusted according to their performance. In fact, the weight of a classifier is halved after a wrong

prediction and it is maintained at 1 if the expert predicts correctly. Once the weight adjustment task is achieved, DWM uses each expert's prediction and its weight to evaluate each class. Accordingly, the class with the highest weight is maintained to be the global prediction. If the classifier weight reaches the fixed threshold discussed earlier, the expert will be removed. Finally, each time the global prediction is wrong, a new expert should be added to the ensemble.

The same process would continue for each new batch. At each step, there will be removal and adding of experts.

This step may be criticized. In fact, if we suppose that there are 50 instances in the batch, a classifier that makes 10 mistakes with the first instances or at the end of the batches would end with the same weight and would be treated the same way. In fact, in this example, σ would be equal to 0.5^{10} whatever is the time of the error.

Assume we have trained k batches, so experts with the same weights, are analyzed with the same way. They would be removed or maintained similarly, whereas one learner may make a correct prediction during all batches and wrong predictions only on last batches, and another one may be added later and quickly makes many mistakes since its belonging to the ensemble.

Algorithm 2.1: Optimized DWM-WIN algorithm

input : D = $t_1, t_2, t_3, \ldots, t_n, t_{n+m}$: Data stream
$(\overrightarrow{X_i}, y_i)$: Training Data and class label
m: number of base classifiers in the ensemble
$\overrightarrow{e}$: vector of base classifiers
$\overrightarrow{w}$: vector of weight of base classifiers
$\overrightarrow{\sigma}$: total sum of weighted prediction of each class
Λ: Global prediction of the ensemble
λ: local prediction of each individual classifier
$\tilde{n}$: number of batches used to built a base classifier
n^*: number of batches used to test the ensemble

```
1  InitializeDWMWIN;
2  while (New coming data) do
3      σ⃗ = 0⃗;
4      Window ← Window ∪ { new batch };
5      i ← i+1;
6      if i mod n* = 0 then
7          for j ← 1, ..., m do
8              λ ← classify(e_j, x⃗);
9              if λ ≠ y_i then
10                 UpdateBaseLearnersWeight;
11                 σ_λ ← σ_λ + w_j // Weighted sum for each class
12             end
13         end
14         Λ ← argmax_j(σ_j) // Maximum of the weighted sum predictions
15         w ← WeightNormalization(w);
16         {e, w } ← UpdateBaseLearnerState ({e, w }, θ);
17         {e, w } ← UpdateLearnerEnsemble({e, w }, θ);
18     end
19     if i mod ñ = 0 then
20         if ( Λ ≠ y_i) then
21             m ← m+1 ;
22             e_m = AddNewLearner () ;
23             w_m = 1
24         end
25     end
26     for j ← 1, ..., m do
27         e_j ← TrainModel(e_j, x⃗_i, y_i);
28         Λ ← ComputeGlobalPrediction(Λ)
29     end
30 end
```

Figure 2.2: Overview of the first steps of DWM in Batch 1: removing and adding experts.

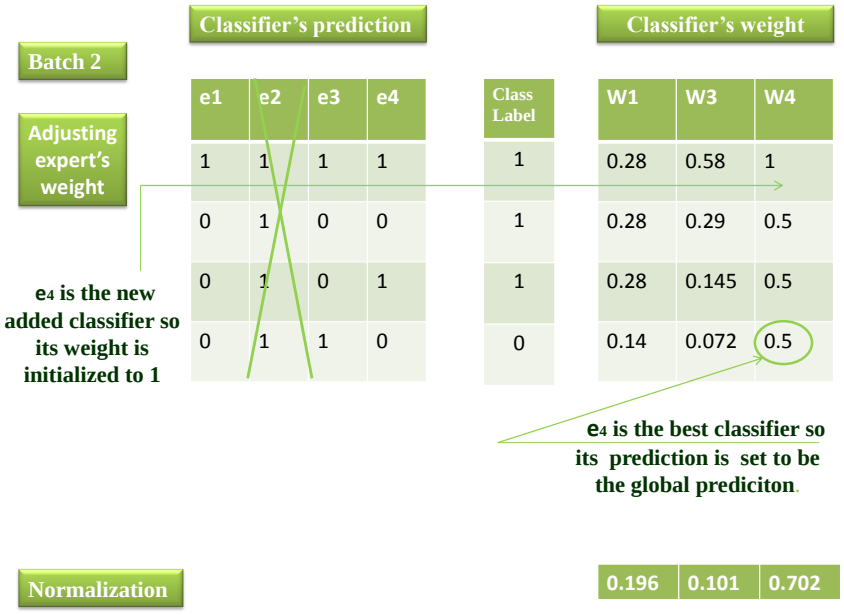

Figure 2.3: Overview of the steps of DWM in Batch 2: adjusting expert's weight and normalization.

Figure 2.4: Overview of the last steps of DWM in Batch 2: removing and adding experts.

Accordingly, it is clear that the first classifier has greatly contributed to the learning process of the ensemble, while the second one does not and despite that they are analyzed the same way according to their similar weights.
This limitation is a serious drawback of DWM algorithm because the contribution of experts to the learning process during the training is not considered. Also, keeping experts based on their past reputation is very important to detect concepts which will recur in the future. For that, including the age of the expert in an ensemble method to track concept drift during the learning would be a way to improve the accuracy of previous algorithms.

In DWM algorithm, a classifier making incorrect prediction is punished by reducing its weight after each error along the training process. It can be removed from the ensemble if its weight is under the threshold θ. For that, it will be more reasonable to reward a classifier that predicts correctly by giving it more chance to exist and to contribute to the learning process.
This idea is based on the Winnow approach of **[Littlestone and Warmuth, 1994]**. This method is based on decreasing the weight of a classifier, successively by multiplying it by a parameter $\beta \leq 1$, each time a classifier makes a wrong prediction and increasing its weight by multiplying it by a parameter $\eta \geq 1$, if its prediction is correct as described in Algorithm (2.3). Figures (2.5), (2.6), (2.7) and (2.8) illustrate a detailed example of proposed method.

If we add this reasoning to the algorithm of **[Kolter and Maloof, 2007]**, DWM will lead to a new algorithm that performs better than the previous one. We call the proposed algorithm DWM-

WIN. Subsequently, performant experts will have more chance to remain in the ensemble and will not be removed from the ensemble due to their high weight.
Algorithms (2.1), (2.2), (2.4), (2.3) and (2.5) describe DWM-WIN algorithm and detail its different functions. It is necessary to note that there is no difference between the two algorithms when experts missclassify all instances during the same batch. Algorithm (2.1) details the different steps of DWM-WIN algorithm.

Algorithm 2.2: InitializeDWMWIN

input : m: initialized ensemble size

```
for i← 1, ..., ñ do
    New Coming Data;
    Window = Empty set
end
m_e ← Initialize ensemble size
c_e ← Construct base classifiers
w_1 ← 1
i ← 0
```

Figures (2.5) and (2.6) give an overview of the different steps of DWM-WIN during the first batch. Figures (2.7) and (2.8) illustrate the different steps in Batch 2.

2.5 Summary of DWM-WIN algorithm

DWM-WIN is proposed as an enhanced method of DWM by using four mechanisms to cope with concept drift. First it trains online learners of the ensemble. Second it weights them based on their performances, if after many training time steps one or many learners make several mistakes it removes those learners (third step), and finally it adds new ones based on the global performance of the ensemble. Different types of base learners could be used in this purpose. Here we use CART of **[Breiman et al., 1984]**. Inputs of the algorithm are a set of m training experts, each with a weight of one, and n training examples each consisting of a feature vector and a class label. DWM-WIN takes into consideration many parameters such as β which is a multiplicative factor to decrease the expert's weight when it makes a wrong prediction. It begins by creating a fixed number of experts and a single batch from the stream. At the beginning, the learner could predict a default class so DWM-WIN takes a batch from the stream and presents it to the single learner to classify. If the learner's prediction is wrong, then DWM-WIN decreases the learner's weight by multiplying it by β. Otherwise, if the classifier makes a correct prediction, its weight is increased by a parameter η to consider its contribution to the learning process and to take into account the age of classifiers in the ensemble as a new criterion of classifiers selection. When only one expert exists in the ensemble, its prediction is DWM-WIN's global prediction.

Algorithm 2.3: UpdateBaseLeanrnersWeight

input : w: the weight of classifiers in the ensemble
β: a parameter for decreasing the weight of learners in the ensemble
η: a parameter for increasing the weight of learners in the ensemble
λ: a local prediction of classifiers

```
for j← 1, ..., m do
    if (λ ≠ y_i ) then
        w_j ← w_j · β
    end
    else
        w_j ← w_j · η
    end
end
```

Algorithm 2.4: UpdateBaseLearnersState

input : θ: a threshold for removing learners from DWM-WIN ensemble according to their weight

```
for j← 1, ..., m do
    if (w_j> θ ) then
        state_j ← good classifier;
    end
    else
        state_j ← bad classifier
    end
end
```

If DWM-WIN's global prediction is incorrect, a new learner is created with a weight of one and DWM trains the ensemble of experts on the new example. After training, DWM-WIN outputs its global prediction. When there are multiple learners, DWM-WIN obtains a classification from each member of the ensemble. It introduces the new batch to the ensemble and train the old classifiers on the new examples.

If an expert's prediction is incorrect, then DWM-WIN decreases its weight and compares the weighted vote prediction of every expert with the local prediction λ. However, if experts predict correctly, their predictions present the global predictions of the ensemble. After training of m experts, the global prediction will be the class of the expert with maximum weight. If after many training episodes, the weight of the classifier is below a threshold θ then DWM-WIN removes it from the ensemble. If the global algorithm predicts incorrectly then the algorithm creates a new

Algorithm 2.5: UpdateLearnerEnsemble

```
for j ← 1, ..., m do
    if state_j = bad classifier then
        Remove classifiers from the ensemble;
        else if state_j = good classifier then
            Classifier remains in the ensemble
        end
    end
end
```

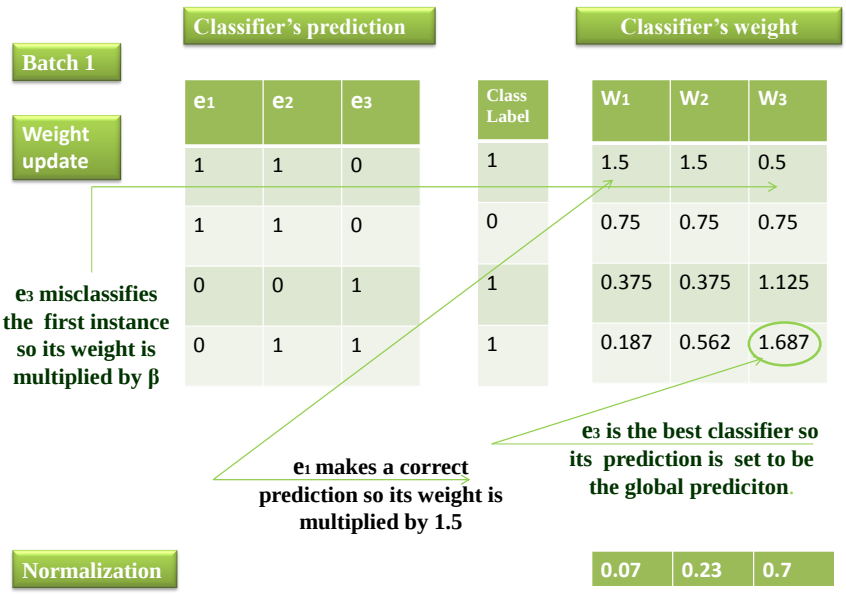

Figure 2.5: Overview of the steps of DWM-WIN in Batch 1: adjusting expert's weight.

classifier with weight of one and trains all experts in the same batch. Every time DWM-WIN decreases or increases the weights of the experts, it normalizes weights at the end of each batch in order to prevent newly added experts from dominating the prediction.

Finally, after using the new example to train each learner in the ensemble, DWM-WIN outputs the global prediction which is the weighted vote of the expert's predictions. Thus, the proposed method highlights the remaining good classifiers contributing to the learning process in the ensemble while considering the history of the process through these classifiers. Actually it considers the age of experts as well as their history as a new criterion of the best ensemble selection.

Figure 2.6: Overview of the steps of DWM-WIN in Batch 1: removing and adding of experts.

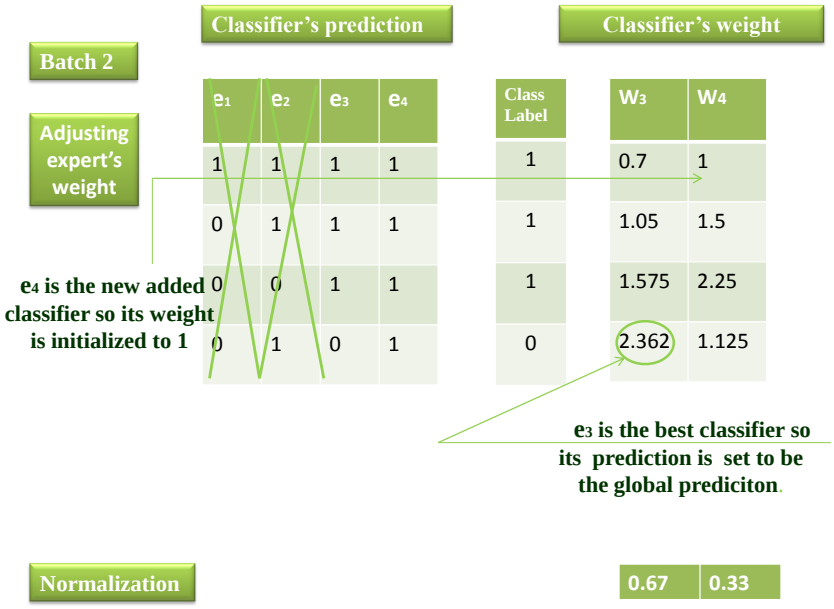

Figure 2.7: Overview of the steps of DWM-WIN in Batch 2: adjusting expert's weight and normalization.

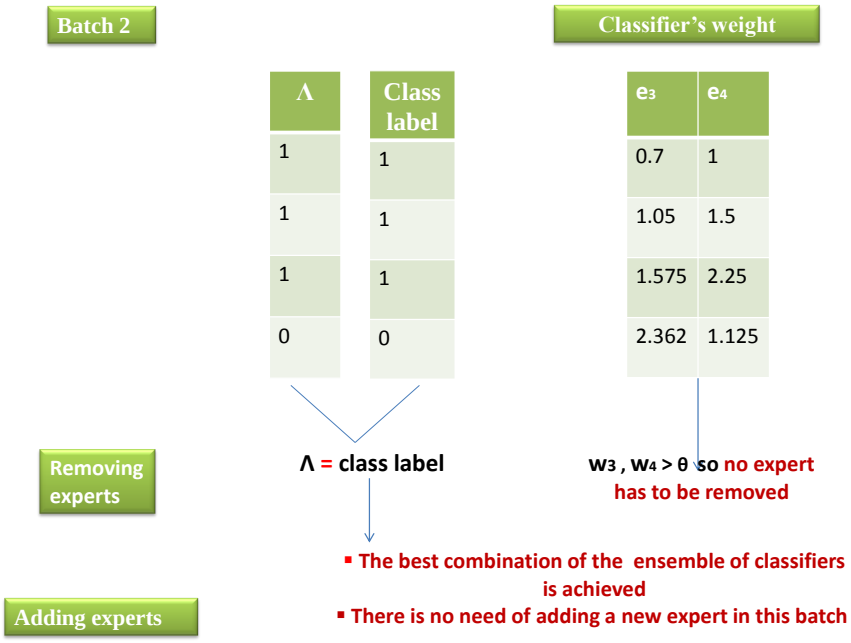

Figure 2.8: Overview of the steps of DWM-WIN in Batch 2: achievement of best combination of classifiers more quickly than DWM.

2.6 Conclusion

This chapter presented an overview of DWM and DWM-WIN methods and detailed the different steps of the proposed method. To obtain the best combination of classifiers that are able to cope with nonstationary environment, a new weight adjusting function which takes into account classifier's contribution to the learning process as well as their age is proposed. In the next chapter, we evaluate the performance of the proposed method on a benchmark dataset and we compare its effectiveness with the basic DWM-WIN.

Chapter 3

Assessment of DWM-WIN Method on Benchmark Datasets

The proposed method is based on two new criteria namely the contribution of good classifiers during the learning process and the age along the history of the classifiers in the ensemble. The proposed method aims to maintain a certain stability in the ensemble by keeping good classifiers longer in the ensemble and benefiting from their learning history of detecting abnormal changes and adapting to new concepts after change occurrence. This chapter investigates the application of the proposed DWM-WIN method and compares it with the basic DWM algorithm based on a benchmark dataset.

Section 1 describes the procedure used to build different classifiers and the methods used for the parameters value selection. Section 2 performs the evaluation of DWM-WIN on a benchmark dataset. Section 3 provides optimal DWM-WIN using different values of β and Section 4 investigates comparisons with the basic algorithm. Section 5 concludes this chapter.

3.1 Classifiers ensemble

In order to construct the ensemble method, different classifiers should be assessed. Frequently, for classification tasks, decision trees are used as the base classifier. The well renowned Classification And Regression Trees (CART) method of [Breiman et al., 1984] is one of the most widely used algorithm to construct classification trees. First, similar decision trees are trained on different subsets in order to build an ensemble of several learners. The method of combining many decision trees is called the random forest of **[Breiman, 2001]**. In this context, **[Hamza and Larocquea, 2005]** show that this method is the best compared to other methods. Second, we vary one parameter of the decision tree to get more diversity between learners. This parameter is the splitmin, which is a number n such that impure nodes must have no more observations to be splitted. Hence,

we fix this parameter at 50% of the data set size so that classifiers are splitted until impure nodes size reach the half of the subset size.

Classifiers are trained on the different batches of data from 1 to i and tested on the new $(i+1)^{th}$ arriving batch. Hence, cross validation is conducted with the arrival of every new batch. Best values of parameters are selected based on the error rates after the cross validation. In each subset, a random proportion of the data is selected to train classifiers based on a sampling with replacement technique of **[Breiman, 1996]**.

Table 3.1: Datasets used in experiments of DWM-WIN.

Datasets	**Instances**	**Number of classes**	**Number of attributes**	**Batch size (no. batches = 20)**	**Batch size (no. batches = 50)**
Pima	768	2	8	38	15
Iris	150	3	4	7	3
Tictactoe	958	2	9	47	19
German	1000	2	20	50	20
Credit Approval	690	2	15	34	13

The second method used to select best parameters values is the use of Evolutionary algorithm. The integration of this technique in DWM and DWM-WIN will be developed in the next chapter. We use m classifiers where m is initialized at 3, and we vary the number of the time step per batch which is defined as the batch rank during the execution process of the algorithm. Different number of batches are used: 5, 20, 50 and 100. Evaluation of the two ensemble methods is done by computing the percentage of performance of each ensemble method in classifying a correct situation. Performance measure is defined as the complement of the error and is computed as follows:

$$Perf_i = (1 - Error_i) \tag{3.1}$$

where i is the time step during the training process. This statistic defines the capacity of the algorithm to cope with the system variation.

3.2 Performance evaluation of DWM-WIN algorithm

For the present study, five data sets from the UCI machine learning repository (http://www.ics.uci.edu/mlearn/MLRepository.html) are considered. They are: Pima, German, Credit Approval, Tictactoe and Iris. These datasets do not contain concept drift, they are used in order to prove that the method can identify fixed concepts. In chapter 5, we will investigate on the assessment on DWM-WIN in datasets with concept drift. Table (9.2) gives a description of some characteristics of the used datasets.

3.2.1 Impact of DWM-WIN on the performance of the algorithm

DWM-WIN supports the idea that every time a classifier makes a correct prediction its weight increases and it will have more chance to not be removed since its weight is higher than the defined threshold. More precisely, this method highlights the importance of the contribution of good classifiers to the global performance of the algorithm in order to reduce the global error rate. Table (3.2) and (3.3) provide the error rates of the two algorithms for the different data sets using a number of batches of 20 and 50 respectively. It indicates that our method has a better performance than DWM method because it considers new criterion for the prediction. In fact, our proposed method outperforms DWM in all batches for all considered data sets. In terms of error rates DWM-WIN has lower error rates since the first batch.

We attribute the performance of DWM-WIN to training learners during different batches. Since our method considers past expert prediction as a criterion, it allows experts with many correct predictions to be maintained in the learning process and to not be removed as opposed to recently added experts that do not contribute to the learning process. Figure (3.1) displays the performance rates of DWM and DWM-WIN on different data sets.

Table 3.2: Error rates of DWM and DWM-WIN based on 20 batches.

Dataset	**Error of DWM**	**Error of DWM-WIN**
Pima	0.24	0.222
Tictactoe	0.24	0.24
German	0.288	0.28
Iris	0.05	0.044
Credit Approval	0.173	0.124

Table 3.3: Error rates of DWM and DWM-WIN based on 50 batches.

Dataset	**Error of DWM**	**Error of DWM-WIN**
Pima	0.289	0.277
Tictactoe	0.231	0.231
German	0.288	0.288
Iris	0.150	0.136
Credit Approval	0.2	0.187

The curve in red presents DWM and the blue one presents DWM-WIN. It can be seen that blue curves are most of the time above the red curves for all data sets. Thus, the performance of DWM-WIN is higher than the performance of DWM for the majority of instances. Results reach

sometimes as high as 100 % for all data sets. This is explained by the fact that DWM algorithm is more suitable for small data sets. In this study, data are subdivided into many batches to construct a streaming data sets arriving over time. That is why high performance rates are explained by smaller size of batches. Our proposed method increasingly maintains good classifiers in the ensemble to improve the global prediction performance.

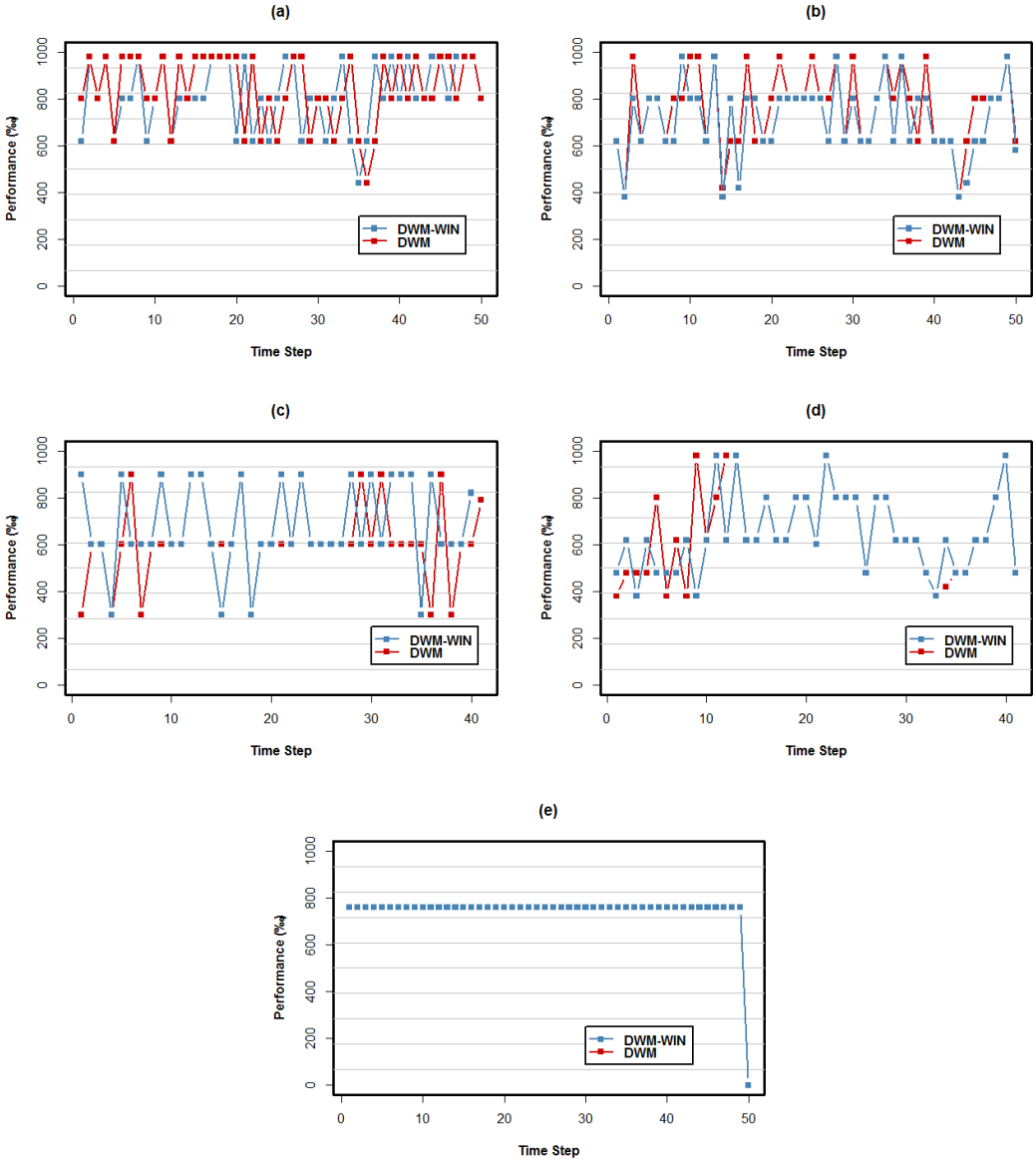

Figure 3.1: Comparison between the error rates of DWM and the proposed algorithm DWM-WIN for 50 batches where (a), (b), (c), (d) and (e) represent, respectively, Credit Approval, Pima, Iris, German and Tictactoe data sets.

3.2.2 Impact of our algorithm on the improvement of the prediction from a single to an ensemble of classifiers

In this section, we explain how DWM-WIN algorithm improves the use of a single classifier. To do that, Tables (3.4), (3.5) and (3.6) show the average error rates of the ensemble of classifiers incrementally trained with DWM. We conclude that the error of the ensemble is low or equal to the single algorithm's error for five batches incrementally learned. This result was proved whatever is the parameter β that reduces the expert's weight and the threshold θ for removing experts having a weight lower than this parameter. Yet, it was assumed by **[Kolter and Maloof, 2007]** that these parameters are fixed, respectively, at 0.5 and 0.01. For instance, for 50 batches applied on Pima data set, when we train three classifiers each one individually, their error rates are, respectively, 0.521, 0.478 and 0.492 during the first batch. However, the combination of these three classifiers reduces the error to 0.4638 which is below the three error values cited earlier. Furthermore, for the fourth batch, two classifiers trained separately on the ensemble have equal error rates at the beginning which is 0.362. However, the error was reduced to 0.3189 after their combination. Tables (3.4), (3.5) and (3.6) summarizes our experimental results. They illustrate the fact that DWM error rates are lower than the error rates when applying classifiers separately.

Another case is the one where a single classifier exists in the ensemble and its prediction is identical to the ensemble's prediction. Then, its error is equivalent to the error of the global algorithm (err_{ens}). Consequently, in all cases, err_{ens} never exceeds err_1, err_2 and err_3, it is usually less or equal to those of single learners.

Table 3.4: The error rates of ensemble of classifiers incrementally trained with DWM on Pima and Tictactoe. Empty cases represent removed classifiers with weights lower than the threshold θ=0.01.

Dataset	Pima				Tictactoe			
	err_1	err_2	err_3	err_{ens}	err_1	err_2	err_3	err_{ens}
Batch 1	0.52	0.47	0.49	0.46	0.34	0.34	0.34	0.34
Batch 2	0.59	0.51	-	0.51	0.34	0.75	-	0.35
Batch 3	0.32	-	-	0.32	0.34	0.71	-	0.34
Batch 4	0.36	0.36	-	0.32	0.34	0.46	-	0.34
Batch 5	0.37	0.48	0.36	0.36	0.35	0.38	-	0.35

On the other hand, Tables (3.8), (3.7) and (3.9) show how DWM-WIN algorithm improves the prediction of an incremental algorithm versus a single classifier. In fact, when each classifier is individually trained in the first batch, error rates are, respectively, 0.42, 0.347 and 0.376. However, this error is only 0.275 due to applying a set of classifiers. Similar results are found for all batches.

Finally, compared to DWM, this improvement from the single classifier to the ensemble is more significant when using DWM-WIN as shown in Figure (3.2).

Table 3.5: The error rates of ensemble of classifiers incrementally trained with DWM on German and Credit Approval.

Dataset	German				Credit Approval			
	err_1	err_2	err_3	err_{ens}	err_1	err_2	err_3	err_{ens}
Batch 1	0.815	0.98	0.49	0.46	0.34	0.34	0.34	0.42
Batch 2	0.82	0.88	-	0.82	0.46	0.22	-	0.22
Batch 3	0.84	0.98	-	0.84	0.16	0.14	-	0.13
Batch 4	0.83	0.84	-	0.83	0.21	0.18	-	0.18
Batch 5	0.83	0.92	-	0.83	0.14	0.13	-	0.13

Table 3.6: The error rates of ensemble of classifiers incrementally trained with DWM on Pima and Tictactoe.

Dataset	Iris			
	err_1	err_2	err_3	err_{ens}
Batch 1	0.6	0.63	0.66	0.56
Batch 2	0.6	0.03	-	0.03
Batch 3	0.1	0.1	-	0.07
Batch 4	0.06	0.06	0.23	0.003
Batch 5	0.001	0.001	0.067	0.003

Table 3.7: The error rates of ensemble of classifiers incrementally trained with DWM-WIN on Pima and Tictactoe.

Dataset	Pima				Tictactoe			
	err_1	err_2	err_3	err_{ens}	err_1	err_2	err_3	err_{ens}
Batch 1	0.42	0.34	0.37	0.27	0.29	0.34	0.34	0.28
Batch 2	0.44	0.43	0.51	0.37	0.29	0.39	-	0.34
Batch 3	0.47	0.31	-	0.27	0.34	0.34	0.71	0.34
Batch 4	0.36	0.36	-	0.31	0.34	0.34	0.46	0.34
Batch 5	0.37	0.43	0.47	0.34	0.35	0.35	0.38	0.35

3.2.3 Impact of the value of β on the number of trained experts

DWM algorithm adds and removes experts from the ensemble according to the classifier's performance. Actually, this mechanism has a great influence on the error rates of the algorithm. As already shown, β's role is to reduce the weight of classifiers with wrong prediction at each instance from each batch. Each change made on this parameter causes a great influence on the number of classifiers and on the error rates of the algorithm. In the following section, we present simulation results for different values of β. Values of β are chosen based on the value used by **[Kolter and Maloof, 2007]** $\pm$ 0.4 which leads to the parameter values from 0.1 to 0.9. Next we will apply an optimization method for best values of β which uses smaller as well as larger values of this parameter. In fact, the use of high values of β reduces the number of misclassified learners and leads to more classifiers in the ensemble. However, for small values of β, if the learner misclassifies a number of instances, its weight will be more reduced. This leads to have classifiers of low weights which are more likely than others to reach the minimum threshold of existence θ in the training and to be very quickly eliminated from the ensemble. In fact, for small values of β, such as 0.1 and 0.3, curves are presented in the lower part of Figure (3.3) and in most cases the number of experts is between 1 and 4. This is explained by the fact that the expert's weight decreases quickly and consequently the number of classifiers decreases accordingly. When β is high, the weight of classifiers making wrong predictions is slightly reduced. In fact, the probability that this classifier achieves the threshold of elimination becomes smaller. Therefore, there will be more experts existing in the pool and more training time is needed. For higher values of this parameter, curves are represented in the interval $[3,9]$ (cp. $\beta = 0.9$ in Figure(3.3)) and this is tested on successive 50 and 100 batches for five data sets.

Table 3.8: The error rates of ensemble of classifiers incrementally trained with DWM-WIN on German and Credit Approval.

Dataset	German				Credit Approval			
	err_1	err_2	err_3	err_{ens}	err_1	err_2	err_3	err_{ens}
Batch 1	0.82	0.89	0.98	0.81	0.34	0.11	0.29	0.116
Batch 2	0.82	0.88	-	0.81	0.16	0.39	0.22	0.16
Batch 3	0.84	0.98	-	0.79	0.11	0.16	0.14	0.11
Batch 4	0.83	0.84	-	0.76	0.18	0.21	0.19	0.18
Batch 5	0.83	0.92	-	0.77	0.14	0.13	-	0.13

Table 3.9: The error rates of ensemble of classifiers incrementally trained with DWM-WIN on Iris.

Dataset	Iris			
	err_1	err_2	err_3	err_{ens}
Batch 1	0.63	0.63	0.3	0.3
Batch 2	0.2	0.03	-	0.03
Batch 3	0.1	0.1	-	0.06
Batch 4	0.06	0.06	0.23	0.006
Batch 5	0.001	0.001	0.067	0.001

3.2.4 Impact of β on the error rate and execution time

Changes in the number of experts used in the training has a great importance for the algorithm's error rates. To illustrate these results, Table (3.10) shows how the error rate increases as β increases. In fact, the smaller β is, the more a new classifier has a chance to exist instead of an old one with wrong predictions. This is also confirmed by the second result shown in Table (3.10), where reducing the parameter β has not only an impact on the error rates but also on the execution time. Note that we can state that higher values of β are costly in terms of time. In addition, good results are achieved for small value of β by the fact that the error may decrease each time expert's number decreases. For example, fixing β at 0.3 training the algorithm over $10,000$ observations needs 2 minutes and 48 seconds. However, the same algorithm needs 3 minutes and 44 seconds for $\beta = 0.8$.

This is more significant for large data sets. As a summary, the smaller the value of the parameter of decreasing expert's weight, the more accurate is the algorithm since both the error rate and the execution time are reduced. Hence, a value of 0.3 instead of 0.5 will have a significant impact on DWM algorithm. As mentioned in the last chapter, DWM-WIN algorithm requires the use of a fixed parameter denoted by θ, to remove experts from the ensemble. The threshold for removing incorrectly predicted classifiers has a direct impact on the number of experts needed in training. Nevertheless, in terms of error, there is no impact on the value of the error rate. Accordingly, we simply maintain the value of θ at the fixed value 0.01.

Table 3.10: Impact of changing β on the error rates and execution time.

	$\beta = 0.3$	$\beta = 0.4$	$\beta = 0.5$	$\beta = 0.6$	$\beta = 0.7$	$\beta = 0.8$
Error rates	0.179	0.182	0.189	0.189	0.192	0.2
Time (min)	2.48	3.12	3.185	3.13	3.32	3.44

3.3 Optimal algorithm: DWM-WIN using optimal value of β

In Sections 3.2.3 and 3.2.4, we used many cross validations to prove that $\beta = 0.3$ is generally the best value. Also, we discussed DWM using Winnow's approach called DWM-WIN and we have proved its efficiency compared to DWM. In this section, we combine the two results and we discuss DWM-WIN using the optimal value of β.

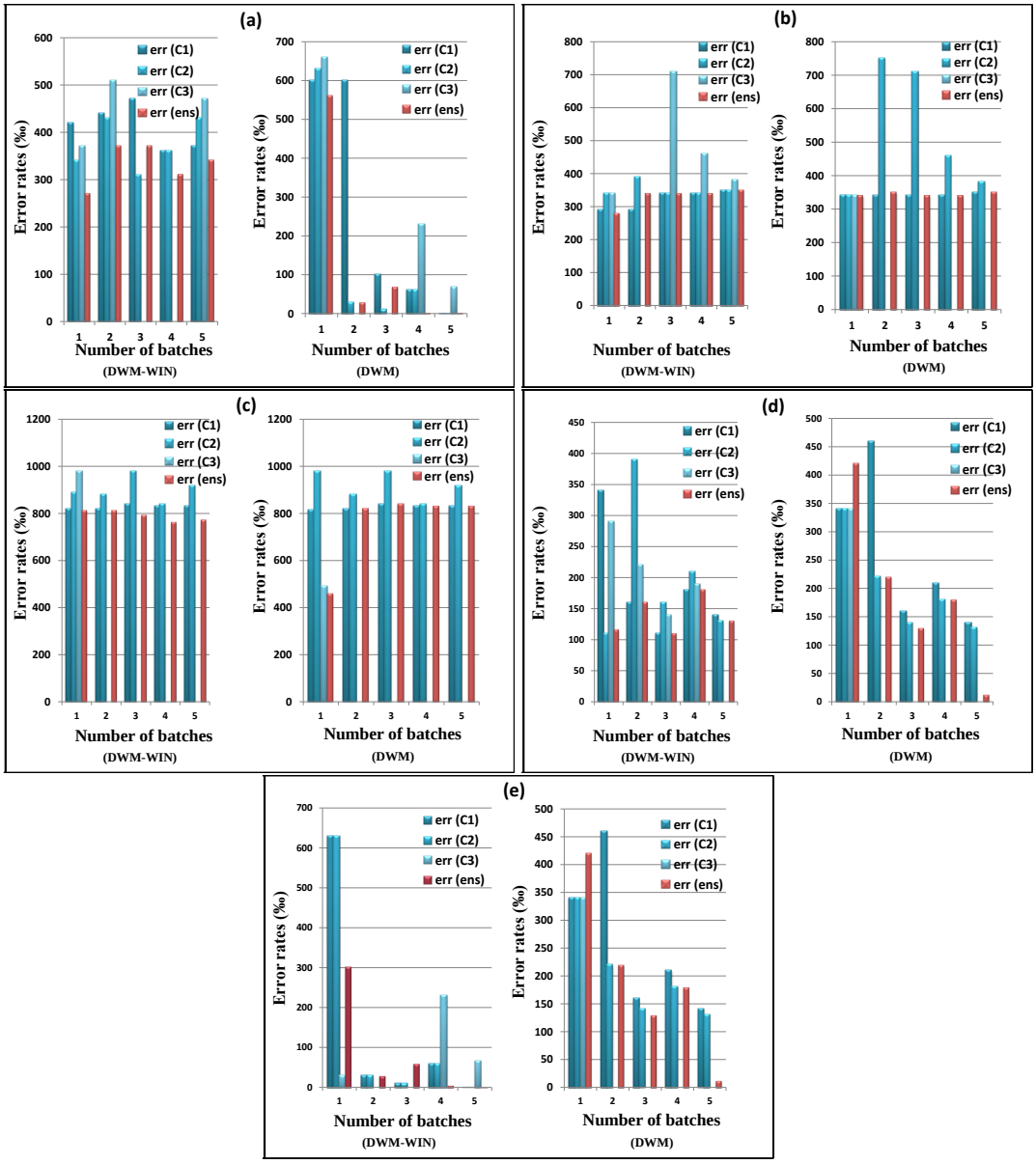

Figure 3.2: Error rates when applying classifiers each one individually and when applying the ensemble for DWM on the left and DWM-WIN on the right where (a), (b), (c), (d) and (e) represent, respectively, Pima, Tictactoe, German, Credit Approval and Iris data sets.

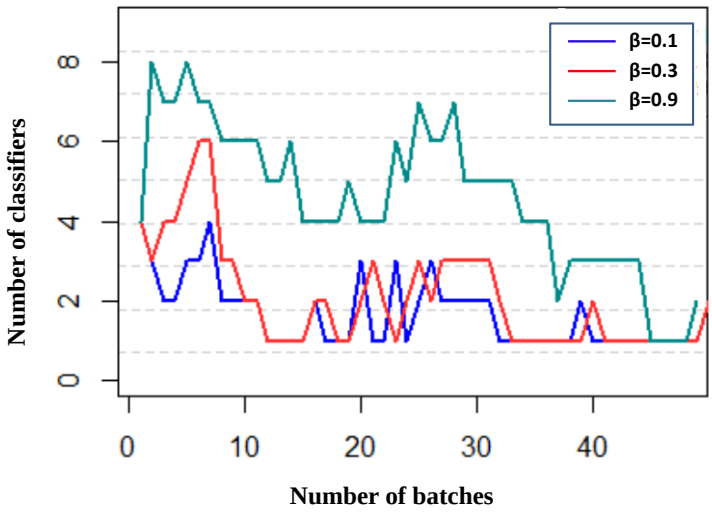

Figure 3.3: Evolution of number of experts versus different values of β.

Empirical results show that using an optimal value of β brings an improvement to the algorithm's performance. As shown in Figures (3.4), (3.5) and (3.6), DWM curve with $\beta = 0.5$ is the highest one since it has the biggest error rates. On the other hand, DWM-WIN using $\beta = 0.3$ and $\eta = 1.5$ is below other curves. In fact, when we reduce the parameter β, from 0.5 to 0.3, classifiers making wrong predictions are seriously punished. Indeed, good classifiers are rewarded for their correct predictions and consequently, the algorithm keeps only the best classifiers. Then, it is directed more quickly towards optimal prediction. As shown in Figure (3.4), DWM-WIN, using $\beta = 0.3$, outperforms the other algorithms by minimizing the algorithm 's error.

3.4 Comparison of execution time of DWM-WIN and DWM

In terms of time of computation, DWM-WIN with $\beta = 0.5$ and $\eta = 1.5$ is faster than DWM algorithm with $\beta = 0.5$ for the different data sets as shown in Table (3.11) because it is able to reach quickly the best combination of experts. This is due to the fact that it rewards good classifiers in the ensemble by increasing their weights as well as decreasing the weight of classifiers making wrong predictions. In consequence, it reaches the optimal algorithm faster than DWM algorithm. For Iris data set the size of data is very small so that the time of execution is low for both algorithms. For that, we limit comparison only on three data sets as shown in Table (3.11).

Table 3.11: Comparison of time execution of DWM and DWM-WIN for 100 batches.

	Pima	Tictactoe	Credit Approval
DWM	15.63 seconds	10.86 seconds	11.14 seconds
DWM-WIN	9.86 seconds	6.68 seconds	7.39 seconds

3.5 Summary of the experiments

Based on DWM algorithm, a modified algorithm called DWM-WIN is proposed. We discuss the ability of an expert to learn from incrementally updated data drawn from a nonstationary environment, where the underlying data distribution shifts in time according to the target class change. Our proposed method creates and removes experts based on their performance while considering their age.

Subsequently, it copes with experts depending on their weight adjusted during the training phase. Particularly, it reduces the weight of misclassified learners and rewards those contributing to the correct global prediction.

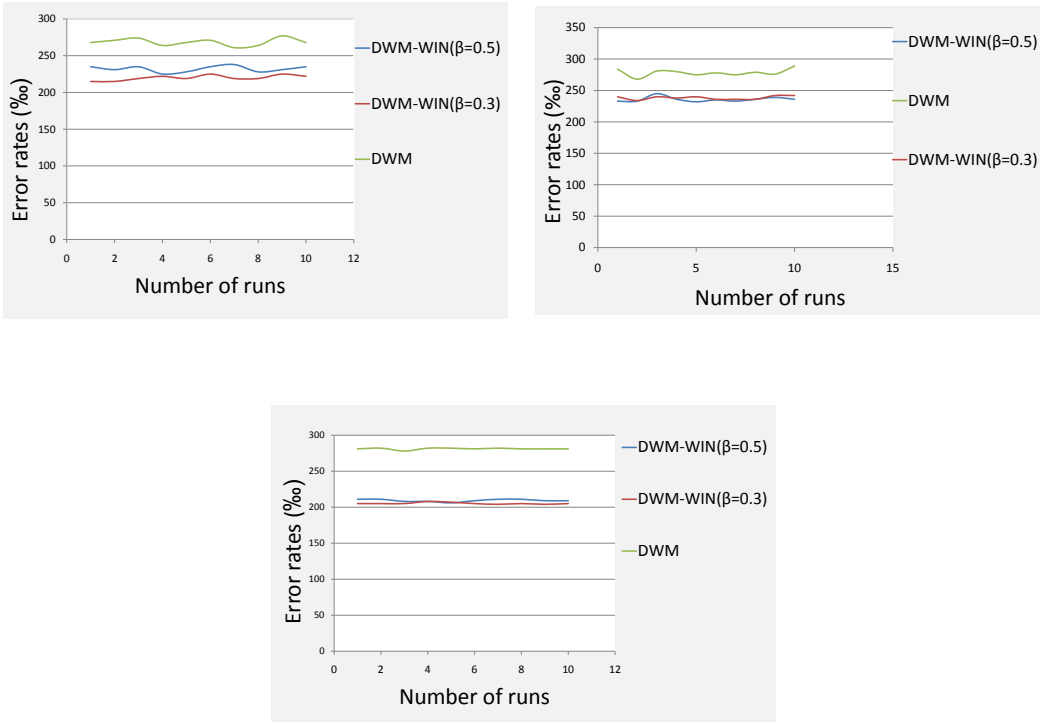

Figure 3.4: Comparison between error rates of DWMWIN using $\beta = 0.3$, $\beta = 0.5$ and the DWM in different chunk sizes, where (a) represents 20, (b) 50 and (c) 100 on Pima Data sets.

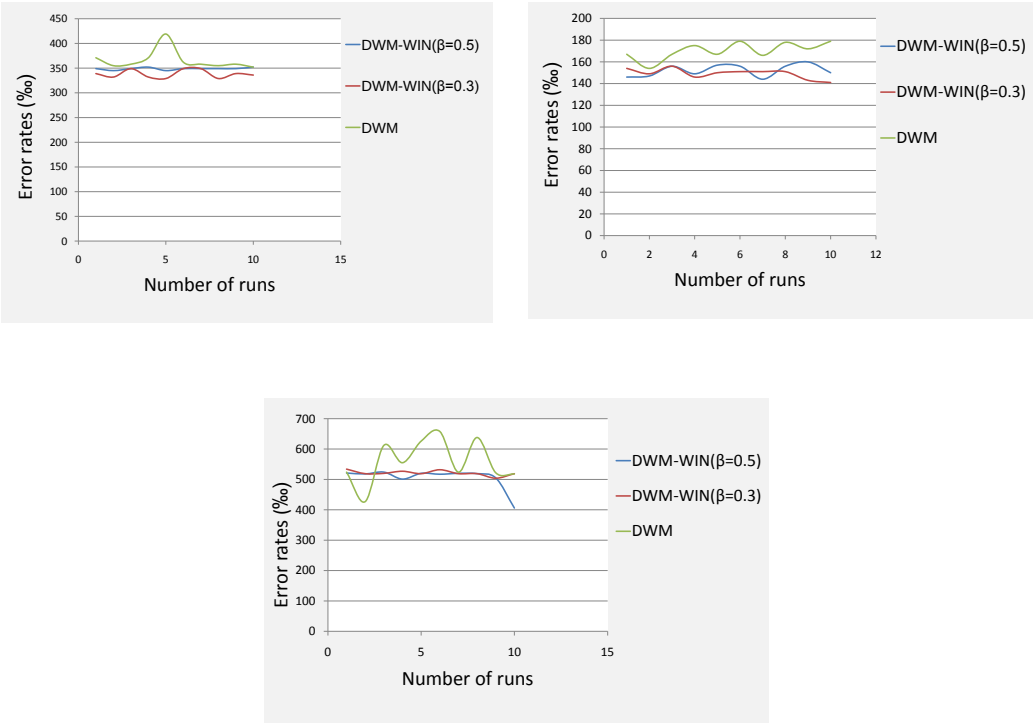

Figure 3.5: Comparison between error rates of DWMWIN using $\beta = 0.3$, $\beta = 0.5$ and the DWM in different chunk sizes, where (a) represents 20, (b) 50 and (c) 100 on German Data sets.

We implemented DWM-WIN using the CART of **[Breiman et al., 1984]** as a base learner. Experimental results show that our method outperforms DWM for different data sets. Also, DWM-WIN reduces experts number and hence the algorithm's execution time. We select the most suitable values of the parameter of adjusting weight β and the threshold θ. Results show that the highest performance is achieved for smaller values of β. Changes in the θ value have an impact on the number of experts and the execution time, but no significant impact on the performance. Concerning the execution time of DWM-WIN is faster than DWM because it quickly achieves the best combination of experts.

3.6 Conclusion

In this chapter, we proposed an improved version of DWM algorithm by including expert's age in the ensemble as well as the contribution of classifiers on the learning process as two new criteria to improve DWM error rates. Although DWM-WIN achieves good performance values, DWM still has other possible improvements. In fact, parameter values of DWM and DWM-WIN are either fixed or randomly chosen. The problem is that cross validation does not present a good and

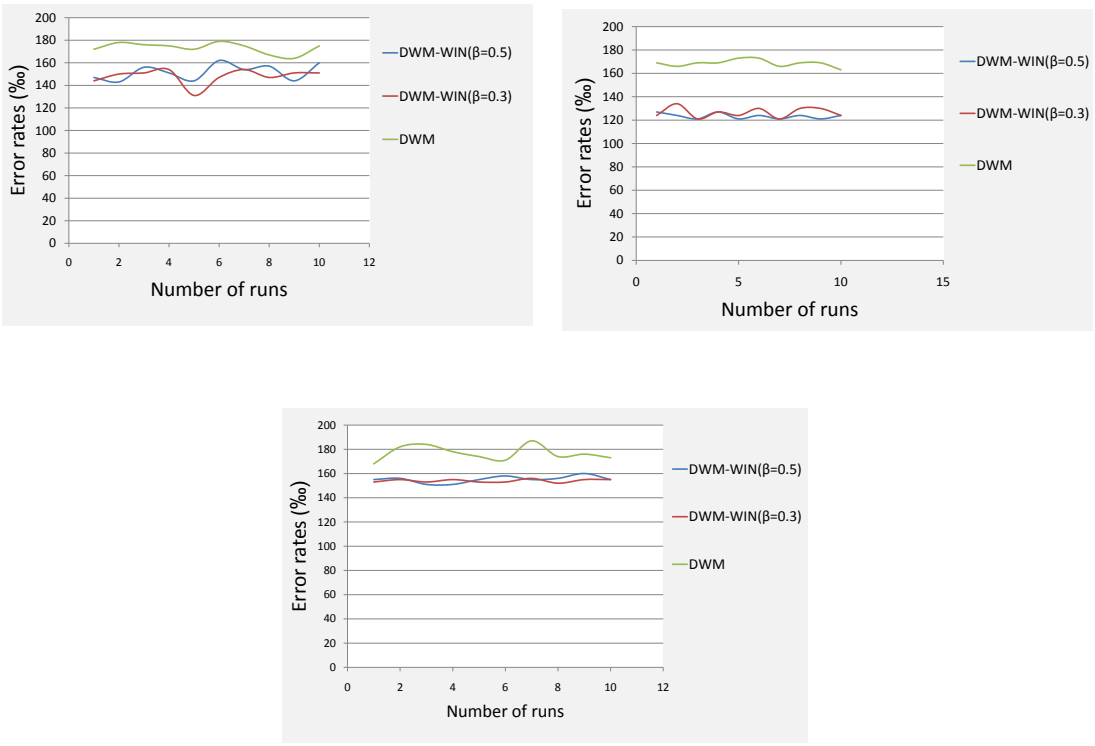

Figure 3.6: Comparison between error rates of DWMWIN using $\beta = 0.3$, $\beta = 0.5$ and the DWM in different chunk sizes, where (a) represents 20, (b) 50 and (c) 100 on Credit Approval Data sets.

automatic method for the parameters' selection since the number of parameters to be tested is not so large compared to many optimization methods and also the answers about the optimized parameters with cross validation are not getting better with time. Consequently, for the 3 parameters, β, θ and η we propose the use of an optimization method to optimize the choice of these 3 parameters since their values greatly impact the classification method's performance. In the next chapter, an automatic optimization technique for choosing the best values of DWM-WIN key parameters based on GA is proposed.

Chapter 4

Optimization of DWM-WIN Parameters using GA

One of the most important and basic principles in our world is to look for an optimal state. An issue in DWM and DWM-WIN algorithms is to select the best parameters values to use during the learning process. In fact, the setting of these parameters impacts on the classification accuracy. These parameters are randomly chosen based on the user decision in DWM and based on a cross validation in DWM-WIN. However, cross validation used in the previous chapter is time consuming and can lead to problems of computation. Also it does not represent an automatic method for parameter selection. The objective in this chapter is to optimize the choice of these parameters. We use Genetic Algorithm (GA) as an optimization method in order to dynamically search for the best parameter values of DWM-WIN and improve the classification accuracy. To do so, DWM-WIN error rate resulted from the testing is used as a fitness function of GA to be optimized. This chapter is outlined as follows: in Section 1, the problem of parameter selection is described. In Section 2, our proposed algorithm is introduced. Section 3 investigates the application and the comparison of the proposed method. Section 4 provides the experiments. Finally, Section 5 summarizes this chapter.

4.1 Parameter optimization

Many mathematical programming methods have been developed to solve optimization problems. Several optimization techniques were proposed in the literature. **[Gimeno and Nave, 2009]** used GA to find the best values of interest rate. They show that using GA leads to better fitted estimations of the parameters needed for computing financial cash flows and testing the effectiveness of monetary policy decisions. Also, **[Lessmann et al., 2006]** proposed a GA for Support Vector Machine (SVM) to detect values of kernel parameters. **[Tan et al., 2008]** used a multiple feature selection criterion to find the ensemble of features that optimizes the classification accuracy. Par-

ticle Swarm Optimization (PSO) developed by **[Eberhart and Kennedy, 1995]** is also a widely used optimization method. PSO optimizes a problem by using some candidate solutions called "particle". These particles are moved in a search space depending on a mathematical formula. Each particle is directed toward the best known position in the search space which is updated when better positions are found. **[Dimitris et al., 1997]** proposed a new hybrid interval GA (HIG) based on branch and bound principle to obtain a small region where the candidate solution lies using an interval subdivision model algorithm. They update the bound in each iteration with an efficient termination criterion. In recent years, ensemble methods have attracted the attention of many researchers in classification domain. One of the recent approaches based on ensemble methods is the work of **[Herwartz and Xu, 2009]**. They propose a new bootstrap method which approximates an F statistic, and show that the new bootstrap scheme is robust against heteroscedastic errors. In this chapter, the DWM-WIN algorithm is used as an ensemble method which optimizes the misclassification error rates function using GA to better cope with non-stationary environment. In the next Sections we first introduce the problem of parameter selection and then detail our proposed algorithm.

4.2 Problem of parameter settings

Many parameters affect the accuracy rate of the DWM algorithm. The parameter β is used to reduce the classifier's weight when it makes a wrong prediction. The threshold parameter θ is introduced to remove experts from the ensemble if their weights are below it. Additional to these two parameters, DWM-WIN uses another parameter, denoted by η used to increase the weights of good classifiers in the ensemble taking into account their contribution to the learning process. All these parameters are randomly chosen without any rational selection. They are chosen by the users and could negatively affect the classification performance in case of bad choice decision. Surely, assigning random values to the parameters of any algorithm could have an important impact on the global performance. On the other hand, selecting the best parameters can be time consuming with problems of computation. A simulation study based on a cross validation was conducted by **[Mejri et al., 2013]** to select the best value of β, while θ and η remain random. Results based on n-fold cross validation where n is the number of batches used during the learning process show that $\beta = 0.3$ is the best value.

For this reason, an optimization method is needed to solve this parameter selection problem. In this context, GA is an interesting evolutionary algorithm for searching the best solutions in complex spaces. The objective of this proposal is to continuously tune the parameters, β, θ and η based on the predictive performance of the classifiers of the ensemble by applying GA optimizer. Afterwards, the classification performance of this model is compared to general DWM-WIN model and to the most widely used standard classification methods which is the regression tree (CART) of **[Breiman et al., 1984]**. Hence, the process of choosing parameter values has to be automated in a rational manner. In order to optimize the process of parameter selection and to find

the optimal combination of the parameters, a GA is applied in the training phase and the results are used in the testing phase of DWM-WIN for each dataset. The next section describes in details the use of GA optimizer in DWM-WIN algorithm to find the best values of its three parameters.

4.3 Proposed method: genetic algorithm for DWM-WIN parameter optimization

Our proposed procedure is to integrate GA into DWM-WIN algorithm in order to deal with the parameter selection problem and to improve the accuracy rate of DWM-WIN algorithm as an online classification method that copes with concept drifting data streams. We call the new proposed algorithm DWM-WIN-GA. Our objective is to find a population of optimal values for each parameter in order to minimize the misclassification error rates. Note that the average error rate is the misclassification error rate of the ensemble of classifiers with respect to current part of stream. The key idea of optimizing DWM-WIN parameters is to consider the error rates of the ensemble of classifiers dynamically trained as a fitness function to be minimized with GA. In fact, GA finds the optimal combination of parameters from the initial population.

It searches an ensemble of hypotheses consisting of different combinations of the three parameters β, θ and η, called initial population. Each hypothesis is evaluated according to the fitness function of the previous population. The used function is represented by a row vector of length m where m is the number of initial combinations. Each value of this vector represents the accuracy of DWM-WIN algorithm for each hypothesis. After the three steps of GA: selection, crossover and mutation, a new population is created updating the parameter selection process each time a new data generation is presented. A population is first initialized, crossover and mutation are applied. Then, DWM-WIN algorithm divides the current part of data stream into blocks and checks the error rate of all classifiers present in the ensemble with respect to each block. Based on the error rate, individual weights of the classifiers are adjusted. DWM-WIN can also create a new classifier or drop some classifiers. The average error rate of DWM-WIN which is the misclassification rate of the ensemble is considered as a fitness function. Then, a selection procedure is applied until reaching the generation number. When the latter occurs, optimized parameters are computed and used in DWM-WIN with the first training data. All GA steps are described below.

4.3.1 Initialized population

A collection of subsets of parameters is selected with GA to find the optimal parameter subsets (β, θ, μ). A population of size (m,n) is constructed, where m is the number of possible values of each parameter and n is the number of parameters. Upper and Lower bound are used for each parameter value. Three intervals are considered in this study: $\beta \in [0,1]$, $\theta \in [0.01, 0.1]$, and $\eta \in [1,2]$. The first step in GA is to generate randomly an initial population. An example of initial population is given in Table (4.1). This population is constructed with an ensemble of strings called genotype or chromosomes. When an initial population is created all strings are

Algorithm 4.1: Pseudo code of DWM-WIN-GA.

input : Training Data ($\overrightarrow{X}$,y, β, θ, μ)
output: Optimal solution C_{max}

Step 1: Generate a feasible solution C_1 randomly ;
Step 2: Compute the fitness function f(x) = DWM-WIN-errors() of each chromosome in the population ;
Step 3: Create a new solution $C_1^{'}$ for C_1 by applying the following steps:
Step 3.1: Selection: choose two parents with the biggest fitness function
Step 3.2: Crossover: perform a crossover over the parents to get a new offspring (child)
Step 3.3: Mutation: swap one or more gene values in a chromosome from its initial state ;
Step 4: Compute individual's fitness function as in Step 2;
if *the obtained solution $C_1^{'}$ is better than C_{max}* **then**
    Update C_{max} with $C_1^{'}$
end
Return the optimal solution C_{max}

evaluated with a fitness function. Evaluation and computation of fitness function are conducted. The first one is a measure of performance of a set of strings in relation to a set of parameters and is independent from one string to another. However, fitness function is usually related to all strings from current population representing the first batches. Average error rate of the ensemble of classifiers is tested with respect to the current block. GA begins with a "current population" representing initial population. Then, after computing $\frac{f_i}{\sum f_i}$ where f_i is the fitness function of the i^{th} string, for all strings, a selection step takes place. It is applied to current population and leads to an intermediate population. Then, recombination and mutation is applied to the "intermediate population" to lead to the "final population".

Table 4.1: Example of an initial generation.

θ	β	η
0.02	0.4	1.1
0.08	0.1	1.5
0.03	0.7	1.8
⋮	⋮	⋮
0.07	0.2	1.4

4.3.2 Representation of the hypothesis: Encoding

Each subset of the parameters is encoded with n bit binary vectors. In our study, the problem is to minimize the error rates of DWM-WIN algorithm which is a function of the parameters (β, θ, η). We can represent each variable by a 30-bit binary number. **[Aljahdali and Telbany, 2008]** assumed that each individual of a GA population is a possible solution of a problem to optimize, encoded as a binary string called a chromosome. Hence, chromosomes should contain three genes and 30 binary digits. In this work, the chromosomes are represented as strings of real numbers, encoding each parameter of the classification method. Parameters have to be encoded into a binary chain. For this, three intervals are set in GA as follows: $\beta_{min} < \beta < \beta_{max}$, $\theta_{min} \leq \theta \leq \theta_{max}$ and $\eta_{min} \leq \eta \leq \eta_{max}$.

When the optimization problem contain more than one variable to be optimized, the coding requires a linking with each variable **[Rangel-Merino et al., 2005]**. In order to encode β, θ, and η, a discretization step takes place in the search space. Then, 30 bits encoding scheme for the three parameters θ, β and η are given respectively by:

$$\theta_{bi} = \frac{\theta_i - \theta_{min}}{\theta_{max} - \theta_{min}} \cdot 2^m \tag{4.1}$$

$$\beta_{bi} = \frac{\beta_i - \beta_{min}}{\beta_{max} - \beta_{min}} \cdot 2^m \tag{4.2}$$

$$\eta_{bi} = \frac{\eta_i - \eta_{min}}{\eta_{max} - \eta_{min}} \cdot 2^m \tag{4.3}$$

This coding means that variables are encoded in a way which allows its use and manipulation during the next steps (selection, crossover and mutation). The above formulation is obtained based on an analogical formulation to the equation given by **[Budin, 1996]** based on the fact that an m digit binary string has 2^m possible 0 and 1 combination. The formula used by these authors to code each gene of length m under some explicit constraints (x_{i1}, ..., x_{ih}) for $i \in (1, 2, \ldots, n)$ is:

$$B_i = \frac{x_i - x_{i1}}{x_{ih} - x_{i1}} \cdot 2^m \tag{4.4}$$

4.3.3 Fitness function: Application of DWM-WIN

The fitness function is an objective function that makes use of a population which artificially reproduces a test solution. It searches the optimal solution from the initialized population and drives the entire population of solutions towards a globally best solution. An ensemble of classifiers is constructed to evaluate the fitness function of each subset by computing the error rate of an ensemble of classifiers trained together. We denote by DWM-WIN error rates the misclassification rates of the ensemble of classifier with respect to the different parts of stream. Note that the average error rate is the mean of misclassification error rate over the different batches, and is given by

$$DWM-WINerrorrates = \frac{1}{n}\sum_{b=1}^{n} error_i \tag{4.5}$$

where b is the number of batches.

4.3.3.1 Operators of genetic algorithm

The operation of GAs begins with a population of random strings representing decision variables. In our study, these decision variables are the parameters of DWM-WIN. The initialized population is then operated by three main operators: selection, crossover and mutation.

a) Selection Chromosomes are selected for reproduction and mutation based on their fitness function value. Best selection will guarantee that the fittest chromosomes are passed on from one generation to the next. **[Rangel-Merino et al., 2005]** assumes that the probability of selecting an individual is expressed mathematically by the following formula:

$$P_i = \frac{f_i}{\sum_{i=1}^{P} f_i} \tag{4.6}$$

where P_i is the selection probability, f_i is the fitness function of the i^{th} individual or string and $\sum f_i$ is the sum of the population's fitness.

b) Crossover: Crossover is the process of taking more than one parent solution and producing a child solution from them. In fact, GA randomly chooses a crossover point to merge the genetic information of two individuals. Several crossover methods were used in the literature, we can set "one point crossover" where all data beyond a single crossover point is swapped between the two parents, "two points crossover" or "cut and splice". In this part, we are interested in a "single crossover point" where the first parent and second parent are combined to generate a child and the crossover point determines which part is taken from parent 1 and from parent 2. In fact, two chromosomes switch portions of their code to create a couple of new individuals.

c) Mutation Mutation is a way in the process of randomly disturbing genetic information. This step represents the random choice of bits that are reversed in each new generation. According to **[Akorede et al., 2011]**, mutation is the genetic operator responsible for maintaining diversity of the population. Based on some probability $\frac{1}{p}$, mutation randomly flips bits of the chromosome to explore the solution space. We note that p is the length of binary vector. The fitness function is computed for each possible combination of the initialized parameters until finding the best subset of parameters using selection, crossover and mutation as operators.

4.4 Experiments

Four data sets are considered in the experiments. They are downloaded from the UCI machine learning repository (http://www.ics.uci. edu/mlearn/MLRepository.html) of **[Asuncion and Newman, 2007]**. These datasets are: Pima, Iris, Tic-tac-toe and Credit Approval datasets. Implemen-

Table 4.2: Parameters used in GA.

Parameter	Value
Population size	20
Number of Generation	10
Crossover rate	0.8
⋮	⋮
Crossover operator	Scattered function
Selection function	Stochastic Uniform
Mutation function	Adaptive feasible

tations were conducted with Matlab. DWM-WIN error rate is induced as a fitness function in GA. We note that we test the method in dataset without concept drift to proof that the optimized method can identify fixed concepts. An assessment of DWM-WIN in datasets with concept drift will be detailed in the next chapter.

4.4.1 Impact of applying GA in DWM-WIN

For DWM-WIN, we divide the data into 20 and 40 blocks to obtain different batch sizes. For GA parameters, we use the following parameter settings shown in Table (4.2). In fact, crossover rate specifies the fraction of the next generation, that are produced by crossover. The remaining individuals in the next generation are produced by mutation. *Scattered function* creates a random binary vector. It then selects from the first parent if the vector is 1, and where the vector is 0 from the second parent, and combines the genes to form the child. For example, if the first parent is $[a,b,c,d,e,f,g,h]$ and the second parent is $[1,2,3,4,5,6,7,8]$, the random crossover vector is $[1,1,0,0,1,0,0,0]$ and the child is $[a,b,3,4,e,6,7,8]$. *Adaptive feasible* is a mutation function which randomly generates the directions that are adaptive with respect to the last successful or unsuccessful generation. Directions means the step length that satisfies bounds and linear constraints and they are also called search regions. Table (4.3) shows a comparison between the error rates of DWM-WIN with and without optimization. The second and the third columns of this table represent the mean of the error rates over the different runs whereas the last column entitled BestFitness indicates the best DWM-WIN error rate obtained during the different runs.

In fact, experimental results show that using GA as an optimization algorithm to select the best combination of the parameters (β, θ, η) for DWM-WIN decreases the error rates for the 4 datasets. In Table (4.3), 40 batches were used for each dataset. For these datasets, it is clear that the use of the optimized parameters computed with GA outperforms the error rates of DWM-WIN algorithm.

Table 4.3: Comparison between error rates of DWM-WIN before GA parameter optimization and after parameter optimization using 40 batches with population size 20.

Dataset	DWM-WIN	DWM-WIN-GA	*BestFitness*
Pima	0.222	0.222	(0.2206)
Iris	0.09	0.0844	(0.0801)
Tictactoe	0.2415	0.2146	(0.2027)
Credit Approval	0.124	0.113	(0.1053)

4.4.2 Best parameters values

GA is applied to detect the best combination of DWM-WIN parameters. For different subsets, the optimal combination of DWM-WIN parameters varies with each subset and impacts the fitness function. Note that **[Kolter and Maloof, 2007]** used random values of the three parameters, and **[Mejri et al., 2013]** used many cross validations to select the best values. In fact, in both cases the choice of these parameters is not automatic and without any rational choice. In this chapter, these parameters are automatically determined improving the accuracy rate of the dynamic ensemble method. Also, this selection of parameters enables the algorithm to cope very well with online data sets and to update the parameter values each time a new stream of data arrives over time. Table (4.4) illustrates the optimal parameter values using 20 batches for the 4 simulated datasets. Different optimal combinations of the three parameters were obtained. Concerning β, it can take values between 0 and 1. If $\beta = 0$, this means that when a classifier predicts incorrectly an instance, it is removed from the ensemble and another one will take its place. In consequence, a small number of classifiers will be obtained at the end. If $\theta = 0$, this means that a classifier is removed when its weight reaches 0, this can only occur when $\beta = 0$. Else, if $\beta \neq 0$ no classifier can be removed from the ensemble and a large number of classifiers will be obtained at the end.

Table 4.4: Best parameter values of best performances (20 Batches).

Dataset	θ	β	η	*BestFitness*
Tictactoe	$4.31 \cdot 10^{-5}$	0	1.095	(0.2027)
Pima	0.1	0.5077	1.6929	(0.2206)
Iris	0.1	0	1.01	(0.0801)
Credit Approval	0	0	1.01	(0.1053)

4.4.3 Effect of selecting appropriate starting population on the error rate of DWM-WIN based on GA optimization

The population size is an important parameter of GA. We studied its influence on the optimization reliability. In order to compare the impact of modifying the size of population on GA accuracy we focus on modifying the population size and fix all the other parameters.

In Table (4.5), a simulation on different datasets with different population sizes: 50, 300 and 500 for 20 batches is presented. We note that error rates depend on the population sizes. Small changes in the population size can affect the algorithm's accuracy. Actually, for small population sizes, there are good and bad cases. The larger the initial population is, the better is the capacity to determine the best combination of the parameters and to decrease the error rates of DWM-WIN algorithm achieved by GA.

For Credit Approval dataset for example, the fitness function has significantly improved from 10.53 % when population size is 20, as shown in Table (4.4), to 8.7 % with a population size 500 in Table (4.5). This means that using a large population size clearly reduces the average error rate. In fact, the error rate values of DWM-WIN-GA corresponding to the largest population size 500 is lower than other error rate values in most of the cases for the different datasets.

Also, the values corresponding to the population size 300 are in most cases below the values corresponding to population sizes 50 and 100. The best parameter combinations are given in Table (4.5).

4.4.4 Comparison of DWM-WIN-GA with other classification methods

In order to show the effectiveness of the proposed DWM-WIN-GA algorithm versus other traditional methods, we have compared its error rate with CART of **[Breiman et al., 1984]**. For the latter, all observations are used together and not in an incremental way. Table (4.6) based on 4 datasets illustrates how our proposed method outperforms the existing ones. In consequence, the optimized DWM-WIN-GA outperforms the DWM-WIN as well as traditional classification methods such as decision trees.

4.5 Conclusion

We introduced an improved version of DWM-WIN algorithm of **[Mejri et al., 2013]** entitled DWM-WIN-GA based on GA as an optimization technique. In fact, experimental results show that combining several classifiers using a dynamic ensemble method technique with GA optimization leads to an improvement of the accuracy for many datasets. This successful optimization technique of a dynamic ensemble method technique is adaptable for different population sizes and for many batches while automating the choice of the parameter values for each dataset. The larger is the population size, the better is the performance of the optimized DWM-WIN-GA and the

Table 4.5: Best parameter values of best performance (20 Batches).

Dataset	θ	β	η	$BestFitness$
Tictactoe (pop size = 50)	0.1	0	1	(0.2027)
Pima (pop size = 50)	0.1	0	1	(0.2142)
Iris (pop size = 50)	0	0	1	(0.074)
Credit Approval (pop size = 50)	0.1	0	1.1909	(0.1003)
Tictactoe (pop size = 100)	0.	1	1	(0.2027)
Pima (pop size = 100)	0.1	0.01	1	(0.2041)
Iris (pop size = 100)	0.1	1	1	(0.08)
Credit Approval (pop size = 100)	0	0.01	1	(0.1053)
Tictactoe (pop size = 300)	0	0.858	1.254	(0.197)
Pima (pop size = 300)	0	0.01	1	(0.2011)
Iris (pop size = 300)	0.1	0.01	1	(0.074
Credit Approval (pop size = 300)	0	0.7143	1.6235	(0.095)
Tictactoe (pop size = 500)	0.	0.1	1	(0.2263)
Pima (pop size = 500)	0.1	0.1	1.190	(0.198)
Iris (pop size = 500)	0.1	0.1	1	(0.063)
Credit Approval (pop size = 500)	0	1	1	(0.087)

Table 4.6: Comparison between error rates of DWM-WIN with GA parameter optimization using 20 batches with populationsize 500 and the CART of **[Breiman et al., 1984]**.

Datasets	CART (Breiman)	DWM-WIN-GA
Pima	0.3	0.198
Iris	0.26	0.063
Tictactoe	0.4	0.2263
Credit Approval	0.15	0.087

lower becomes the error rate. As a future work, it will be interesting to look for feature selection optimization and applying it to multivariate SPC.

Chapter 5

Assessment of DWM-WIN algorithm on SEA Dataset for Concept Drift

After optimizing DWM-WIN in Chapters 2, 3 and 4, here we investigate to apply the enhanced algorithm on a dataset with concept drift. In fact, in real life domain issues are changing over time and the target concept to be learned may change accordingly. Our aim is to monitor classification error rates of an ensemble method and to detect concept drift based on classifier's performance during an online process. The best solution is to adapt the combination of classifiers in the ensemble with each new batch arriving over time by removing some bad classifiers and adding new ones. We propose an heuristic which is able to detect a change without forgetting previous knowledge about the age of the classifiers as well as the past correct prediction in the ensemble. And also there is a need to distinguish between concept drift and out of control situations caused by the non stationarity of the environment. We present an application of detecting concept drifts in SEA data sets with concept drift and its variants. We study the impact of the classifier diversity, the noise level, the permutation of the sequences of concept drift and the number of batches on the DWM-WIN capacity to react to the concept drift. We analyze the results using ANOVA and Tukey's test.

First an insight into SEA dataset is given in Section 1. Section 2 presents the impact of permutation, noise level, number of batches and the classifier type on DWM-WIN performance. In Section 3, we analyze the robustness of DWM-WIN to handle problems of non-linearity. Section 4 analyses the method applied to SEA with and without concept drift. Finally, Section 5 contains our final remarks.

5.1 SEA: A dataset for concept drift

We analyze our method on different variants of SEA dataset of **[Street and Kim, 2001]**. We use the R package mlr of **[Bischl and Richter, 2014]** for calling the R classifiers. We use ensembles

only based on decision trees (rpart), nearest neighbors models (kknn), and naive Bayes models (naiveBayes), respectively. In the following, we study the impact of the batch size, the noise level, the permutation and the capacity of DWM-WIN in detecting and adapting to the concept drift. All results are summarized in Table 5.1. We analyze the results by means of ANOVA using Friedman's test (see Tables (5.2) and (5.3)) and by Tukey's test (see Table (5.5) and (5.4)).

5.1.1 Definition

The SEA dataset was first used by **[Street and Kim, 2001]**, then used by **[Kolter and Maloof, 2005b]** to test the Add-Exp algorithms. It is downloaded from Stream Data Mining repository (http://www.cse.fau.edu/ xqzhu/stream.html) of **[Zhu, 2010]**.
SEA presents a binary classification problem with 60000 observations. Features are independent and identically distributed based on a Uniform distribution $U[0, 10]$. The target concept to be learned is determined based on the function $x_1 + x_2 \leq b$, where $b \in \{7, 8, 9, 9.5\}$. Two classes are distinguished, one where this condition is satisfied and one where it is not. First the data is divided into 20, 50, and 100 batches. Four different concepts occur in the data by adaptation of the class labels in SEA dataset when changing the value of b. For the first 250 batches, the target concept is b=8, e.g., for the second concept $b = 9$, the third target $b = 7$ and the fourth $b = 9.5$. We consider all permutations of the ordering of these four concepts.

5.2 Performance evaluation analysis on concept drift

5.2.1 Impact of the permutation on the concept drift

Reaction to the concept drift: Two types of drift are distinguished: the gradual concept drift and the sudden concept drift. Gradual concept drift is represented by the sequences $(7, 8, 9, 9.5)$ and $(9.5, 9, 8, 7)$. Whereas the other sequences are considered as sudden drift since there is no special characteristic in the concept sequences.
Figures (5.1) and (5.2) present the error results in sudden and gradual drift. The algorithm is trained based on 1000 batches and each concept is presented in 250 batches.

Statistical tests: According to one way ANOVA given in Table (5.2), there is a significant difference in the error means between the different permutations. This result is also confirmed by the two way ANOVA (see Table (5.3)) where the interactions between permutation and learner and permutation and noise are significant.

According to Figure (5.1), the error rates are relatively stable and the algorithm perfectly deals with the gradual drift by learning the drift and using stored information to adapt the algorithm after each drift detection. Concerning some sudden drifts, DWM-WIN shows a different behavior in the error rates. As shown in Figure (5.2), DWM-WIN performs better after detection of the first concept change. In fact both figures show that DWM-WIN error rates are stable after the

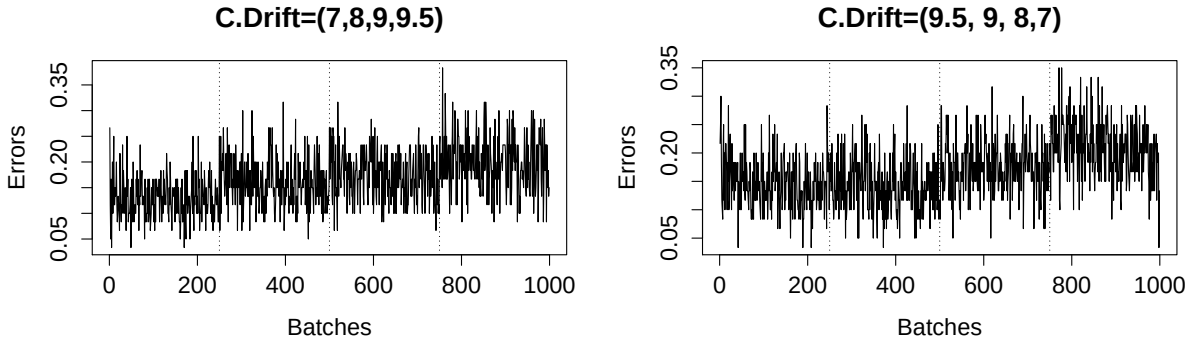

Figure 5.1: Reaction capacity of DWM-WIN error rates on gradual drift.

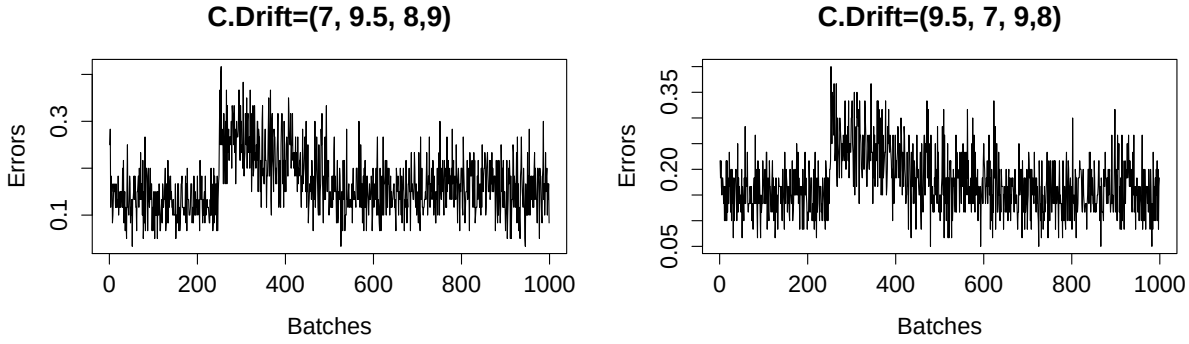

Figure 5.2: Reaction capacity of DWM-WIN error rates on sudden drift.

first concept change. Thus, differences between sudden and gradual drifts might be bigger than between gradual changes themselves.

Conclusion: DWM-WIN performs very well with concept drift for gradual as well as sudden drift. The reason of that is the flexible adaptation of the ensemble of classifiers to the different sequences of the drift occurrence.

5.2.2 Impact of varying the batch size on the error rate

Reaction to the concept drift: The performance of the algorithm is not affected by the number of batches. The DWM-WIN does not show a significant change in the error rate levels. This is due to the good and quick reaction capacity of DWM-WIN in detecting the concept drift and adapting the classifier's ensemble to this change.

Statistical tests: Based on Friedman's test of the one way ANOVA, shown in Table (5.2), we do not reject the hypothesis that the algorithms have the same performance on average for different

no. of batches. For this test, we consider two situations. First when testing the difference between 3 different no. of batches $(20, 50, 100)$ without considering kknn since it does not work in all cases when $N = 100$ (see Table (5.1)). In this case, we have an F value of 0.1399 and a p-value of 0.7079.

Respective results based on Tukey test are shown in Table (5.4). In the second situation, we consider DWM-WIN based on rpart, kknn and naiveBayes where we test the difference only between number of batches of 20 and 50. Results are an F-statistic of 1.053 and a p-value of 0.3505. This result is also confirmed by Tukey's test in Table (5.5) where lwr indicates the lower end point of the interval and upr the upper end point of the interval. Based on the two way ANOVA the interaction of the number of batches with the other factors is not significant at the 5% level. Respective p-values in Table (5.3) are 0.804, 0.69 and 0.0574 for rpart, kknn and naiveBayes.

Conclusion: DWM-WIN has a noticeable robustness on concept drift for the different batch values. Indeed, it is quite interesting that DWM-WIN quickly adapt itself to the concept drift for small batches (10 instances per batch when no. of batches is 100) as well as for large batches (50 instances per batch when the no. of batches is 20).

Table 5.1: Mean error rates for different number of batches $(20, 50, 100)$ versus different basic classifiers, N. perm = 24, N.rep = 100, prob.noise = (0.1, 0.2) for DWM-WIN algorithm.

Classifier	rpart						kknn						naive.Bayes					
	Noise = 10%			Noise = 20%			Noise = 10%			Noise = 20%			Noise = 10%			Noise = 20%		
no. of batches	20	50	100	20	50	100	20	50	100	20	50	100	20	50	100	20	50	100
1.(7, 8, 9, 9.5)	0.19	0.18	0.19	0.29	0.29	0.29	0.18	0.19	0.19	0.31	0.31	-	0.2	0.19	0.2	0.29	0.29	0.29
2. (7, 8, 9.5 9)	0.22	0.21	0.2	0.29	0.3	0.29	0.19	0.18	0.18	0.31	0.31	0.31	0.32	0.32	-	0.28	0.28	0.28
3. (7, 9.5 8, 9)	0.19	0.19	0.22	0.3	0.29	0.31	0.18	0.18	-	0.3	0.3	0.3	0.18	0.19	0.19	0.24	0.26	0.25
4. (9.5, 7, 8, 9)	0.195	0.19	0.19	0.28	0.27	0.28	0.17	0.18	0.17	0.27	0.27	-	0.17	0.17	0.17	0.25	0.25	0.25
5. (9.5, 7, 9, 8)	0.19	0.17	0.19	0.3	0.3	0.3	0.155	0.154	-	0.3	0.31	0.31	0.14	0.145	0.15	0.27	0.28	0.28
6. (7, 9.5, 9, 8)	0.17	0.18	0.19	0.32	0.29	0.31	0.17	0.17	0.17	0.3	0.31	-	0.17	0.18	0.17	0.27	0.28	0.27
7. (7, 9, 9.5, 8)	0.18	0.19	0.19	0.26	0.25	0.26	0.16	0.16	0.16	0.26	0.27	0.27	0.17	0.16	0.17	0.25	0.24	0.23
8. (7, 9, 8, 9.5)	0.16	0.18	0.19	0.27	0.28	0.29	0.15	0.15	-	0.27	0.27	-	0.15	0.17	0.16	0.24	0.25	0.24
9. (9, 7, 8, 9.5)	0.22	0.22	0.23	0.29	0.28	0.32	0.21	0.2	-	0.3	0.3	-	0.18	0.18	0.19	0.27	0.26	0.27
10. (9, 7, 9.5 8)	0.19	0.21	0.20	0.29	0.27	0.28	0.17	0.17	-	0.3	0.29	-	0.17	0.17	0.17	0.25	0.24	0.24
11. (9, 9.5 7, 8)	0.2	0.2	0.21	0.27	0.29	0.3	0.18	0.18	-	0.28	0.3	-	0.16	0.18	0.17	0.26	0.25	0.26
12. (9.5, 9, 7, 8)	0.19	0.2	0.19	0.27	0.29	0.28	0.18	0.16	-	0.29	0.28	-	0.16	0.15	0.15	0.25	0.25	0.25
13. (9.5, 9, 8, 7)	0.2	0.2	0.2	0.26	0.27	0.26	0.17	0.17	-	0.26	0.27	-	0.18	0.17	0.18	0.24	0.27	0.25
14. (9, 9.5 8, 7)	0.2	0.2	0.2	0.28	0.28	0.28	0.19	0.18	-	0.27	0.28	-	0.17	0.17	0.17	0.26	0.26	0.25
15. (9, 8, 9.5, 7)	0.19	0.2	0.19	0.27	0.27	0.27	0.17	0.17	-	0.27	0.26	-	0.17	0.17	0.17	0.27	0.26	0.26
16. (9, 8, 7, 9.5)	0.18	0.19	0.2	0.27	0.28	0.27	0.175	0.18	-	0.26	0.27	-	0.17	0.17	0.16	0.26	0.27	0.26
17. (8, 9, 7, 9.5)	0.2	0.19	0.2	0.25	0.28	0.25	0.174	0.17	-	0.25	0.24	-	0.18	0.17	0.17	0.22	0.23	0.23
18. (8, 9, 9.5, 7)	0.2	0.2	0.2	0.27	0.28	0.28	0.184	0.18	-	0.28	0.28	-	0.19	0.18	0.18	0.25	0.25	0.26
19. (8, 9.5, 9, 7)	0.19	0.21	0.2	0.3	0.31	0.3	0.17	0.17	-	0.31	0.29	-	0.18	0.17	0.17	0.3	0.31	0.3
20. (9.5, 8, 9, 7)	0.21	0.22	0.2	0.3	0.3	0.29	0.19	0.2	-	0.28	0.29	-	0.19	0.18	0.18	0.27	0.27	0.28
21. (9.5, 8, 7, 9)	0.18	0.2	0.21	0.27	0.29	0.3	0.17	0.17	-	0.29	0.27	-	0.16	0.16	0.16	0.26	0.26	0.26
22. (8, 9.5, 7, 9)	0.2	0.2	0.19	0.31	0.3	0.3	0.19	0.19	-	0.32	0.31	-	0.18	0.18	0.18	0.28	0.27	0.28
23. (8, 7, 9.5 9)	0.2	0.22	0.2	0.29	0.3	0.3	0.19	0.18	-	0.29	0.29	-	0.18	0.17	0.18	0.25	0.25	0.25
24. (8, 7, 9, 9.5)	0.2	0.2	0.19	0.29	0.3	0.29	0.16	0.16	-	0.29	0.3	-	0.18	0.17	0.17	0.27	0.27	0.26

5.2.3 Impact of varying the noise level to the error rate

Reaction to concept drift: As expected, changing the level of the noise, shows high impact on the error rate. Changing the noise level from 10% to 20% impacts the error increasing it from 19% to 29% during the first permutation $(7, 8, 9, 9.5)$ for a number of batches of 20 using an ensemble of rpart in DWM-WIN. For naiveBayes, e.g., in the 19^{th} permutation the rate increases by 14% with no. of batches of 50 (see Table 5.1).
Results of DWM-WIN based on naiveBayes are better than those of DWM-WIN based on rpart and kknn for the small noise level = 10%. However, when increasing the noise level, naiveBayes based DWM-WIN approximately achieves the same level of error rates as the other methods.

Statistical tests: Friedman' s test rejected the null hypothesis that the algorithms have similar performance when changing the noise level. In fact, given F statistic, F = 4143.0868 and p-value $< 2.2 \cdot 10^{-16}$, shown in Table (5.2), the algorithm performs differently when the noise level differs. These results were confirmed by the two way ANOVA in Table (5.3) where the Friedman test rejected the null hypothesis that the algorithms have similar performance on average when considering the two factors noise and permutation or noise and type of the learner with a p-value $< 2.2 \cdot 10^{-16}$. This result is also confirmed by Tukey's test as shown in Table (5.5).

Conclusion: We conclude that introducing more noise in the data impacts on the general error level. Naive bayes performs better than other classifiers when handling problems of concept drift. The good performance of naiveBayes classifier in the presence of noise level can be explained by the fact that naiveBayes's structure allows it to be changed with the training data without the need to reconstruct the model. This characteristic makes it cope very well with noisy data in nonstationary environment.

5.2.4 Impact of varying the type of the classifier on the error rate

Reaction to concept drift The capacity of DWM-WIN in detecting concept drift is affected by the type of the learner. As comparison to DWM-WIN based on rpart and naiveBayes, DWM-WIN based kknn outperformed other learners for the different batch sizes and noise levels during the 24 possible permutations. This is due to the fact that the time needed by kknn to update the internal knowledge of DWM-WIN is less than the time needed by other learners, that is why it has smaller error rates. This fact is useful in many concept drift applications where the updating procedure requires to be processed with many steps in a short time.

5.3 Gradual versus sudden concept drift analysis

According to Figure (5.3), the error rates are relatively stable and the algorithm perfectly deals with the gradual drift by learning the drift and using stored information to adapt the algorithm

after each drift detection. Concerning some sudden drifts, DWM-WIN shows a different behavior in the error rate. As shown in Figure (5.3), DWM-WIN performs better after the second concept detection and results are more stable than before. Thus, DWM-WIN requires more time to have regular results in terms of classifiers in sudden drift than in gradual one. Differences between sudden and gradual drifts might be bigger than between gradual changes themselves.

Conclusion: DWM-WIN performs very well with concept drift for gradual as well as sudden drift. The stability in the error results is more quickly achieved in gradual drift than in sudden drift. This is because the gradual concepts including a certain order in the sequences are easier to learn than the sudden ones.

5.4 Linear versus nonlinear concept drift analysis

DWM-WIN algorithm demonstrated a competitive behavior in nonstationary linear online environment and copes very well with sudden drift as well as gradual drift. In the different cases we studied until now, the decision boundaries of our problem are linear. However real world classification issues are often nonlinear. That is why we study the impact of incorporating nonlinear concept drift on DWM-WIN algorithm performance.

Several forms of nonlinearity exist in the literature, however for our concept drift model we choose the following formulas:

$$\begin{cases} (x_{1i})^2 + (x_{2i})^2 < a^2, \\ (x_{1i})^2 + (x_{2i})^2 < b^2, \\ (x_{1i})^2 + (x_{2i})^2 < c^2, \end{cases} \tag{5.1}$$

where a, b and c are the values of the target concept of each subset of the dataset.

Reaction to concept drift Comparative results about the error rates in case of linear and nonlinear concept drift are presented in Figure (5.4). It presents the error rates obtained in DWM-WIN in case of linear sudden, linear gradual, nonlinear sudden and nonlinear gradual drift. The absciss axis represents the no. of batches and the ordinate axis presents the misclassification errors

Table 5.2: Friedman test for one way ANOVA.

Measure	Df	Sum Sq	Mean Sq	F value	Pr (>F)	
p	23	0.323	0.0014	9.1656	$< 2.2 \cdot 10^{-16}$	***
learner	2	0.02859	0.01429	93.2841	$< 2.2 \cdot 10^{-16}$	***
Noise	1	0.63480	0.63480	4143.0868	$< 2.2 \cdot 10^{-16}$	***
N.batches (20,50)	1	0.00002	0.00002	0.1389	0.7097	
N.batches (20,50,100)	2	0.00025	0.00012	1.0534	0.3503	
Residuals	254	0.03892	0.00015			

of DWM-WIN based on an ensemble of rpart, naiveBayes or kknn based on 100 simulations and 100 batches. First of all, we begin with a global analysis of the behavior of DWM-WIN through the different datasets and analyze the behavior from one concept to another.

As it is shown in Figure (5.4), the changes in the first concept are easily detected by distinguishing between the different DWM-WIN methods. The first concept is the time of learning of the first stage of the problem. It does not show any peak and the aspects of the errors are quasi-stable. In the second concept, DWM-WIN behaves better, it shows a high error at the beginning of the concept then it quickly decreases to indicate the self adjustment and the coping with this new concept.

During the third and the fourth concept, although the error rates are higher than in linear case, the behavior of DWM-WIN in all cases during these concepts is very good. There is a peak of the errors in the beginning, then the error quickly converges to the initial error and recovers this increase. As a first summary for this result, DWM-WIN has a very good adaptation reaction after drift detection. Thus, DWM-WIN shows a very good robustness to quickly recover the increase in errors by adapting the algorithm to the detected problem. Now, we perform an analysis in terms of datasets and differences between the ensemble of classifiers. For SEA dataset with gradual and linear concept drift, DWM-WIN-kknn and DWM-WIN-rpart perform better than DWM-WIN-naiveBayes after the first concept. The distinction between the different methods is clearly observed. For SEA dataset with gradual and nonlinear concept drift, the DWM-WIN-naiveBayes performs better than DWM-WIN-kknn and DWM-WIN-rpart. This is expalined by the fact that naiveBayes has more robustness to learn concepts with nonlinearity better than rpart and kknn. Nonlinearity is a desired feature for naiveBayes in datasets with concept drift. Concerning SEA dataset with sudden linear concept drift, the three methods perform differently during the four concepts. In fact, while DWM-WIN-naiveBayes maintains lower error rates in the first, the second and the third sudden and nonlinear concepts, errors are higher during the fourth concept. However, DWM-WIN-rpart begins with relatively high error rates compared to other

Table 5.3: Friedman test for Two Way ANOVA.

Measure	Df	Sum Sq	Mean Sq	F value	Pr (>F)	
p vs learner	46	0.00492	0.00011	1.7223	0.007411	**
p vs noise	23	0.01491	0.00065	10.4332	$< 2.2 \cdot 10^{-16}$	***
p vs batch	23	0.00118	0.00005	0.8286	0.691683	
learner vs noise	2	0.00778	0.00389	62.6127	$< 2.2 \cdot 10^{-16}$	***
learner vs batch	2	0.00036	0.00018	2.91	0.057417	.
Noise vs batch	1	0.000	0.000	0.0612	0.804962	
Residuals	157	0.00975	0.00006			

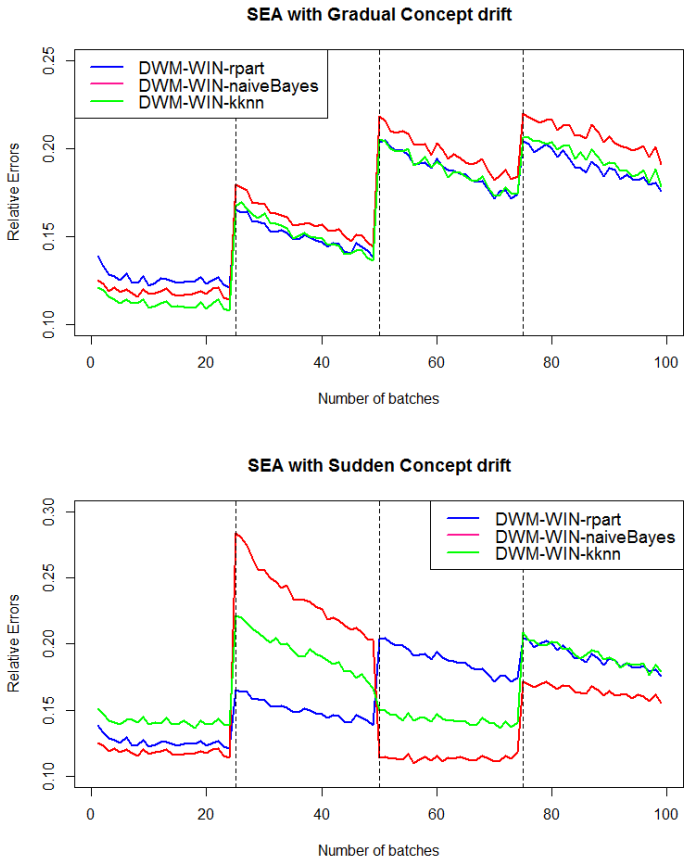

Figure 5.3: Comparative plots on sudden $(8, 7, 9.5, 9)$ and gradual $(7, 8, 9, 9.5)$ concept drift with no. of runs = 100.

methods in the three first concepts, it reaches the best error rate in the last concept. This shows that DWM-WIN-naiveBayes's performance is not stable in sudden concept drift. DWM-WIN-rpart and DWM-WIN-kknn need more time to learn the concepts. This is explained by the fact that naiveBayes performs better in detecting such concepts.

For the sudden and nonlinear concepts in SEA dataset, the DWM-WIN-kknn was surprisingly the best method coping with the problem of nonlinearity concept drift.

One way ANOVA test: According to Friedman's test we reject the hypothesis H_0 that the methods perform equally on average with an F statistic of 165.8 and a p-value $< 2.2 \cdot 10^{-16}$ when comparing linear and nonlinear drift. This result were confirmed by the two way ANOVA test as shown in Table (5.7) where the interaction between all variables is tested to see the robustness over nonlinearity issues. As it is shown, nonlinearity is highly impacted by noise and permutation. Neither the linearity nor the noise and permutation are impacted by the type of the learner. However, the linearity is highly impacted by the noise level with a p-value of 0.0085 and with permutation

it gives a p-value of 0.0078. These tests confirm that the difference performance between linearity and nonlinearity is significant because the algorithm is impacted by the nonlinearity problem. However, this does not make the fact disappear that it is robust to handle nonlinear problems and to adapt the algorithm to this issue.

5.5 Model with drift vs model without drift

Figure (5.5) shows that the model with concept drift increases the errors of classification as compared to the model without concept drift. However, this increase in the errors is absorbed by weighting and classifiers' adapting procedures of DWM-WIN. To analyze the model reaction capacity to the concept drift, we analyze the effect of modifying the base learners of the ensemble. Therefore, an analysis of the performance of DWM-WIN is performed whatever is the type of the classifier. However, as shown in Figure (5.5), naiveBayes has better capabilities of adaptation to

Table 5.4: Tukey's Test for rpart and naiveBayes for no. of batches = $(20, 50, 100)$.

	Diff	lwr	upr	p_{adj}
Learner				
naiveBayes vs rpart	-0.0254	-0.0279	-0.0228	0
Noise				
10% vs 20%	0.0871	0.0845	0.0896	0
Batch				
50-20	0.001471	-0.0022	0.00521	0.623
100-20	0.00226	-0.00147	0.006	0.326
100-50	0.000796	-0.00294	0.0045	0.87

Table 5.5: Tukey's Test.

	Diff	lwr	upr	p_{adj}
Learner				
kknn vs rpart	-0.0117	-0.016	-0.007	0
naiveBayes vs rpart	-0.024	-0.028	-0.0203	0
kknn vs naive.Bayes	-0.0128	-0.017	-0.0085	0
Noise				
10% vs 20%	0.094	0.0915	0.0973	0
Batch				
20-50	0.00054	-0.0023	0.0034	0.709

concept drift problems than rpart and kknn. This is explained by the smaller distance surfaces between the error rates of the model trained on data with concept drift and the data without concept drift. As it is shown that, naiveBayes allows quicker adaptation especially on the first three concepts. This is due to the fact that naiveBayes is a posterior probability based method facilitating the updating procedure.

Table 5.6: Nonlinear concept drift (number of batches = 20).

Permutation	naive.Bayes		kknn		rpart	
	10 %	20 %	10 %	20 %	10 %	20 %
1. (7, 8, 9, 9.5)	0.189	-	0.231	0.347	0.242	0.315
2. (7, 8, 9.5, 9)	0.263	0.378	0.221	0.336	0.231	0.336
3. (7, 9.5, 8, 9)	0.252	0.326	0.263	0.326	0.242	0.347
4. (9.5, 7, 8, 9)	0.221	0.336	0.231	0.336	0.189	0.326
5. (9.5, 7, 9, 8)	0.273	0.368	0.21	0.347	0.242	0.305
6. (7, 9.5, 9, 8)	0.1368	0.326	0.126	0.347	0.115	0.336
7. (7, 9, 9.5, 8)	0.189	0.336	0.242	0.347	0.189	0.326
8. (7, 9, 8, 9.5)	0.231	0.3052	0.2	0.294	0.221	0.326
9. (9, 7, 8, 9.5)	0.147	0.357	0.2	0.357	0.168	0.357
10. (9, 7, 9.5, 8)	0.284	0.305	0.231	0.389	0.242	0.357
11. (9, 9.5, 7, 8)	0.21	0.41	0.189	0.357	0.178	0.305
12. (9.5, 9, 7, 8)	0.252	0.4	0.273	0.336	0.294	0.357
13. (9.5, 9, 8, 7)	0.168	0.336	0.115	0.336	0.136	0.336
14. (9, 9.5, 8, 7)	0.147	0.336	0.136	0.347	0.168	0.336
15. (9, 8, 9.5, 7)	0.252	0.347	0.273	0.389	0.252	0.252
16. (9, 8, 7, 9.5)	0.252	0.378	0.263	0.315	0.221	0.389
17. (8, 9, 7, 9.5)	0.252	0.3052	0.273	0.378	0.263	0.368
18. (8, 9, 9.5, 7)	0.221	0.315	0.242	0.326	0.231	0.315
19. (8, 9.5, 9, 7)	0.221	0.368	0.136	0.315	0.21	0.389
20. (9.5, 8, 9, 7)	0.273	0.326	0.231	0.326	0.221	0.326
21. (9.5, 8, 7, 9)	0.252	0.336	0.263	0.357	0.242	0.294
22. (8, 9.5, 7, 9)	0.273	0.315	0.273	0.357	0.252	0.326
23. (8, 7, 9.5, 9)	0.315	0.34	0.263	0.294	0.252	0.368
24. (8, 7, 9, 9.5)	0.263	0.347	0.252	0.357	0.252	0.378

Table 5.7: ANOVA test of nonlinearity.

	Df	Sum Square	Mean Square	F value	Pr (>F)	
p	1	0.00487	0.00487	4.6074	0.032716	*
noise	1	0.83981	0.83981	795.0831	$< 2.2 \cdot (10^{-16})$	***
Linearity	1	0.17513	0.17513	165.8064	$< 2.2 \cdot (10^{-16})$	***
Learner	2	0.00191	0.00095	0.9018	0.407045	
vs noise	1	0.00150	0.00150	1.4209	0.234289	
vs linearity	1	0.00758	0.00758	7.1797	0.007821	**
noise vs linearity	1	0.00054	-0.0023	0.0034	0.709	
p vs learner	2	0.00034	0.00017	0.1622	0.850329	
noise vs linearity	1	0.00742	0.00742	7.0264	0.008501	**
noise vs learner	2	0.00385	0.00193	1.8247	0.163221	
linearity vs learner	2	0.00681	0.00340	3.2220	0.041395	*
Residuals	273	0.28836	0.00106			

5.6 Conclusion

In this chapter the mining of online data streams where the data distribution may change over time and the concepts may drift is discussed. The performance of ensemble methods in detecting concept drifts in data streams is analyzed. Based on the SEA dataset, we studied the impact of the performance capacity reaction of DWM-WIN in detecting the concept drift and adapting the algorithm to the drift. It has been shown that DWM-WIN has a robust capacity to adapt to different situations of concept drift with several variants of the data. It quickly adjusts itself after a concept drift detection and maintains a high performance for different drift situations with different noise levels. Further work can be carried out in studying the learners stability in the ensemble and to introduce other situations of real concept drift using generators under Massive Online Analysis (MOA) of **[Bifet et al., 2010b]**. Also a comparison with other methods such as ADaptive WINdowing (ADWIN) algorithms of **[Bifet et al., 2010a]** can be investigated.

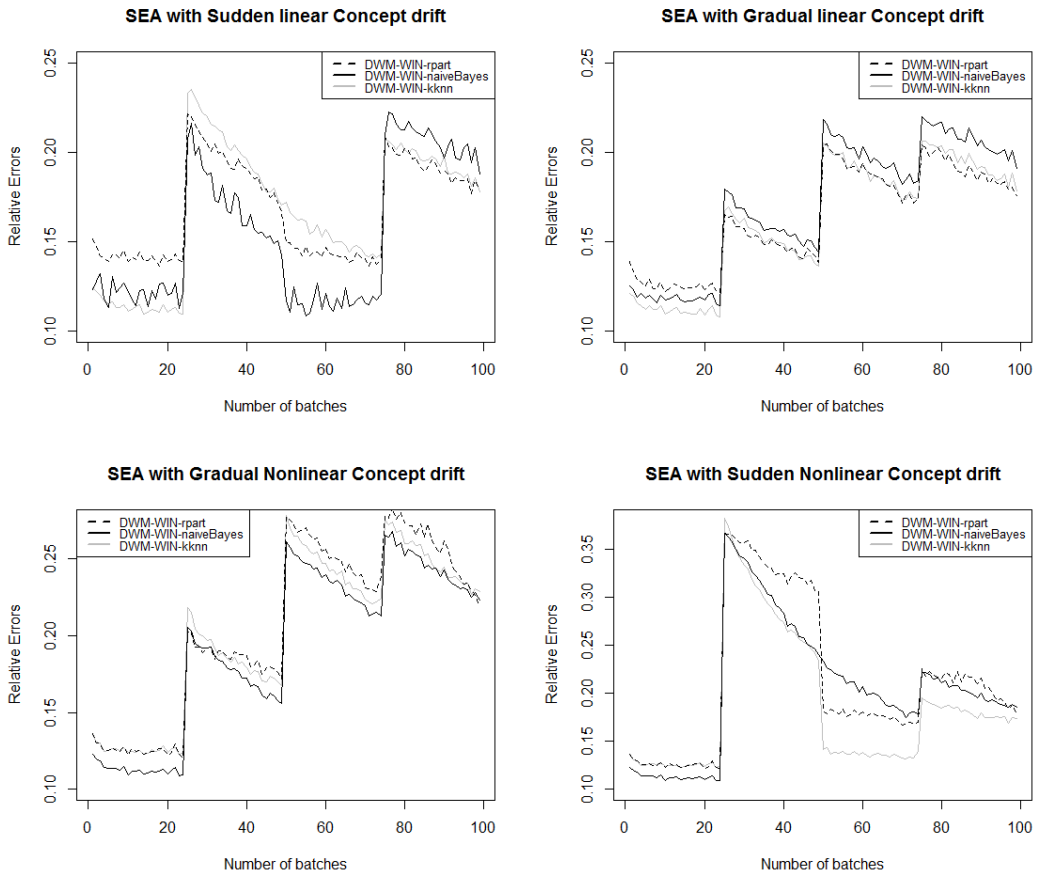

Figure 5.4: Comparison between the robustness of DWM-WIN in SEA dataset with linear Sudden, linear gradual, nonlinear sudden and nonlinear gradual concept drift where sequences used for sudden concept drift are $(7, 9.5, 8, 9)$ and $(7, 8, 9, 9.5)$ for gradual concept drift.

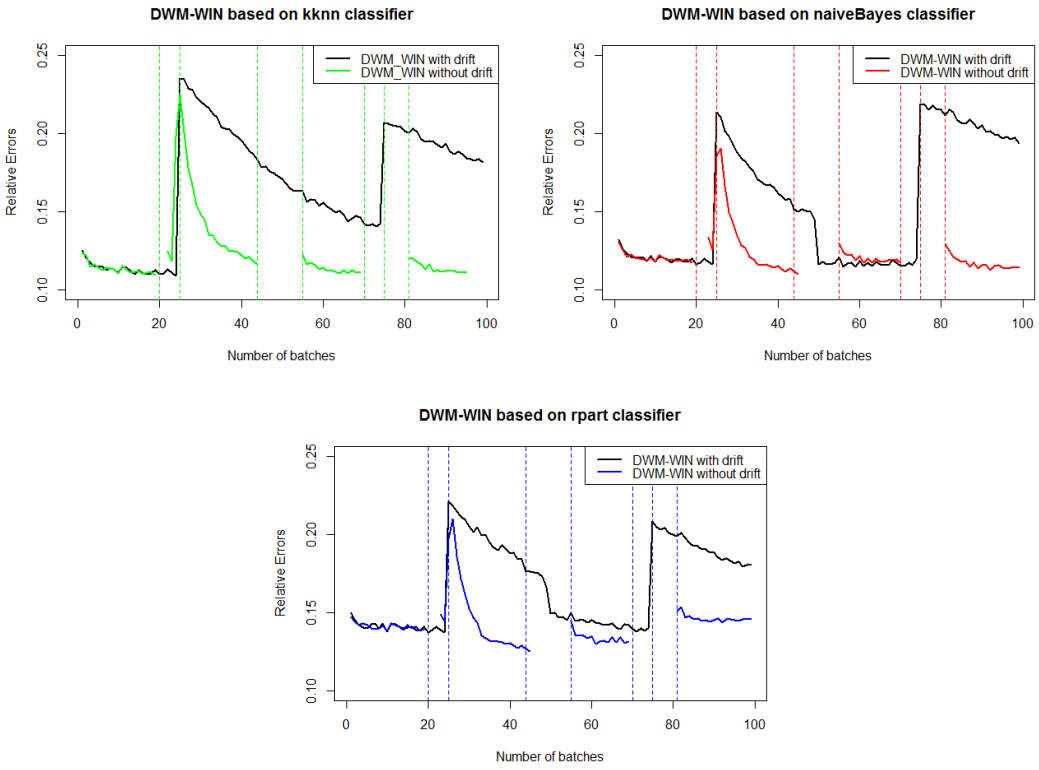

Figure 5.5: Comparison between the error rates of DWM-WIN algorithm based on different ensemble of classifiers in SEA dataset with and without drift history.

Part II

A Time Adjusting Control Chart For Monitoring DWM-WIN Classification Errors

In recent times, new challenges to monitor reliability, control performance and improve effectiveness of data growing over time, usually as a continuously and time changing stream of information, have increased impressively (**[Gama et al., 2013]**, **[Gama and Gaber, 2007]**, **[Kuncheva, 2009]** and **[Zhu et al., 2013]**). Sensor networks, security monitoring and fraud detection are examples of high speed changing data. In fact, learning in such dynamic environment requires some challenges: (1) data arrives over time and the target concept to be learned changes accordingly causing the problem of concept drift (2) changes in the distribution in nonstationary environment (3) variation of the noise level, (4) problems of nonlinearity of the data construction. Traditional methods, especially the non adaptive ones, usually assume that the dataset to be drawn is stable. However, the underlying distribution is changing over time and does not distinguish between drifting situations due to the nonstationarity of the process and the out of control targets. In the literature, several new methods in machine learning were proposed to deal with data stream and concept drift problems. The main ideas are (1) ensemble methods (**[Polikar, 2006]**, **[Kolter and Maloof, 2005b]**), (2) dynamic batch learning (**[Mejri et al., 2013]**, **[Maloof and Michalski, 2000]**, **[Maloof and Michalski, 2004]**) (3) online time adjusting (**[Garnett, 2010]** and **[Gao et al., 2007]**).

However, using data mining techniques for nonstationary environments is not enough to detect concept drift in offline processes. Even when an ensemble of classifiers is applied to detect drifting situations, the size of the batch to be learned is the key to know which instances are responsible for the out of control target. Also, deciding which classifiers are no longer contributing to the classification performance is not an easy task.

Although new methods were recently proposed in the domain of concept drift detection, there are many challenges: (1) Learning the drift which occurred in the process in order to quickly detect it (2) Monitor the classification method with dynamic control charts for concept drift learning and detection (3) Study the robustness of the classification method by swapping the target concept sequences (4) Study the robustness of the classification method via nonlinearity and different noise levels, and (5) Making performance analysis with high dimensional datasets designed for concept drift.

This part discusses two aspects. First, we propose a Time Adjusting Control Limit (TACL) chart designed to monitor concept drifting data streams in nonstationary environments. In fact, we propose a new formula of charting statistics and a Control Limit (CL) that is adjusted during the learning process.

Second, we propose an advanced TACL chart called Two Stage TACL (TS-TACL) chart. We apply our proposed method to a high dimensional space real dataset with concept drift and we propose many variants of the data. Finally, we compare our results to other time varying control charts.

Thus, Part II is outlined as follows: Chapter 6 explains the two new TACLs strategies. Also, empirical results and comparison of TACL and TS-TACL charts with the most recent heuristics are discussed. Chapter 7 details the assessment of the proposed method on many variants of the SEA dataset and other data with concept drift. We study the robustness of the proposed method based on several performance measures.

Chapter 6

Time Varying Control Chart for Non-stationary Data Streams Process

This chapter is organized as follows. Sections 1 and 3 provide a detailed explanation of two proposed time adjusting control Limit (TACL) and Two Stage TACL (TS-TACL) charts respectively. Section 2 presents the experimental results and comparison of the updating chart model with charts with fixed Control Limits (CLs).

6.1 Design of a TACL chart

In traditional control charts, Upper Control Limits (UCL) and Lower Control Limits (LCL) are computed based on the underlying distribution of the monitoring statistic. If an observation falls outside the Control Limits (CLs), the process is considered to be out of control. However, the distribution of a monitoring statistic is often unknown specially in the case of data arriving over time. Indeed, the target concept to be learned may change from training to testing causing the problem of concept drift. For this, change in the distribution has to be taken into account during the process monitoring in order to distinguish between real outliers and drifting concepts. This motivates the development of an appropriate new procedure to establish the new CLs. One way to do so is to adjust the CLs each time an adjustment condition is satisfied. The proposed time adjusting CLs chart consists of three steps. A training step where we initialize parameters, a testing phase based on the parameters already calculated in the first step to decide whether to accept or to reject the null hypothesis that the data is in control. Finally, a decisive step to decide about the out of control observation. During the training, the dataset is assumed to be stationary and the shift generated by the data distribution occurs in the testing phase. Thus, when a data point falls outside the CLs during the testing phase, the null hypothesis is rejected and a shift is detected. It is important to note that our control chart is proposed to detect two types of nonstationarity. First, we assume that the input distribution changes between training and testing which is defined as the

"covariate shift". Second we assume a change in the target concept is to be learned i.e. a change in the dependence of y on X which is defined as "concept shift" or "concept drift".

The proposed TACL chart can be described in the three following steps:

6.1.1 First step

Generate n batches and compute z_1, z_2, z_3, ..., z_n, where z_i for $i = 1, \ldots, n$, are independent random variables normally distributed with mean μ_n and variance σ_n^2 where z_i represents the monitored data, and n is sample size used at each time interval. To estimate the mean and the variance of z_i, we use the following formulas:

$$\widehat{\mu_n} = \bar{z} \tag{6.1}$$

and

$$\widehat{\sigma_n^2} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2. \tag{6.2}$$

6.1.2 Second step

CLs are fixed in Phase I and they are computed based on L sigma limits, where L is a numeric value specifying the number of sigmas to use for computing control limits.

Phase I: Consider z_1 , z_2, z_3, ..., z_n, as the observations of Phase I, then time-adjusting UCLs and LCLs in Phase I are respectively computed as follows:

$$UCL_n = \bar{z} + L\widehat{\sigma}_n \tag{6.3}$$

and,

$$LCL_n = \bar{z} - L\widehat{\sigma}_n. \tag{6.4}$$

For this study, we assume that the standard deviation σ_n is constant and we have only a shift in the mean.

Phase II:

• Check the adjustment condition

If a percentile of the dataset size denoted P is detected to be out of control for k new observations then the adjustment condition is satisfied.

We denote T the number of out of control observations that should be detected before adjusting the CLs. T is computed as follows:

$$T = \lceil P \cdot k \rceil, \tag{6.5}$$

where P is the percentage required to satisfy the adjustment condition. For example, if the

process size is k = 400 instances, then the 5^{th} percentile of the dataset size is 20. Then, after 20 observations detected out of control, the CLs has to be adapted to the shift occurred in the process.

- **If the adjustment condition is satisfied, then adjust**

Decide whether z_{n+1} ,z_{n+2}, z_{n+3}, ... ,z_{n+k} are in or out of control based on the fact that if z_i exceeds the CLs, so the instance is out of control. Then, if the adjustment condition is satisfied, then we adjust the CL using:

$$x_{n+k} = \frac{UCL_{n+k} + LCL_{n+k}}{2} \tag{6.6}$$

which is the mean of the previous UCL and LCL, by defining

$$UCL_{n+k+1} = \lambda x_{n+k} + (1-\lambda) z_{n+k} + L\widehat{\sigma}_n, \tag{6.7}$$

and

$$LCL_{n+k+1} = \lambda x_{n+k} + (1-\lambda) z_{n+k} - L\widehat{\sigma}_n \tag{6.8}$$

where x_t represents the historical data, z_t is the new data and σ_n is a constant. The starting values of LCL and UCL are given by Phase I, λ is a constant between 0 and 1, representing the weight assigned to the current observation. The value of λ depends on the user's preference to choose how much he/she wants to consider the recent data or the historical data in the computation of the new CLs. If λ is larger than 0.5, this means that the historical data are more representative in the CLs computation and these values are useful for forecasting. However, if λ is less than 0.5, this means that we take into account the recent data more than the historical data in the update of the CLs. In the experiments, we choose a typical value of λ = 0.2 to be close to the value of λ chosen by **[Steiner, 1999]** in EWMA with time varying CLs chart which is between 0.05 and 0.25. Stored variables in the updating algorithm are only: UCLs, LCLs and $\widehat{\sigma}_n$.

6.1.3 Third step

For monitoring Phase II observations, a batch is declared out of control if z_i exceeds the CLs, and then, the process is out of control. Algorithm (6.1) explains in more details the different steps of TACL chart.

Before applying our proposed method to a binomial distribution with probability 0.2 and 400 observations with shift in the mean, we first illustrate the behavior of a standard control chart with fixed CL where the charting statistics are the observations themselves. Figure (6.1) illustrates how a large number of observations are considered as out of control observations after the shift occurred in the process. However this way of considering the control is not consistent with suddenly occurring shift especially for online data streams arriving in forms of batches. Thus, these CLs should be "time adjusting". This is what we investigate in the following experiments.

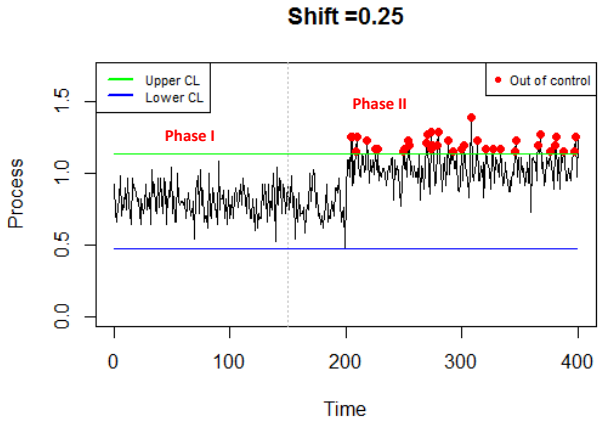

Figure 6.1: Control chart with fixed control limits for datasets with concept drift (shift = 0.25, L =3).

A simulated dataset is used with different sizes of shift in the mean. At first, a data set with Binomial distribution with a probability equal to 0.2 is used to evaluate the performance of the control chart in a general case. The reason of that the proposed TACL chart will be applied later to monitor the misclassification error rates. So a binomial distribution with a probability p is considered as a first representation of this misclassification error rates. Indeed, we generate a small shift in the mean in order to vary the distribution. Figure (6.2) illustrates the three steps described in previous sections. In fact, Phase I and Phase II are separated using a vertical line at observation 150. Then, when a shift is detected in time point 200 to the probability $p = 0.2$ and UCLs and the LCLs are adjusted. In observation 300 another shift is detected so we readjust the CLs in the same manner used for the first shift and so on.

This procedure is repeated until no out of control is detected. For each new data, a shift might be detected and learned. All the information about the shift history is memorized and considered in the adaptation of the monitoring systems.

6.2 Performance versus fixed chart model

The simulated dataset used in this chapter contains 400 data points and the non stationarity happens in the middle. The distribution changes from a Bin(200, 0.2) to a Bin(200, 0.2+ δ) where Bin(n, p) indicates the binomial distribution with $\mu = np$ and $\sigma = \sqrt{np(1-p)}$. We first compare the TACL chart with a control chart based on fixed upper and lower CLs. To do so, we compare the updating model illustrated in Figure (6.2) and the fixed CLs model illustrated in Figure (6.1) in terms of 100 runs of the number of observation to Detection (PD). In the next subsections, the computation and the analysis based on these measures are detailed.

Algorithm 6.1: Algorithm TACL.

Input : L: a parameter specifying the number of sigmas to use for computing control limits.

Consider an online data stream process of size n for the training data

Collect the new coming data batch by batch in Phase II and monitor as follows IF (An out of control satisfies the adjustment condition) THEN (Note the point out of control and prepare for an updating procedure) ELSE(Join the new arriving data and continue the monitoring)

Output: Instances or Batches out of control

Training Phase

Select n training data denoted Z(i) for i = 1, …, n

Compute the mean $\overline{Z} = \frac{1}{n}\sum_{i=1}^{n} Z_i$

Compute the variance $\widehat{\sigma_n^2} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \overline{z})^2$

Compute $UCL_0 = \overline{Z}$+$L \cdot \widehat{\sigma_n}$ *and* $LCL_0 = \overline{Z}$-$L \cdot \widehat{\sigma_n}$

Testing Phase

for *each new data* Z_{n+K} *for* $K = 1, 2, \ldots k-1$ **do**

  Check on condition adjustment

  if *Condition adjustment is satisfied* **then**

    Compute $X_{n+K} = \frac{UCL_{n+k}+LCL_{n+k}}{2}$

    Update $UCL_{n+K+1} = \lambda \cdot X_{n+K}$ +(1-λ) $\cdot Z_{n+K} + L \cdot \widehat{\sigma_n}$

    Update $LCL_{n+K+1} = \lambda \cdot X_{n+K}$ +(1-λ) $\cdot Z_{n+K}$ - $L \cdot \widehat{\sigma_n}$

    if $LCL_{n+K+1} < Z_{n+K} < UCL_{n+K+1}$ **then**

      Continue monitoring the new data

    end

    else

      An alarm is signaled in Z_{n+K}

    end

  end

  else

    Continue monitoring the new data

    $UCL_{n+K+1} = UCL_{n+K}$ *and* $LCL_{n+K+1} = LCL_{n+K}$

  end

end

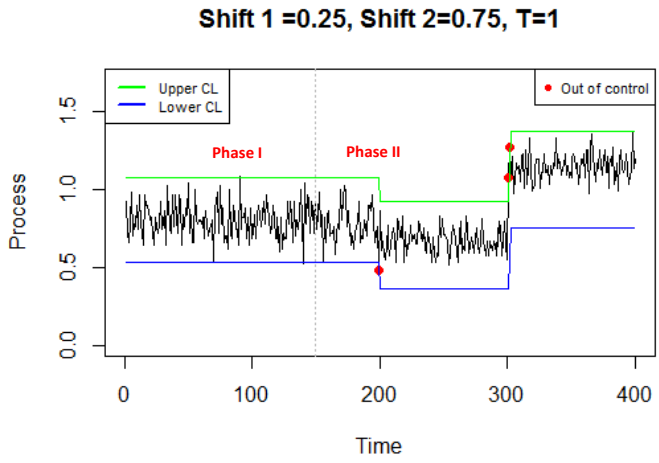

Figure 6.2: TACL chart based on a binomial distribution, Bin(400, 0.2) with shift $\delta = 0.25$ and 0.75 respectively, $L = 3$, parameter of the adjustment condition $P = 0.25\%$ and $T = 1$.

6.2.1 Probability of detection

The Probability of detection is defined by the following formula:

$$PD = \frac{\text{Number of Out of Control Observations}}{N} \tag{6.9}$$

where N is the total number of observations.

Figure (6.3) provides results over 100 runs of TACL with updating CLs compared to TACL with Fixed CLs. Results are representative for the capacity of the proposed control chart to detect the out of control targets for different shift ranges $\delta \in \{0.25, 0.5, 0.75, 1, 1.5\}$ for T =1.

TACL chart is deemed to be more sensitive to detect a correct shift than the no updating chart. Updating the CLs after the shift has occurred in the data involves a significant effect on detection of the different concept drift situations which can happen during the process. Therefore, as expected, TACL based on updated CLs is more robust to reduce the false alarms than the fixed CLs charts. In fact, the ability of the TACL to reduce the number of false alarms compared to fixed CLs model is also explained by the parameter T. To study the effect of T on the robustness of our proposed method to reduce false alarms, we study the effect of this parameter on the denoted measure.

6.2.2 Impact of the waiting time parameter T

Results presented in the previous sections assumed that the parameter of the condition adjustment is $T = 1$. The question of which parameter value has to be chosen for the concept drift detection to be quick and efficient in nonstationary environments is an important task. Thus, the impact of T = 1, T = 10 and T = 20 is introduced in this subsection. Analysis of the detection capability based

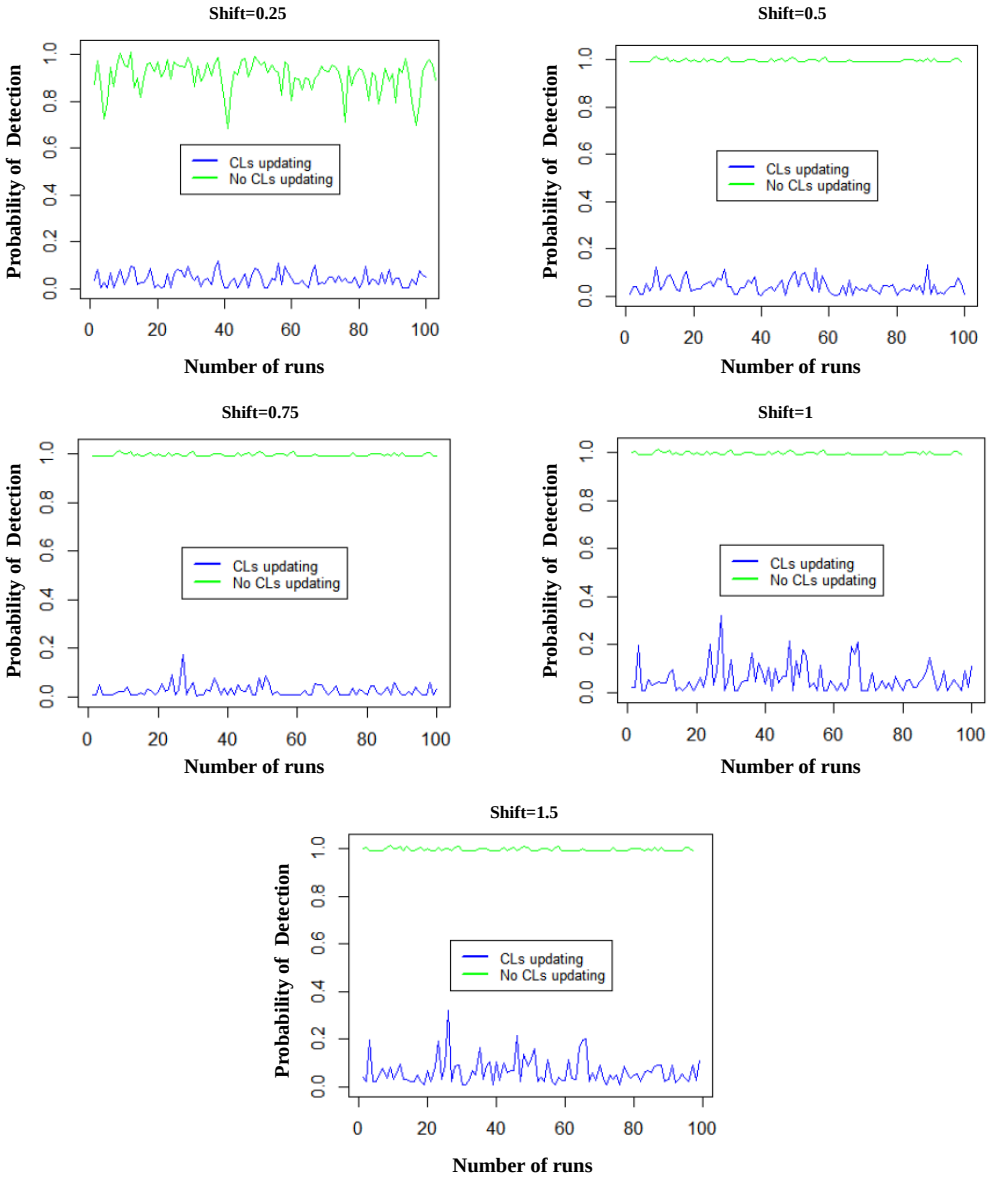

Figure 6.3: Comparison of the probability of detection of TACl and TACL with fixed CLs based on different shifts in the mean and the variance of Bin(400, 0.2).

on these values is presented in Figure (6.4) which illustrates the average number of data points to detection. Above all, we notice that $T = 1$ provides always the highest mean number of data points detected compared to larger values of T.

This is due to the fact that the TACL chart updated itself after each out of control detection and hence detects more out of control observations and does more adaptation. One of the benefits of this successive adjustments is to speed up the concept drift detection, but one drawback is that

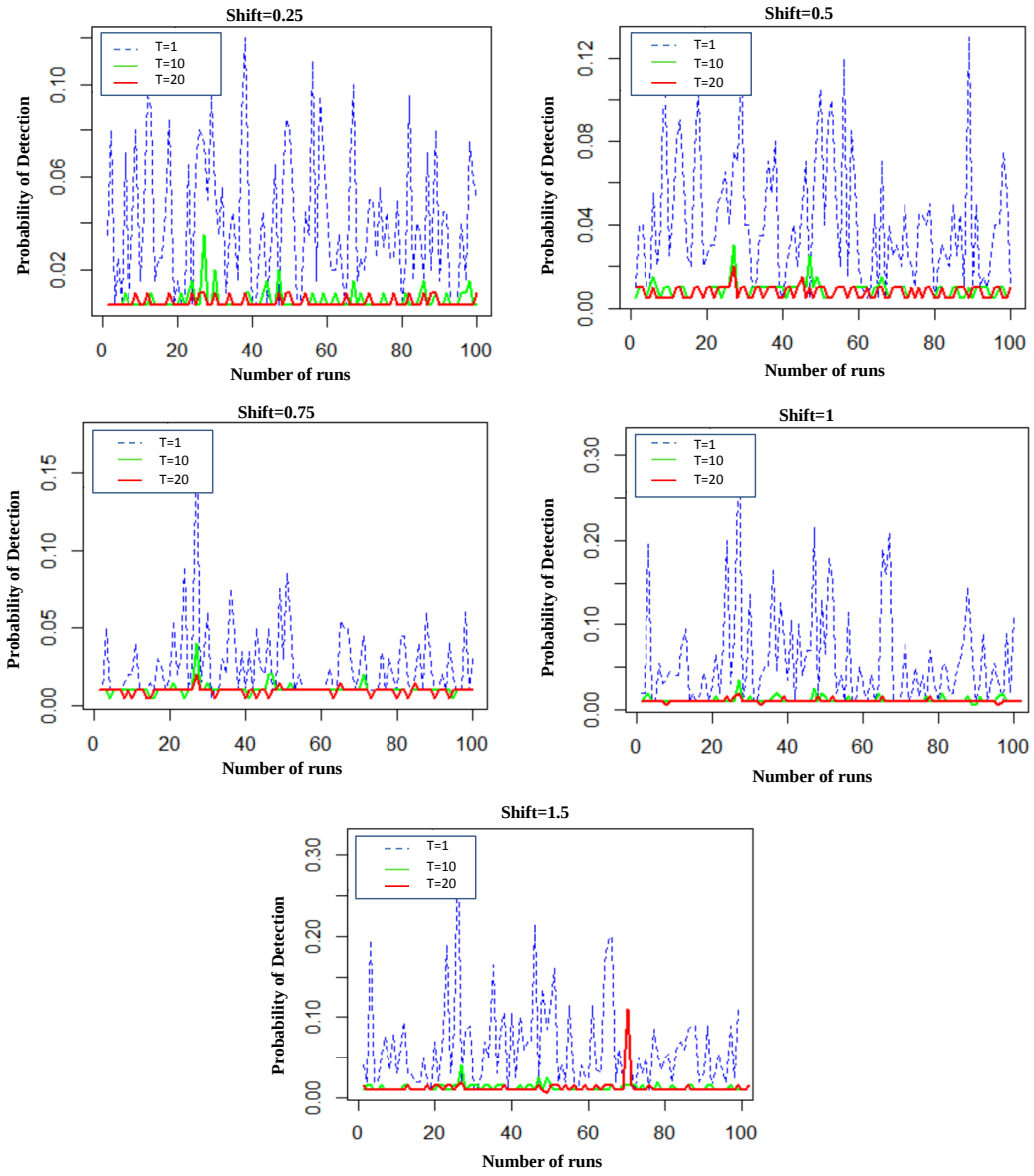

Figure 6.4: Illustration of the impact of the value of T on the number of observations to detection for 100 runs.

this behavior could lead to a high false alarm rate. In the case of $T = 10$ and $T = 20$, results are close to each other. The probability of detection decreases when the adjustment condition requires a high number of outliers in order to be satisfied like $T = 10$ or $T = 20$. This parameter depends on the size of the data and on the expected shift occurrence.

In the next chapter, we analyze the proposed method in terms of FPs, FNs among other measures of performance.

Although this shift detection strategy is well suited for fast data stream problems, it may generate extreme number of false alarms, which represent an obstruction in real world processing. For reducing the number of false alarms, a two stage time adjusting shift identification method is proposed. This method should enhance the proposed TACL chart by presenting a full general Algorithm for concept drift detection under non stationary processes.

6.3 Design of Two-Stage TACL chart

We propose a TS-TACL chart as an improvement of a TACL chart. In Stage I, we use a TACL chart as an online method for shift detection in a non stationary data stream process. In Stage II, we use a statistical test to confirm the correctness of the shift detected in Stage I. Our aim when using a Stage II test is to reduce the number of false detections. The pseudo code of TS-TACL chart is given in Algorithm (6.2). This technique was used by **[Raza et al., 2015]** for reducing the false alarms of the SD-EWMA chart proposed by **[Raza et al., 2013]**. The new proposed method is described in the following.

6.3.1 Stage I

As explained previously, TACL works through three different steps and two phases: training and testing. In phase I, step 1 is applied to compute the UCL_0 and the LCL_0 whereas in Phase II, Step 2 and 3 are applied to check if $LCL_i \leq Z_i \leq UCL_i$ and an out of control is detected in Stage I. After that, this shift is validated in Stage II based on a non parametric test of non stationarity. The aim of Stage II is to decrease the false detection rate in favor of the true detection rate.

6.3.2 Stage II

In order to validate the robustness of TACL chart in detecting a true positive alarms during Stage I, we apply the two-sample Kolmogrov-Smirnov test to the observations detected as out of control in stage I to validate the shift already detected. This test was also used by **[Raza et al., 2015]** when they needed to validate the shift detected by the Two Stage Shift Detection EWMA (TSSD-EWMA) chart. The test is defined as follows:

$$\begin{cases} H_0 : the\,data\,in\,the\,subsample\,are\,stationary. \\ H_1 : the\,data\,in\,the\,subsample\,are\,nonstationary. \end{cases} \tag{6.10}$$

To apply this test, the dataset containing the detected out of control is divided into two different subsequences n_1 and n_2 and then the two sample Kolmogrov-Smirnov test is applied. The statistic of the test is given by the following formulas:

$$D_{n_1,n_2} = sup \mid F_{1,n_1}(x) - F_{2,n_2}(x) \mid, \tag{6.11}$$

Algorithm 6.2: Algorithm TS-TACL.

Input : L: a parameter specifying the number of sigmas to use for computing the CLs.
Consider an online data stream process of size n for the training data ;
Collect the new coming data batch by batch in Phase II and monitor as follows;
IF (An out of control satisfies the adjustment condition);
THEN (Note the point out of control and prepare for an updating procedure);
ELSE(Join the new arriving data and continue the monitoring)

Output: Instances or Batches out of control

Training Phase `// From line 2 to 22, refer to TACL algorithm.`
Select n training data denoted $Z(i)$ for i = 1, ..., n
Compute the mean $\overline{Z} = \frac{1}{n}\sum_{i=1}^{n} Z_i$ *and the variance* $\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2$
Compute $UCL_0 = \overline{Z} + L \cdot \sigma_n$ *and* $LCL_0 = \overline{Z}\text{-}L \cdot \sigma_n$
Testing Phase
for *each new data* Z_{n+K} *for* $K = 1, 2, \ldots k-1$ **do**
    Check on condition adjustment
    if *Condition adjustment is satisfied* **then**
        Compute $X_{n+K} = \frac{UCL_{n+k}+LCL_{n+k}}{2}$
        Update $CL_{n+K+1} = \lambda \cdot X_{n+K} + (1\text{-}\lambda) \cdot Z_{n+K} \pm L \cdot \sigma_n$
        if $LCL_{n+K+1} < Z_{n+K} < UCL_{n+K+1}$ **then**
            Continue monitoring the new data
        end
        else
            An alarm is signaled in Z_{n+K}
        end
    end
    else
        Continue monitoring the new data
        $UCL_{n+K+1} = UCL_{n+K}$ *and* $LCL_{n+K+1} = LCL_{n+K}$
    end
end
Stage II
For each alarm Z_j for j = n, ..., n+K. Partition the data around time i into two samples. First sample instances from i-(t-1) to i and second sample from time i+1 to (i+t)
Apply Kolmogorov Smirnov (KS) test to the divided data
if *KS test statistic > critical value of the F distribution* **then**
    Reject the null hypothesis that the two subsamples belong to the same distribution:
    The signal detected in Stage I is true.
end
else
    Signal detected in Stage I is false and discarded.
end

where F_{1,n_1} and F_{1,n_2} are the empirical distribution functions of the first and the second sample respectively. Sup represents the supremum of a subset S of a partially ordered set T. It is defined as the value which is larger than or equal to all elements of the dataset T. For a given level of α, H_0 is rejected if:

$$D_{n_1,n_2} > c(\alpha)\sqrt{\frac{n_1 n_2}{n_1 + n_2}} \tag{6.12}$$

where c is taken from the table of critical values for the two-sample Kolmogorov-Smirnov test.

Algorithm (6.2) details the different steps of TS-TACL chart.

6.4 Conclusion

This chapter presented novel methods for concept drift and covariate shift detection based on a time adjusting two stage charts to monitor non stationary processes. The first proposed chart is based on adjusting the CLs each time a condition adjustment is satisfied. The second proposed chart is based on a two stage model to reduce the false detection alarms of the first proposal. In the first stage, the TACL chart is used to detect the shift occurred during the process. In the second stage, a Kolmogorov Smirnov test is applied to validate the out of control detected in Stage I. The adaptive chart model has been found to be more effective in reducing the number of observations to detection when compared to fixed CLs model. This is is due to its efficiency to distinguish between real shifts and outliers caused by the nonstationary of the process. The proposed charts require that an adaptive measure of performance should be used. In the next chapter, different performance measures are used to evaluate and compare the proposed methods.

Chapter 7

Assessment of TACL and TS-TACL on Simulated Datasets

In this chapter, we present an assessment of TACL and TS-TACL chart on several simulated datasets with and without concept drift. We use SEA dataset, one of the most widely used to study the problem of concept drift. We provide an experimental comparison of our proposed methods with Shift Detection EWMA (SD-EWMA) chart of **[Raza et al., 2013]** and with Two Stage SD-EWMA (TSSD-EWMA) chart of **[Raza et al., 2015]**. This chapter is outlined as follows: Section 1 explains our methodology in conducting our experiments. Section 2 discusses the assessment of TACL and and TS-TACL on a synthetic dataset (D1). Section 3 conducts the application on two variants of Jumping dataset (D2) used by **[Liu et al., 2013]**. Section 4 is dedicated to study the robustness of our proposals in SEA dataset. Section 5 discusses the results and Section 6 gives the results of the performance of the different algorithms using ANOVA and Tukey's test of significance.

7.1 Methodology

In this section, we present the parameters used in the experiments, we give information about the dataset and we analyze the results. The proposed algorithms were implemented using R packages and the results were performed on a computer using an Intel ® core (TM) $i3-2312$ M CPU @ 2.10 GHz with a RAM of 8 GB. We use the following parameters: for TACL chart, $\lambda = 0.5$ and $L = 3$, for TS-TACL the values of λ were chosen as 0.5 and 0.2 and the value of L as 2, 2.5 and 3. The θ values used for SDEWMA and TS-SDEWMA are 0.01, 0.05 and 0.1, the λ is fixed to 0.4 as proposed by **[Raza et al., 2015]**.

On each dataset, the CCs are evaluated based on several evaluation metrics. We define them in the following:

- **True Positive (TP)**: It happens when a test signals an alarm in the process when it is not there

(true detection).
- **True Negative (TN)**: It happens when a test signals an alarm when it is there.
- **False Positive (FP)**: It happens when a test signals an alarm in the process when it is not there (false detection): type I errors.
- **False Negative (FN)**: It happens when a test does not signal an alarm when it is there (misdetection): type II errors.
- **Recall**: True Positive Rate.
- **Accuracy**: is computed as follows:

$$Accuracy = \frac{TP+TN}{N}, \tag{7.1}$$

where N is the total of the population.
- **Precision**: is the positive predicted rate and it is computed as follows:

$$Precision = \frac{TP}{TP+FP} \tag{7.2}$$

- **F-measure**: Is the harmonic mean of precision, and Recall and is computed as follows:

$$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{7.3}$$

The reason that we use F-measure together with the accuracy is that displaying TACL and TS-TACL results based on only Accuracy would be misleading because of the number of failed detection. Therefore, if all concept drift situations are detected, Accuracy measure would be competitive because of the high number of TP and TN observations. Thus, neglecting the false detection can lead to a misunderstanding of the results. Hence, we take into account the false detection by considering FP observations as well as FN ones by computing the F-Score measure **[Rijsbergen, 1979]**.

7.2 D1: Dataset with abrupt change

7.2.1 Dataset description

To validate the effectiveness of TACL and TS-TACL we first perform results on a synthetic dataset with abrupt shift. The process consists of 2000 observations and follows a normal distribution with mean μ and standard deviation σ where the non stationarity occurs in the middle. During the first 1000 data points, the process follows a $N(\mu_1, \sigma^2)$, then the process is shifted to $N(\mu_2, \sigma^2)$. In this study parameters used for TACL are λ = 0.5 and L = 3 whereas the values of λ in TS-TACL chart are set 0.5 an to 0.2 and the values of L are 2, 2.5 and 3. The θ values used for SD-EWMA and TSSD-EWMA are 0.01, 0.05 and 0.1. An illustration of TACL and SD-EWMA charts in the synthetic dataset is presented in Figures (7.1) and (7.2). The CLs vary sequentially with the dataset changing behavior in SD-EWMA. However, the CLs of TACL chart vary based on a specific adjustment rule.

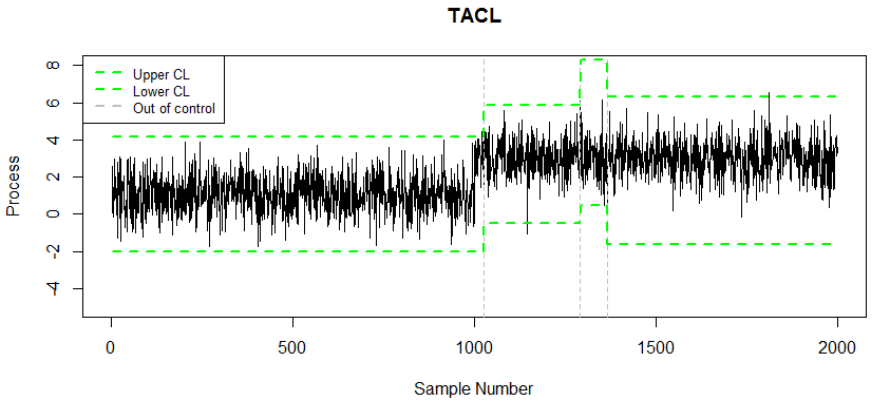

Figure 7.1: Illustration of TACL chart on (D1): dataset with abrupt change, the shifts are detected from observation 1000, TACL detects less FPs than SD-EWMA.

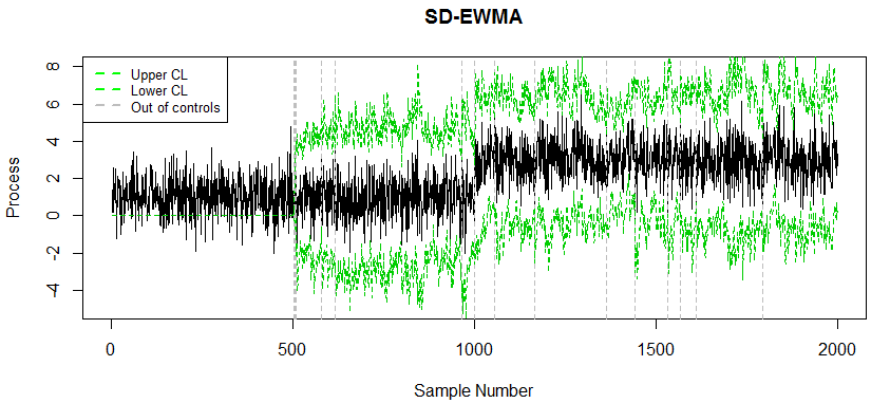

Figure 7.2: Illustration of SD-EWMA chart on (D1): dataset with abrupt change, the shifts are detected from observation 1000 on, TACL detects less FPs than SD-EWMA.

7.2.2 Results analysis

We assume that in Phase I, the process is in control and that the shift occurs in Phase II. As seen in Table (7.1) TACL always stands out as the main chart in detecting the shift compared to the different variants of SD-EWMA chart. These values show that the new approach achieves better results than the other CCs on synthetic dataset. For the TS charts, the proposed TS-TACL chart improves TACL chart's performance by decreasing the FP rate and increasing the Accuracy. The reason for that is that when applying the two sample Kolmogrov-Sminov test on the out of control observations detected by TACL in Stage I, we confirm the correctness of the detected alarms and if the null hypothesis is rejected, the detected alarm is considered false. Indeed, the number of signaled alarms when they are not correct is reduced in TS-TACL chart. Then when the FP rate decreases,

the Accuracy increases. FN and Recall rates are optimal for TACL as well as TS-TACL. Figure (7.3) reports results over 100 runs of the different performance measures. The behavior of TSSD-EWMA chart in outperforming SD-EWMA approximates the behavior obtained by TS-TACL and TACL chart in terms of reducing the FP rate and increasing the Accuracy. As it is shown in Table (7.1), TS-TACL chart always reported the best FN, FP, Accuracy and Recall values. However, although TS-SDEWMA reduces the FP and the FN rate compared to SD-EWMA, the TS-TACL chart still has the highest Accuracy and Recall measures in all cases and the lowest FP measures. So, in general, TACL and TS-TACL are more able to handle abrupt change. Moreover, TS-TACL is far superior to the other charts in detecting abrupt change under nonstationary processes by reducing the FP rates.

Table 7.1: TACL and TS-TACL chart compariosn with SD-EWMA and TSSD-EWMA based on D1 dataset.

Algorithms	Param	FP	FN	Recall	Acc
TACL	($\lambda = 0.5, L = 3$)	0.0037	0	1	0.996
SD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 2$)	0.0455	0	1	0.954
SD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 2.5$)	0.0160	0.005	0.995	0.983
SD-EWMA	($\lambda = 0.5, \theta = 0.01, L = 3$)	0.0100	0.315	0.717	0.989
SD-EWMA	($\lambda = 0.5, \theta = 0.05, L = 3$)	0.0054	0.265	0.735	0.994
SD-EWMA	($\lambda = 0.5, \theta = 0.01, L = 3$)	0.0160	0.225	0.775	0.993
TS-TACL	($\lambda = 0.2, L = 2.5$)	0.00034	0	1	0.999
TS-TACL	($\lambda = 0.5, L = 3$)	0.00034	0	1	0.999
TS-TACL	($\lambda = 0.2, L = 3$)	0.00034	0	1	0.999
TSSD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 2$)	0.00210	0.025	0.975	0.997
TSSD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 2.5$)	0.00070	0.150	0.850	0.999
TSSD-EWMA	($\lambda = 0.5, \theta = 0.01, L = 3$)	0.00034	0.155	0.845	0.999
TSSD-EWMA	($\lambda = 0.5, \theta = 0.05, L = 3$)	0.00023	0.145	0.855	0.999
TSSD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 3$)	0.00020	0.175	0.825	0.999

Figure (7.3) reports results over 100 runs of the different methods in terms of FP, FN, Recall and Accuracy. An overall performance of the two methods is achieved compared to SD-EWMA and TS-SDEWMA. The behavior of TSSD-EWMA in improving the results of SD-EWMA in terms of FP and Accuracy approximates the behavior obtained by TS-TACL and TACL chart. As shown in Table (7.1), TS-TACL chart always reported results as the best FN, FP, Accuracy and Recall values. The reason of that is that TACL chart adjusts the CLs only when it is necessary

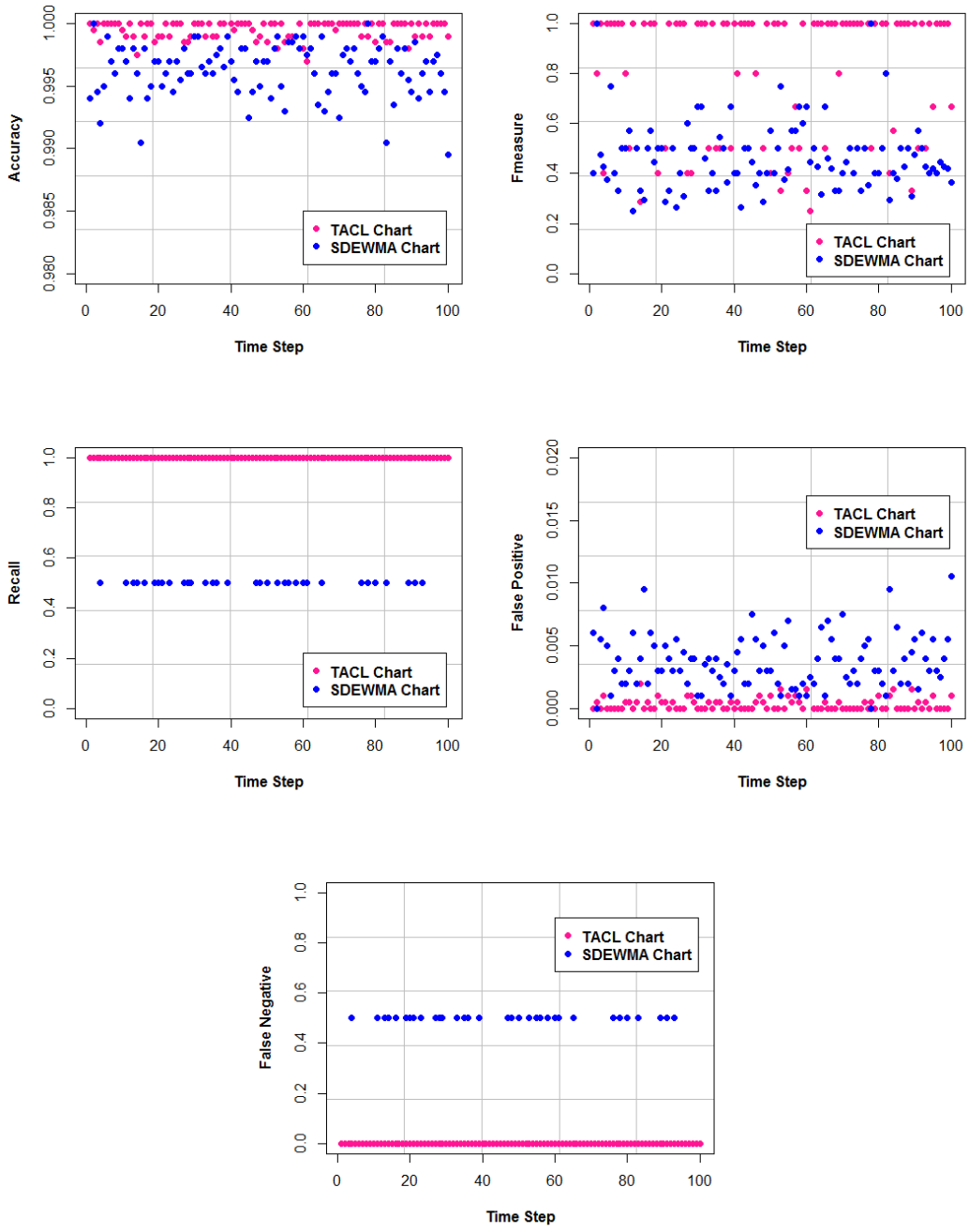

Figure 7.3: Comparison of TACL and SDEWMA on D1 dataset.

based on a specific adjustment condition. However, SD-EWMA adjusts the statistics and the CLs each time with any new data. Figure (7.4) illustrates the performance of TS-TACL and TSSD-EWMA in terms of Accuracy.

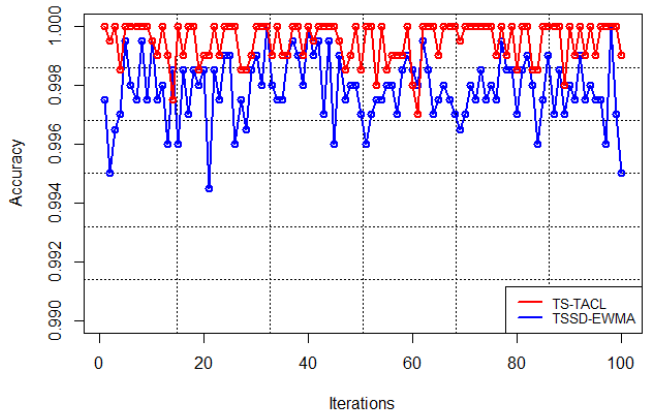

Figure 7.4: Illustration of TS-TACL chart and TS-SDEWMA chart on D1: dataset with abrupt change, the shifts are detected from observation 1000 on, TACL has better Accuracy than SD-EWMA.

7.3 D2: Jumping mean dataset

7.3.1 Dataset description

Jumping mean is an artificial time-series dataset given in **[Liu et al., 2013]** based on the work of **[Takeuchi and Yamanishi, 2006]**. It consists of 5000 samples (i.e., t = 1, ..., 5000) based on the following formulas:

$$Z(t) = 0.6 \cdot Z(t-1) - 0.5 \cdot Z(t-2) + \varepsilon_t, \tag{7.4}$$

where ε_t is a Gaussian noise: N(μ, σ^2 = 1.5). The initial values are set to be $Z(1) = Z(2) = 0$. At each time step, a change point is inserted by setting the noise mean μ at time t as:

$$\mu_N = \begin{cases} 0 & \text{if } N = 1 \\ \mu_N + \frac{N}{16} & \text{if } N = 2, \ldots, 49 \end{cases} \tag{7.5}$$

where N is a natural number and μ_N is valid for : 100(N-1) +1 $\leq t \leq$ 100 N.

Figures (7.5) and (7.6) show an example of the shift detection with TACL and SD-EWMA charts. To perform the covariate shift detection in D_2, we consider 500 observations for training and 4500 for testing. Performance of the results of the TACL and SDEWMA chart in D_2 is given in Table (7.2).

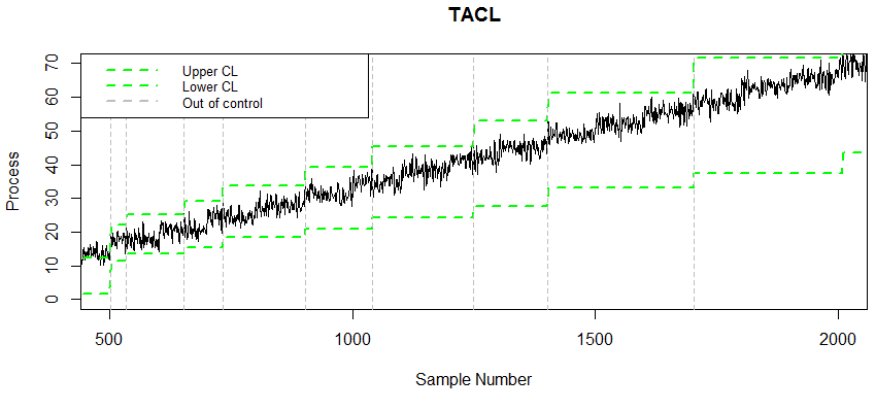

Figure 7.5: Illustration of TACL chart in D_2: Jumping mean dataset.

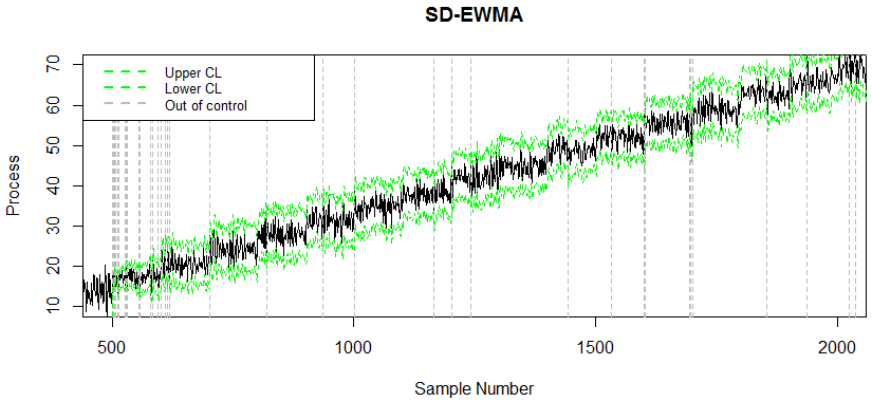

Figure 7.6: Illustration of SD-EWMA chart on D_2: Jumping mean dataset.

7.3.2 Result analysis

We first compare TACL's Accuracy, FP, FN and recall rates to those of the SD-EWMA chart. Table (7.2) shows that the TACL chart achieves better FP and accuracy than the SD-EWMA when detecting the covariate shift of the jumping mean dataset but the worst Recall and FN. The reason for the worst FN rate is that the TACL chart in some cases does not detect some outliers which were correctly identified by SD-EWMA chart. This is due to the fact that the UCLs and LCLs of SD-EWMA are computed based on one ahead prediction which also facilitates the change point detection ability in some cases. In fact, when many shift points exist, the SD-EWMA is more able to detect them. This is what explains that the FNs based on the misdetection points are smaller in SDEWMA than in TACL. However, the UCLs and the LCLs of TACL chart are based on a linear combination of the historical observations and the new coming one which highlights the ability of

TACL chart to identify correct alarms and accordingly to reduce the false alarms (and reduce the FP rate).

In this regard, the ability to identify a situation correctly decreases when the FN are high which explains the low values of recall in TACL compared to SD-EWMA chart. In general when monitoring data with high numbers of change points like D_2, it is better to use TACL if the user is more interested in reducing the number of alarms (FPs). However, when the user's interest is to reduce the number of non-detected signals when they are there (FN), the SD-EWMA is better.

We concentrate now on comparing TACL and TS-TACL charts. In fact, the two stage structure outperforms TACL chart based on all evaluation metrics. This behavior is expected because the Kolmogrov Smirnov test validates the correctness of the shift detected in Stage I.

Table (7.2) shows that an improvement over the different performance measures is achieved by TACL chart after adding the second stage. It increases the Recall of TACL by 0.319 and decreases the FN rate by 0.305 for $\lambda = 0.5$ and $L = 3$.

Now, we compare TS-TACL and TSSD-EWMA charts. First, TS-TACL is more robust than TSSD-EWMA to detect the shifts by achieving a certain stability when having the same results for the different values of λ, θ and L. Unlike TS-TACL, TSSD-EWMA chart is very dependent on the parameters values.

Results in the Table (7.2) show that TS-TACL is usually better in terms of accuracy, sometimes worse in terms of FP ($\lambda = 0.5$, $\theta = 0.05$, $L = 2.5$) and sometimes better ($\lambda = 0.4$, $\theta = 0.1$, $L = 2$) in terms of FP than TSSD-EWMA. In terms of FNs and Recall, TS-TACL chart is better than TSSD-EWMA in all cases except when ($\lambda = 0.5$, $\theta = 0.1$, $L = 2$). The reason that TSSD-EWMA when $\lambda = 0.5$, $L = 2.5$ and $\theta = 0.05$ is better than TS-TACL is due to the very small weight given to the current errors ($\theta = 0.05$) in TSSD-EWMA together with the small value of $L = 2.5$. In fact only when decreasing these values, θ and L, simultaneously in TSSD-EWMA, the ability of not signaling an alarm when it is not there increases. That's why the FP rate in TSSD-EWMA is better than TS-TACL only in this case. However, in all the other cases, our CC outperforms TSSD-EWMA in terms of FP.

7.4 D3: Analysis on 72 variants of SEA dataset

7.4.1 Methodology

Because of the lack of available public repository problems of data stream with concept drift, some techniques were proposed in the literature in order to test the SEA problem in different types of difficulties. **[Asensio et al., 2014]** propose SEA with varying noise levels, with non linearity, among other variants of the real world data difficulties that were studied in SEA dataset. We first apply our proposals to 24 permutations of sequences of SEA dataset concepts. Second, we propose learning SEA dataset concepts from the underlying distribution of misclassification error

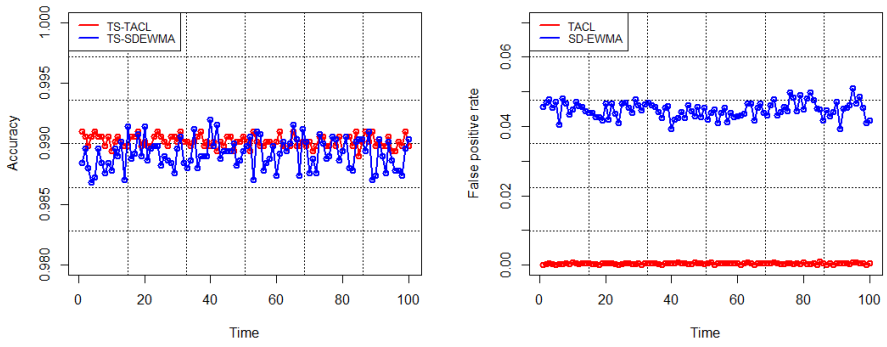

Figure 7.7: Illustration of TACL and SD-EWMA in terms of FP and TS-TACL and TSSD-EWMA in terms of Accuracy based on D2 dataset.

Table 7.2: TACL and TS-TACL chart comparison with SD-EWMA and TSSD-EWMA based on D2: Jumping mean dataset.

Algorithm	Param	FP	FN	Recall	Acc
TACL	($\lambda = 0.5, L = 3$)	0.0028	0.47	0.526	0.99
SD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 2$)	0.0470	0.082	0.917	0.95
SD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 2.5$)	0.0143	0.240	0.757	0.981
SD-EWMA	($\lambda = 0.5, \theta = 0.01, L = 3$)	0.0042	0.402	0.597	0.978
SD-EWMA	($\lambda = 0.5, \theta = 0.05, L = 2.5$)	0.0110	0.260	0.738	0.983
SD-EWMA	($\lambda = 0.5, \theta = 0.01, L = 2$)	0.0130	0.28	0.719	0.981
TS-TACL	($\lambda = 0.2, L = 2.5$)	0.00034	0.165	0.845	0.999
TS-TACL	($\lambda = 0.5, L = 3$)	0.00034	0.165	0.845	0.999
TS-TACL	($\lambda = 0.2$, L = 3)	0.00034	0.165	0.845	0.999
TSSD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 2$)	0.0090	0.1002	0.899	0.989
TSSD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 2.5$)	0.0022	0.2520	0.747	0.992
TSSD-EWMA	($\lambda = 0.5, \theta = 0.01, L = 3$)	0.0012	0.4040	0.595	0.99
TSSD-EWMA	($\lambda = 0.5, \theta = 0.05, L = 2.5$)	0.0002	0.2709	0.729	0.992
TSSD-EWMA	($\lambda = 0.5, \theta = 0.1, L = 2.5$)	0.0029	0.2899	0.710	0.991

rates of DWM-WIN algorithm and we use 3 different classifiers: kknn, naiveBayes and rpart. We apply TACL, TS-TACL, SD-EWMA and TSSD-EWMA to the different variants of SEA dataset and we compare them based on Accuracy, FP, FN and Recall measures. The detailed results of the 24 permutations of SEA dataset based on kknn, rpart and naiveBayes for TACLs and SD-EWMAs charts are illustrated in Tables (11.1), (11.2), (11), (11.4), (11.5) and (11.6) in the Appendix. Some of the 24 permutations lead to a sudden concept drift such as sequences $(9,7,8,8.5)$ and gradual concept drift such as $(7,8,8.5,9)$. An illustration of the different TACL charts in sudden and gradual concept drifts is presented in Figure (7.8). As it is shown, the CLs of TACL are adjusted each time a T out of control observations are detected. It copes with concept drift and adapts the CC to the nonstationarity of the process. We note that the misclassification error rates' behavior is decreasing after each concept drift detection. However this decrease is not going down until the level before the shift's occurrence. The reason of that is that while this misclassification error rates are decreasing over time impacting the learning process, another shift is occurred during the process and thus this new concept impacts a new increase in the misclassification error rates.

Indeed, in order to perform the concept drift detection, we decide monitoring the misclassification error rates instead of monitoring the basic dataset. The reason of that is that the misclassification error rates process are very informative about the state of the process at each time t, they reflect the reactions of the ensemble of classifiers trained by DWM-WIN during the whole process.

In the next sections, results of SD-EWMA and TSSD-EWMA based on 60000 observations of SEA datasets with 3 concept drifts in observations 15001, 30001, 45001 as well as their analysis are provided. The dataset is divided in 100 batches each one with a size of 600 observations. At each time step, a new batch of data arrives and is incrementally added to the monitoring process. The recent $n-1$ batches are used for training and the last arriving new batch is used for testing.

7.4.2 Reason for monitoring based on classification error rates

Our aim when proposing monitoring variants of SEA dataset based on their misclassification error rates is first to benefit from the fact that the reaction capacity to the change point occurrence is immediate and hence, facilitate the concept drift detection. Second, DWM-WIN is a learning classification method that learns the drift and makes its detection faster. Thus, monitoring the error rates would improve the variation detection ability caused by any type of change: covariate shift, concept drift or any other one, better than monitoring the SEA dataset. We note that any classification method designed to cope with concept drift can be used. In this thesis we choose using DWM-WIN because of its ability to detect concept drift and to handle online data stream processes under non stationarity assumptions.

We propose learning SEA problems from the underlying distribution of the misclassification error rates of DWM-WIN algorithm. So, we apply the DWM-WIN based on 100 batches, we use DWM-WIN based on an ensemble of kknn, then the ensemble of naiveBayes and finally an

ensemble of rpart classifiers. We test 24 permutations, the basic sequence proposed by **[Street and Kim, 2001]** and the ANOVA test rejects the null hypothesis that DWM performs the same on average for all 24 permutations.

An illustration of handling some problems of SEA dataset (a) sudden and gradual concept with (b) different base learners: kknn, naiveBayes and rpart classifiers of the different CC is presented in Figure (7.8). Through the updating of the CLs procedure used in TACL as shown in this Figure, the TACL chart has the ability of detecting the drifting concepts and coping with them. This procedure is done in the good way that gives the opportunity to correctly detect the next drifts as well as distinguishing between out of control situations due to changing environment and real situations of concept drift.

Obviously, it can be seen that our proposed new CC performs much better than other traditional CCs in detecting the drifting concepts by updating the CLs of the CC in the correct way that gives the opportunity to correctly detect the next drifts a well as distinguishing between out of control situations due to changing environment and real situations of concept drift.

The fact that the CLs of the TACL chart are updated over time shows the ability to cope with the nonstationarity of the environment. Indeed, these CLs computed recursively take into account the new information of the data construction. Error rates are computed at the end of each batch of the dataset. Thus, the model has a good ability to quickly adapt itself to the new situations during one batch. The comparison of these different CCs is provided in the next subsections.

7.4.3 Global comparison

To provide a global comparison analysis, a mean of the 24 variants of the SEA dataset and the 3 different basic algorithms is illustrated in Table (7.3). As it is shown, TACL outperforms SD-EWMA in terms of Accuracy, FP, FN and Recall. Additionally, TS-TACL outperforms TSSD-EWMA in terms of Accuracy, FN and Recall. A detailed analysis of the difference between methods in terms of different measures is elaborated in the next section.

Table 7.3: Mean performance mean over the different algorithms of TACL based on kknn, naive-Bayes and rpart with 100 runs.

Algorithm	Accuracy	FP	FN	F-measure	Recall
TACL	**0.965**	**0.035**	**0.053**	**0.771**	**0.946**
TS-TACL	**0.989**	**0.006**	**0.053**	**0.916**	**0.946**
SD-EWMA	**0.950**	**0.036**	**0.074**	**0.738**	**0.970**
TSSD-EWMA	**0.979**	**0.005**	**0.085**	**0.891**	**0.910**

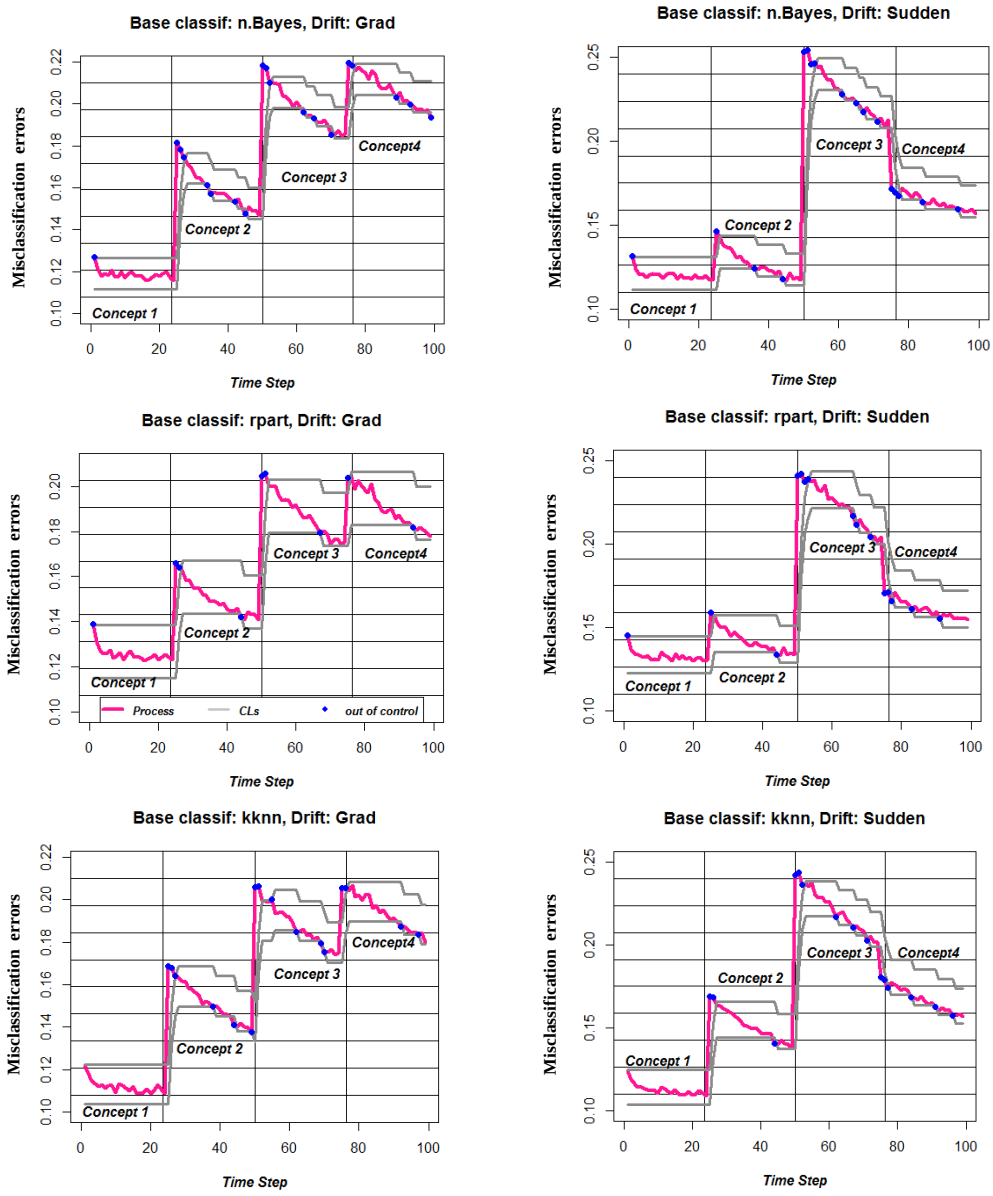

Figure 7.8: Monitoring DWM-WIN error rates on SEA dataset with gradual and sudden concept drifts based on TACL chart.

7.5 Results analysis

7.5.1 Analysis in terms of FP

TACL and SD-EWMA The two methods are very competitive in terms of FP rate. All FPs are very low. This is illustrated in the boxplot of Figure (7.10). The reason of that is due to the fact that the two methods are modeled in a way that adjusts the CLs and adapts itself to the new process

Table 7.4: ANOVA Test for FP in SEA dataset.

	Df	Sum Sq	Mean Sq	F value	p-value
CCs	3	0.059741	0.0189136	151.12	$< \mathbf{2.2} \cdot 10^{-16}$
Residuals	284	0.035544	0.0001252		

variation. The first benefit of the adjustment of the CLs is to reduce the number of false detections. Indeed the adaptation of the CLs allows the distinction between the true outliers and the false ones and accordingly reduces the number of FPs. This is illustrated in the boxplot of Figure (7.10) where TACL performs usually better than SD-EWMA whereas TSSD-EWMA achieves better or similar performance than TS-TACL chart in terms of FPs.

TS-TACL and TSSD-EWMA The use of the two stage procedure, has improved the results of the one stage for both methods. As it is shown in Table (7.3), the FPs are reduced from 0.035 in TACL to 0.006 in TS-TACL which means a decrease of 0.029. Similarly, the FPs of SD-EWMA are reduced from 0.036 to 0.005 which means a decrease of 0.031. Whereas TACL is better than SD-EWMA in terms of FPs, the two stage procedure, in terms of the indicated measure of performance, reduces false detection in SD-EWMA more than that achieved in TACL.

ANOVA test Friedman's test **[Fox, 2011]** rejected the null hypothesis that the different algorithms have the same mean over the different FPs with a computed p-value $< 2.2 \cdot 10^{-16}$. Therefore we apply the post hoc test using Tukey's test to confirm the results of ANOVA test, see Table (7.4).

Tukey's test The Tukey's test was used by **[Yandell, 1997]**. As it is shown in Table (7.5), the test rejects the hypothesis that TACL, TS-TACL, SD-EWMA and TSSD-EWMA perform the same on average. We note that $diff$ is the difference between the two groups mean, lwr indicates the lower end point of the confidence interval, upr indicates the upper end point of the confidence interval and p_{adj} is the Tukey adjusted p-value. In fact the values of p_{adj} indicating a difference between the values of FPs are approximating 0. Although the difference between the two algorithms is not so large, Tukey's test confirms the ANOVA test that the four algorithms perform differently in terms of FPs.

Conclusion Being both very competitive, we conclude that TACL has better ability to reduce the false detections, whereas in the two stage procedure, TS-EWMA achieves better FPs when handling problems of concept drift in SEA dataset.

7.5.2 Analysis in terms of FN

TACL and SDEWMA TACL chart outperforms SD-EWMA in terms of FNs with 0.021 difference as shown in Table (7.3). More specifically, Figure (7.10) illustrates that TACL chart has

usually lower FN than SD-EWMA in the different cases of SEA based on DWM-WIN with kknn, rparts or naiveBayes classifiers. This is due to the fact that TACL chart is designed to reduce the number of misdetection. The reason of that is that TACL updates the CLs only when a shift is detected. Thus, the new CLs contain a history of the detected shifts. However, SD-EWMA chart adjusts the CLs each time with new observation, independently of the fact that whether a shift or a non stationarity is detected or not. Accordingly, our proposed method is more efficient in reducing the FNs by not detecting an alarm when it is not there and it represents a learning shift history chart.

TS-TACL and TSSD-EWMA The use of Kolmogorov Simirnov test of non-stationarity improves the ability of our proposed method to outperform SD-EWMA. In fact, as it shown in Table (7.3), TS-TACL chart achieves an FN rate of 0.053 whereas that of TS-SDWMA is 0.085. The performance of TACL chart is improved after using the test of nonstationarity on Stage II.

Table 7.5: Tukey's test for FP in SEA dataset.

Algorithms	**diff**	**lwr**	**upr**	p_{adj}
TACL-SDEWMA	-0.0047	-0.0095	0.0001	0.0582
TSSDEWMA-SDEWMA	-0.0299	-0.0347	-0.0251	$< 2.2 \cdot 10^{-16}$
TSTACL-SDEWMA	-0.0304	-0.0352	-0.0256	$< 2.2 \cdot 10^{-16}$
TSSDEWMA-TACL	-0.0252	-0.0300	-0.0204	$< 2.2 \cdot 10^{-16}$
TSTACL-TACL	-0.0257	-0.0305	-0.0209	$< 2.2 \cdot 10^{-16}$
TSTACL-TSSDEWMA	-0.0004	-0.0053	0.0043	$< 2.2 \cdot 10^{-16}$

ANOVA test From Table (7.6), we note that the charts are not significantly different at $\alpha = 0.05$. This result is confirmed by Tukey's test.

Tukey's test Tukey's test shown in Table (7.7) confirms the Friedman's test result that the algorithms are not significantly different in terms of FNs. However, TSSD-EWMA performs somewhat different from TACL and TS-TACL.

Conclusion TACL and TS-TACL are robust to detect concept drift in SEA datasets. However, it may appear unexpected that statistical tests do not reflect this result.

7.5.3 Analysis in terms of accuracy

TACL and SDEWMA Table 7.3 shows that TACL chart outperforms SD-EWMA chart in terms of Accuracy, while SD-EWMA has an Accuracy mean of 0.950, and TACL chart achieves 0.965. The results shown in Table (7.3) and illustrated in Figure (7.10) indicate that TACL has higher Accuracy compared to SD-EWMA thanks to its updating CLs function.

Table 7.6: ANOVA Test for FN in SEA dataset.

	Df	Sum Sq	Mean Sq	F value	p-value
CCs	3	0.05844	0.0194805	2.489	0.06063
Residuals	284	2.22275	0.0078266		

TS-TACL and TSSD-EWMA The robustness of TACL chart in detecting concept drift in terms of Accuracy is also maintained when applying the two stage procedure. This is due to the fact that the CLs paradigm of TACL and TS-TACL allows to learn easily the shift and detect it more accurately. Also, the two stage procedure improves the one stage in terms of Accuracy with 0.024 and 0.029 respectively for TACL and SD-EWMA.

Table 7.7: Tukey's test for FN in SEA dataset.

Algorithms	**diff**	**lwr**	**upr**	p_{adj}
TACL vs SD-EWMA	$-1.6083 \cdot 10^{-2}$	-0.0541	0.0220	0.6954
TSSD-EWMA vs SD-EWMA	$1.8472 \cdot 10^{-2}$	-0.0196	0.0565	0.5937
TS-TACL vs SD-EWMA	$-1.6083 \cdot 10^{-2}$	-0.0541	0.0220	0.6954
TSSD-EWMA vs TACL	$3.4555 \cdot 10^{-2}$	-0.0035	0.0726	0.0907
TS-TACL vs TACL	$1.8735 \cdot 10^{-16}$	-0.0381	0.0381	1
TS-TACL vs TSSD-EWMA	$-3.4555 \cdot 10^{-2}$	-0.0726	0.0035	0.0907

ANOVA test According to Friedman's test shown in Table (7.8), the null hypothesis that the CCs perform the same on average in terms of Accuracy is rejected with a p-value $< 2.2 \cdot 10^{-16}$. This is due to the fact that TACL and TS-TACL are better than SD-EWMA and TS-SDWMA in terms of FPs and FNs as shown in Tables (7.4 and 7.6) and that their procedures increase the ability to correctly classify the true in control observations and the true out of control observations with lower mis- and false detection (FP and FN rates). The reason is that our proposed method is constructed based on a specific model for storing the knowledge about the shift history as well as the immediate flexibility to the process change behavior.

Tukey's test From Table (7.9), the Tukey's test confirms that TACL, TS-TACL, SD-EWMA and TSSD-EWMA perform significantly different in terms of Accuracy at $\alpha = 0.05$ and also at $\alpha = 0.01$. This result were conducted based on the $p_{adj} = 3.6 \cdot 10^{-6} < 0.05$ for TS-TACL and TSSD-EWMA and a $p_{adj} = 2.2 \cdot 10^{-16} < 0.05$ for all the other chart comparisons. Also, the lwr and upr have the same sign which confirms that the differences between the algorithms in terms of Accuracy are significant.

Conclusion TACL and TS-TACL behave much better than SDEWMA and TS-SDEWMA respectively in terms of Accuracy. Statistical tests based on ANOVA and confirmed by Tukey's test show this effect. The time varying CLs based on the CLs's history combined with the current observations allows TACL and TS-TACL to achieve a noticeable robustness against concept drift problems better than SDEWMA and TSSD-EWMA.

Table 7.8: ANOVA Test for Accuracy in SEA dataset.

	Df	Sum Sq	Mean Sq	F value	p-value
CCs	3	0.0654	0.021	227.66	$2.2 \cdot 10^{-16}$
Residuals	284	0.0272	0.0000958		

Table 7.9: Tukey test for Accuracy in SEA dataset.

Algorithms	**diff**	**lwr**	**upr**	p_{adj}
TS-TACL vs TSSD-EWMA	0.0083	0.0040	0.0120	$3.6\ 10^{-6}$
TACL vs SD-EWMA	0.0158	0.1161	0.0200	$< 2.2 \cdot 10^{-16}$
TS-TACL vs TACL	0.0152	0.0110	0.0194	$< 2.2 \cdot 10^{-16}$
TS-TACL vs SD-EWMA	0.0394	0.0350	0.0436	$< 2.2 \cdot 10^{-16}$
TSSD-EWMA vs TACL	0.0152	0.0110	0.0194	$< 2.2 \cdot 10^{-16}$
TSSD-EWMA vs SD-EWMA	0.0311	0.0260	0.0350	$< 2.2 \cdot 10^{-16}$

7.5.4 Analysis in terms of recall

TACL and SD-EWMA As shown in Table (7.3), SD-EWMA is best in terms of Recall. It has 0.970 of Recall measure which is the percentage of TPs that the chart can detect. TACL has also a good Recall measure of 0.946 but less than that of SD-EWMA.

TS-TACL and TSSD-EWMA The ability of TACL chart in identifying a true positive observation is clearly improved. Although in first stage charts, SD-EWMA outperforms TACL in terms of Recall, the TS-TACL has better performance in terms of Recall after applying the Kolmogorov Simirov test of nonstationarity than TSSD-EWMA. However, SD-EWMA is affected negatively during the second stage by reducing the Recall from 0.970 to 0.910 as shown in Table (7.3) which is explained by the fact that the use of the second stage is not usually improving the results of **[Raza et al., 2013]**. The reason of the performnace of our methods is that in TACL chart, the CLs adjustment rule updates the CLs based on the centerline of the previous CLs combined with the new current observation. Thus, if an observation is in control, it becomes easier for the TS-TACL chart to classify it as a true positive observation. Another reason for the improvement in terms of

Recall is that the test of stationarity between subsets copes very good with our proposed method because TACL chart's paradigm is to handle such environment thanks to the centerline adaptation which facilitates finding the optimal CLs settings and thus improves the FN and Recall measures.

ANOVA test Although TS-TACL in the mean outperforms TSSD-EWMA in terms of Recall, the one way ANOVA does not reject the null hypothesis with a p-value of 0.416.

Tukey's test The post hoc test based on Tukey's test confirms that all algorithms are significantly similar in terms of Recall. However, less similarity is observed between TS-TACL and TSSD-EWMA with $p_{adj} = 0.520$.

Conclusion Experimental results confirm that not only TACL and TS-TACL are good in terms of Recall but also SD-EWMA and TSSD-EWMA. In particular, TS-TACL and TSSD-EWMA are very competitive. According to Figure (7.10) we notice that our proposed method has more stability in terms of Recall over the different permutations of SEA dataset, and sometimes being better or as good as other charts.

Table 7.10: ANOVA test for Recall in SEA dataset.

	Df	Sum Sq	Mean Sq	F value	p-value
CCs	3	87.6	29.216	0.9501	0.4168
Residuals	284	8732.8	30.749		

Table 7.11: Tukey test for Recall in SEA dataset.

Algorithms	**diff**	**lwr**	**upr**	p_{adj}
TACL vs SD-EWMA	0.0164	-2.3720	2.4048	0.9999
TSSD-EWMA vs SD-EWMA	1.2893	-1.0990	3.6777	0.5034
TS-TACL vs SD-EWMA	0.0302	-2.3581	2.4187	0.9999
TSSD-EWMA vs TACL	1.2729	-1.1154	3.6613	0.5146
TS-TACL vs TACL	0.0138	-2.3745	2.4023	0.9999
TS-TACL vs TSSD-EWMA	-1.2590	-3.6474	1.1293	0.5241

7.5.5 Analysis in terms of F-measure

TACL and SD-EWMA F-measure is the harmonic average of Recall and Precision as shown in equation 7.3. The behavior of TACL and SD-EWMA in terms of F-measure approximates the behavior in terms of Recall. According to Table (7.3) SD-EWMA shows better F-measure than that of TACL chart.

TS-TACL and TSSD-EWMA Similar to Recall measure, the TS-TACL outperforms TSSD-EWMA in terms of F-measure. In fact, applying the Kolmogorov Simirov test of nonstationarity to validate the true out of control detected in Stage 1 and reject the false ones improves the ability of correctly classifying instances and thus the Recall measure and F-measure increase accordingly.

The reason of that is that Precision measure used to compute the F-measure considers the FPs, and when the latter decreases the F-measure increases. Since FPs are smaller in TS-TACL than in TSSD-EWMA the precision increases along with the F measure.

ANOVA test The Friedman's test shows that the four algorithms have similar robustness in detecting concept drifts.

Tukey's test The tukey's test confirms that TACL and SD-EWMA are competitive in terms of F-measure.

Conclusion TACL and SD-EWMA are both very competitive not only in terms of FN and Recall but also in terms of F-measure.

Table 7.12: ANOVA Test for F-measure in SEA dataset.

	Df	Sum Sq	Mean Sq	F value	p-value
CC	3	45.6	15.195	0.7933	0.4984
Residuals	284	5439	19.154		

Table 7.13: TUKEY Test for F-measure in SEA dataset.

Algorithms	**diff**	**lwr**	**upr**	p_{adj}
TACL vs SD-EWMA	-0.9988	-2.8838	0.8861	0.5196
TSSD-EWMA vs SD-EWMA	-0.8770	-2.7621	1.0079	0.6257
TS-TACL vs SD-EWMA	-0.8539	-2.7389	1.0310	0.6458
TSSD-EWMA vs TACL	0.1217	-1.7632	2.0067	0.9983
TS-TACL vs TACL	0.1449	-1.7401	2.0299	0.9972
TS-TACL vs TSSD-EWMA	0.0231	-1.8618	1.9081	0.9999

7.6 Conclusion

Figure (7.9) summarizes the results of the four compared algorithms in terms of Accuracy and FPs. Indeed, Figure (7.10) reflects the different results of Accuracy and F-measure for the different situations of concept drift from permutation 1 to 24 comparing the different CCs. The results shown

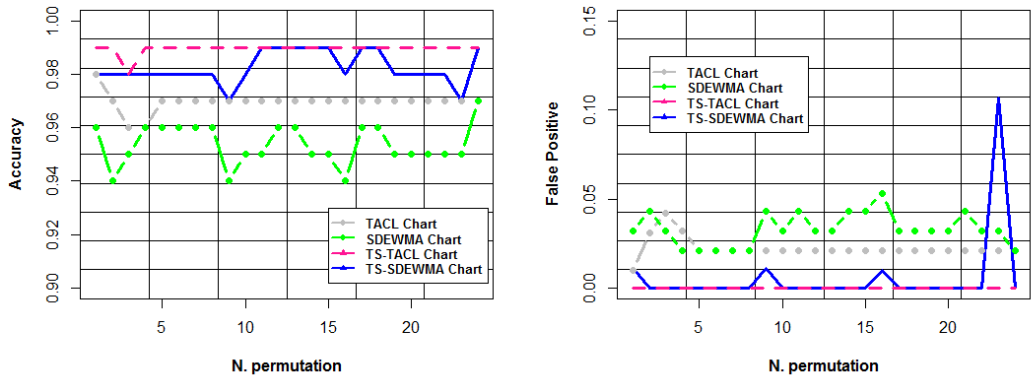

Figure 7.9: Illustration of TACL, TS-TACL, SD-EWMA and TSSDEWMA.

in boxplots summarize all the results and support the robustness of TACL chart which appears as the most adequate CC for learning concept drift under the different situations of environment with small, moderate, large, sudden or gradual concept drifts in the underlying data stream. TACL chart outperforms SD-EWMA in terms of Accuracy and F-measure during the different possible permutations. This is explained by the fact that as soon as a change is detected, TACL learns the drift and updates itself with each new coming observation or batch.

TACL has shown a good performance in terms of Accuracy and F-measure with the error rates of nonstationary classification tasks of DWM algorithm one of the most intelligent heuristics in machine learning. This is due to the adaptive nature of TACL chart and its effectiveness to adjust itself on the changing environments leading to CL settings that best cope with the structure of the concept drift to be detected. In this regard, TACL chart is faster and more sensitive to detect all concept drift situations due to its learning process.

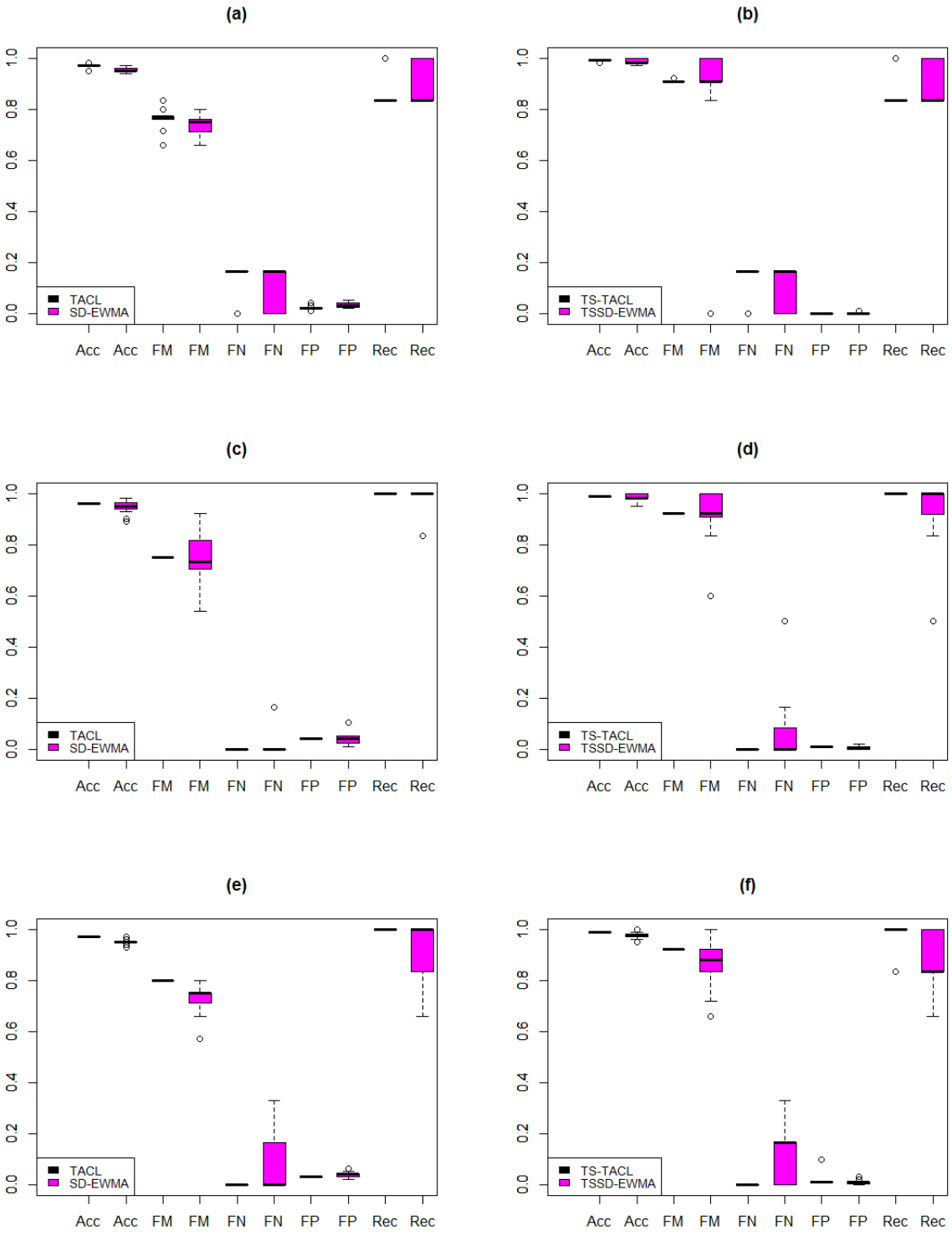

Figure 7.10: Boxplot comparison of TACL with SD-EWMA and TS-TACL with TSSD-EWMA in terms of Accuracy, F-measure, FN, FP and Recall where (a) and (b) are based on the DWM-WIN error rates using an ensemble of kknn, (c) and (d) represent the error rates using an ensemble of naiveBayes, (e) and (f) represent the error rates using an ensemble of rparts.

Part III

Combining Different Control Charts based on Dynamic Ensemble Methods

Monitoring online data stream processes with concept drift based on adaptive CCs has recently interested many researchers due to the need of dealing with time changing data stream processes. However, monitoring a process with individual CCs such as EWMA, CUSUM or XBAR does not have the ability of detecting at the same time small, moderate and large shifts. In fact, individual XBAR, EWMA and CUSUM charts are designed to detect large, moderate and small shifts respectively. However, in most real world applications, it is crucial to simultaneously monitor both small and large shifts in the mean and the variability. Thus, combining CCs has been proposed to look for an overall performance improvement of different shift levels detection. The combined CC transfers some knowledge from one chart to another and benefit from some additional information about the shift. The initial idea of statistically combining CCs was proposed by **[Lucas, 1982]** for Shewhart and CUSUM chart. He applies an intra and inter comparison between historical and new data using a new measurement based on both Shewhart and CUSUM chart statistics formulas. **[Gibbons, 1999]** considers that there is no available computer program except the DUMPStat computer program generating power curves and evaluating the performance of both large and small monitoring programs in real word applications. For that, the author examines the FP and FN rates of the combined Shewhart-CUSUM chart while using a function based on the number of background measurements and other parameters.

[Flaig, 2014a] proposed a new combined CC based on the joint likehood ratio statistics as a combining function. **[Flaig, 2014b]** proposed a combined Shewhart and EWMA chart by presenting EWMA and Shewhart in the same plot. Then, if 4 out of control observations are detected, the program assumes that the process has shifted and a new baseline is reset. **[Sampaio et al., 2014]** proposed a combined np_x - $\overline{X}$ CC to detect process mean shift. The method is based on two stage sampling. The first subsample is evaluated with np_x chart. Then, if an out of control condition is satisfied in the first step, $\overline{X}$ chart is applied on the second subsample of the data. **[Jr and Chob, 2011]** proposed a CC that monitors both the mean vector and the covariance matrix using a variable sampling interval (VSI). Authors propose a VSI CC with another Variable Sampling Rate (VSR) CC based on sequential sampling. This method aims to detect shifts in the mean and the variability of the process based on a combined multivariate EWMA (MEWMA)-type chart which is proved to be more performant than traditional CCs. **[Jr and Chob, 2011]** find it difficult to choose whether to use a Shewhart-type control to monitor large shift in the mean or a MEWMA chart to monitor small shift in the process mean vector. For that, authors considered combining multivariate Shewhart and MEWMA chart to control both the mean vector and covariance matrix. In fact, when monitoring a process that has multivariate normal variables, the Shewhart-type CC traditionally used for monitoring the process mean vector is effective for detecting large shifts. However, for detecting small shifts, it is more effective to use the MEWMA CC. Based on a simulation of different combinations of MEWMA and Shewhart chart, it has been proved by **[Reynolds, 2008]** that combining two MEWMA CCs is better than combining an MEWMA and Shewhart chart.

Additionally, authors prove that using three combined CCs is better than two-combined CCs under some conditions. Indeed **[Reynolds, 2008]** prove that the proposed three CCs combination is sometimes worse than a two-combined CCs for detecting shift in the variability. To determine the coefficients of the combined model, **[Zeng et al., 2007]** used linear regression where predictors are the forecasters and the actual value is the dependent variable. **[Brillman et al., 2005]** proposed a combination of Shewhart chart with a square regression for disease bio-surveillance . **[Reis and Mandl, 2003]** presented a combination of Moving average (MA) and autoregressive (AR) model.

Despite their advantages, most combined CC methods have some weaknesses such as the problem of CC type dependencies. In fact, constructed models are not flexible to be applied with any CC because each combined chart is constructed under specific characteristics of single CCs. Moreover, previous methods use only some specific knowledge from the charting statistics of the individual charts to be combined. Indeed, there is no learning model capable of learning the shift occurring in the process, reducing the false detection and highlight the ability of the identification of recurrent concepts. Indeed, previous combined charts are not adaptive. Thus, fixed CLs charts cannot cope with the non stationarity of the process and cannot adapt the shift detection procedure to such cases in a dynamic self adjustment way.

To treat the above-mentioned issues, Part II of the thesis proposes two Dynamic Ensemble Control (DEC) chart models. They combine different CCs based on dynamic ensemble methods for concept drift, which uses all the knowledge stored in different charting statistics of each individual chart, combine their decisions and monitor both large and small process shift simultaneously. The proposed combination benefits from the online characteristic of DWM-WIN algorithm of **[Mejri et al., 2013]** in detecting the state of the process in nonstationary environment. It consists of three steps: first transforming the task of determining the state of the process into a classification problem by treating CCs as attributes of the data where the drift has to be predicted. Second, DWM-WIN is applied as an ensemble method to combine the different CCs. Third, misclassification error rates of DWM-WIN are monitored based on the time adjusting CC for concept drift detection.

The proposed model would first benefit from all the information stored in each charting statistic. Second, it would be flexible to combine all CCs type thanks to the use of DWM algorithm that is applied as a method for aggregating the different CC's decisions. Also due to the idea of monitoring the misclassification error rates of DWM-WIN, the proposed DEC chart model would be able to learn the shift which occurred in the process, to facilitate the shift detection and to reduce the fault detection rate. Also, because error rates are very informative about the process behavior, some specific monitoring methods could be more suitable than others. This was resolved by using time adjusting CCs for concept drift detection to ensure the correctness of the shift identification. In this regard, we propose an appropriate ensemble chart model to detect all shift sizes under the non stationarity assumption.

In the first chapter of this part, we assume that the class labels are known to assess our method and we present it as an offline learning chart model. Then, in order to make this combined method suitable for online real data applications, we propose an online CC which first predicts the unknown class label with TACL chart. Second, we enhance the application of DWM algorithm to CCs instead of classifiers. Thus, we propose a dynamic modeling based CCs which treats CCs as classifiers. By adding and removing CCs based on their performance to detect a correct state of process based on a weighted majority vote procedure. The online learning adaptive combined chart model updates the weight of the chart as well as the ensemble of the charts at each batch of data streams. Of course, all these methods are used with the aim to improve detection of change points and the identification of concept drifts.

This part presents two new CC combinations: online and offline methods based on three different CCs using a dynamic ensemble method that copes with concept drifting data streams: the DWM-WIN algorithm. The proposed combination benefits from the online characteristic of DWM-WIN algorithm in detecting the state of the process when a stream of data arrives over time. It consists of two steps: first constructing the data based on the combination of the charting statistics. Second, DWM-WIN is applied as an ensemble method to combine the decisions of the different CCs. A normal distribution with different shift values is used to simulate the combined CC. The second proposed CC presents an online method for drift detection which treats CCs exactly as classifiers and improves the overall performance of the individual CCs over the entire process shift range.

This part is outlined as follows. First, chapter 8 proposes the different steps of DEC chart model. Chapter 9 is devoted to the application and evaluation of DEC chart model with different simulated datasets with different shift sizes.

Chapter 8

Combination of Several Control Charts based on Dynamic Ensemble Methods

After optimizing DWM algorithm and assessing the proposed method on different dataset with concept drift in Part I and after proposing TACL chart to monitor non stationary processes and changing environment, we now investigate on proposing a new model based on both DWM and TACL chart to improve the updating procedure when identifying the change points during the process. The chapter is outlined as follows: Section 1 introduces dynamic system modeling. Section 2 presents an overview of the different individual control charts. In Section 3, the problem of individual chart's application is discussed. Section 4 introduces the dynamic ensemble chart model. Section 5 concludes this chapter.

8.1 Dynamic systems modeling

In SPC domain, control charts are usually designed to detect constant shift levels. However, in many industrial applications, the changes are time varying. Thus, it would be of interest to explore the dynamic nature of the shift and to think about new methods which allow attaining a suitable identification of time varying process changes.

[Kaibo and Fugee, 2008] define dynamic processes as the manner in which process variables perform, react and influence each others. **[Nembhard and Kao, 2003]** demonstrate that in a dynamic process it is necessary to define a time interval called "process transition period" rather than immediately respond to the abrupt change in the process. During this process transition from one magnitude to another, a dynamic modeling system should be applied.

Dynamic systems find their applications in mechanical engineering **[Zhang et al., 1991]** or industrial **[Nembhard and Kao, 2003]** processes. An adaptive forecast based monitoring approach was proposed by **[Nembhard and Kao, 2003]** to model a dynamic process for plastic extrusion. Their approach is based on fitting an ARIMA time series model to the process data and then mon-

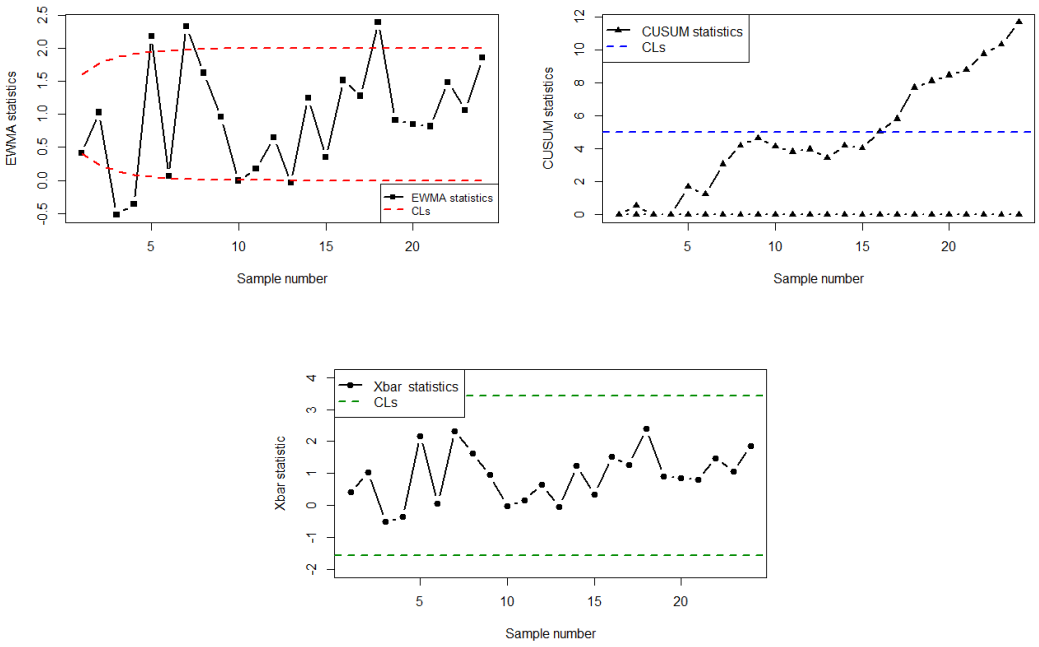

Figure 8.1: Illustative plots of individual EWMA, CUSUM and Xbar charts monitoring a $N(1,1)$ with a shift in the mean of 0.5.

itor the one step ahead forecast errors with traditional charts (such as EWMA, CUSUM or others). Several dynamic modeling systems were discussed in **[Georgiadis et al., 2010]**.

8.2 Detecting a change using control charts

8.2.1 EWMA chart

In SPC, EWMA chart developed by **[Robert, 1959]** is a type of control chart that is used to monitor either variables or attributes-type data using the monitored business or the industrial process entire history of output. Two parameters have to be defined in EWMA chart: $\lambda \in [0,1]$ describing the weight given to the most recent subgroup mean and L denoting the rational subgroup standard deviation in the CLs and it is generally set at 3. An illustration of EWMA chart in a simulated dataset is given in Figure (8.1). Indeed, Table (8.2) details formulas about the charting statistics and the CLs as well as the output decision.

8.2.2 XBAR chart

The Shewhart chart was invented by **[Shewhart, 1931]**. It monitors the process over time based on the mean of the series of instances called subgroups. It is a time based chart which stores

the process history over time. Shewhart-XBAR charts are efficient in detecting large shifts but not small ones. Figure (8.1) provides an illustration and detailed formulas of XBAR chart. More discussion of these charts will be provided in the next Section.

8.2.3 CUSUM chart

The CUSUM chart was first introduced by **[Page, 1954]**. Then, **[Ewan, 1963]**, **[Johnson, 1961]** and **[Johnson and Leone, 1962]** developed its mathematical principles. CUSUM is an efficient alternative to Shewhart procedures. It was constructed to overcome shortcomings of Shewhart control charts. It is well suited for small and moderate mean shift detection thanks to its cumulative sum function. Equations and illustrative plots are given in Table (8.1) and Figure (8.1) respectively. We mention that S_t, E_t and C_t mentioned in Table (8.1) are the decision rules used to identify the out of control situations in XBAR, EWMA and CUSUM charts.

Table 8.1: Properties of three control charts.

	XBAR	EWMA
Statistic	$Shewhart_t = X_i = \sum_{i=1}^{i=n} X_i/n$ **[Montgomery, 2005]**	$EWMA_t = \lambda \cdot Y_t + (1\text{-}\lambda) \cdot EWMA_{t-1}$ **[Montgomery, 2005]**
UCL	$\overline{\overline{X}} + L \cdot \widehat{\sigma} \,/\, \sqrt{n}$	$EWMA_0 + L \cdot (S/\sqrt{n})\sqrt{\lambda/(2-\lambda)[1-(1-\lambda)^{2i}]}$
LCL	$\overline{\overline{X}}$ - $L \cdot \widehat{\sigma} \,/\, \sqrt{n}$	$EWMA_0$ - $L \cdot (S/\sqrt{n})\sqrt{\lambda/(2-\lambda)[1-(1-\lambda)^{2i}]}$
Dec.	S_t= if [$Shewhart_t >$ UCL] or [$Shewhart_t <$ LCL]	E_t=if [$EWMA_t >$ UCL] or [$EWMA_t <$ LCL]
	CUSUM	
Statistic	$C_t^+ = \max(0, C_{t-1} + (z_t\text{-}k))$ for $t = 1, \ldots, n$ $C_t^- = \max(0, C_{t-1} - (z_t + k))$ for $t = 1, \ldots, n$ where k is the reference value and n is the sample size **[Lucas and Crosier, 1982]**	
UCL	h, where h is the decision boundary.	
LCL	-h	
Dec.	C_t = if[$C_t^+ >$ h] or [$C_t^- <$ -h]	

8.3 Problem of individual charts application and motivation

Generally CCs have two important aims. First, they are used as a tool to maintain process stability and control. Thus practitioners have to identify the best CC for any monitoring situation. Second, CCs represent a data analysis tool and learning from the process history behavior over time. Despite their performance, individual CCs suffer from some weaknesses. In this dissertation, we are

interested in improving the performances of CUSUM, EWMA and XBAR charts. In the comparison between the applications of the three CCs separately, the performance and the gain obtained by each CC differs from one method to another depending on the shift size. To evaluate the CC's performance, first simulated results are based on a normal distribution with mean $\mu = 1$ and a standard deviation $\sigma = 1$ and we have simulated different shift ranges. Figure (8.2) provides the best selected measure of performance of each CC. In fact, each CC's performance is highlighted in terms of one or more measures but not all. CUSUM's performance is highlighted in terms of Accuracy and FPs. In EWMA, Accuracy and FN are the best performance features whereas Recall and FNs are the best performance features for XBAR Chart.

First of all it is obvious that each CC has some competencies and some weaknesses. CUSUM chart has higher accuracy and smaller FP for smaller values of shift levels showing a good performance for small shift detection. Nonetheless, it has low performance in detecting moderate and large shifts. This is explained by the high FN and the small accuracy for moderate and large shifts. The out of control scenario is modelled as follows: $\mu_1 = \mu_0 + \delta\ \sigma_0$, where μ_1 is the new mean after the shift is occurred, μ_0 is the initial mean of the distribution, σ_0 is the initial variance of the distribution and δ is the shift size. As compared to XBAR and EWMA charts, CUSUM has better ability of small shift detection. For example, as shown in Figure (8.2) the ability of CUSUM chart in detecting a shift $\delta = 0.5$ increases in terms of accuracy compared to a shift $\delta = 3$ from 0.12 to 0.96, respectively. Also, the ability to reduce the FPs in CUSUM chart while detecting a small shift is more pronounced. For a shift of 1, CUSUM has a FP rate of 0.12 versus a FP of 0.93 for a shift of 4. This is explained by the fact that the charting statistic of CUSUM chart is based on the cumulative sum which makes the detection of small shifts faster. Additional to the weakness of large shift detection, CUSUM has another drawback related to the difficulty to analyze point patterns when all points are highly correlated. Recall and FN measures do not show the effect of CUSUM chart in performing detection of small shifts, that's why we only present the most relevant features of performance.

On the other hand, XBAR chart is unable to detect small shifts. This effect is shown in terms of recall and FN. XBAR registers small recall measure when detecting a small shift. However this measure increases for moderate and large shifts. FP also reflects this positive reaction to moderate and large shifts. As an example, when detecting a shift of 1, the XBAR registers a recall measure of 0.2 and an FN of 0.8 compared to 0.99 and 1 when detecting a shift of 3. Besides the weakness in detecting small shifts, XBAR has another disadvantage that each new point in the monitored process depends only on one subgroup of the data without considering the process history. Moreover, EWMA chart is better in detecting small and moderate shifts than large ones. This is shown in Figure 8.2 in terms of accuracy and FP. A small decrease in the accuracy and an increase in the FP are shown for large shifts. For example, an accuracy of 0.815 when a shift is 1 decreases by more than 0.2 when detecting a shift of 4. Also, the FN increases from 0.15 to 0.36

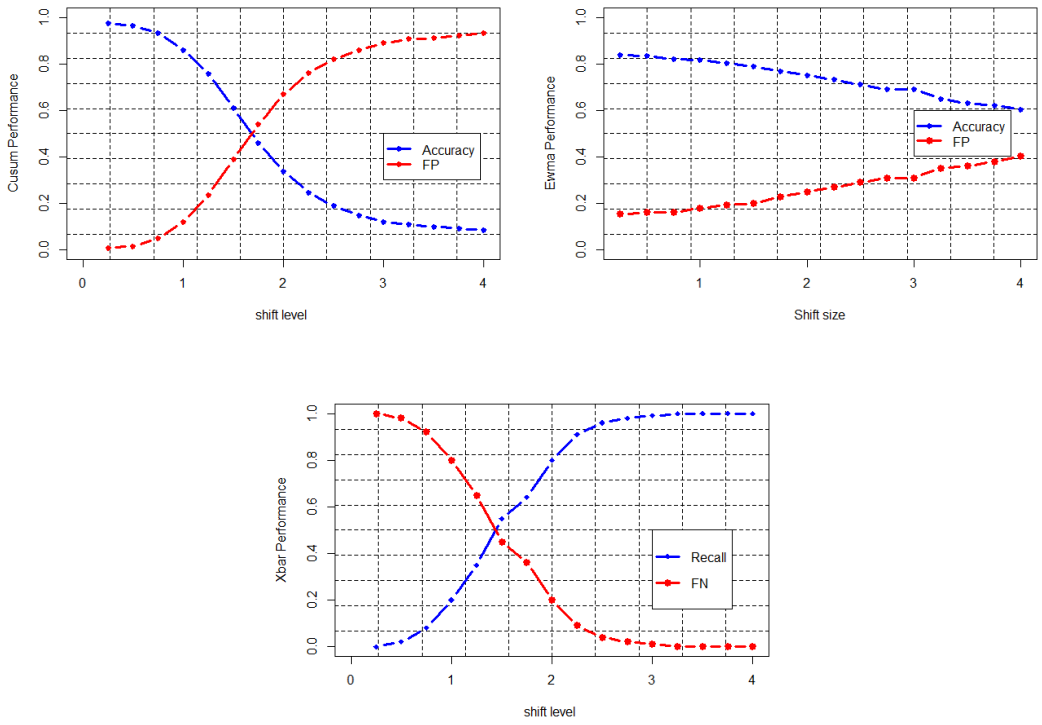

Figure 8.2: Individual charts 's shortcomings in terms of different performance measures where (a) CUSUM chart, (b) EWMA chart and (c) Xbar chart based on $N(1, 1)$ using 400 observations.

for a shift of 0.25 and 3.5, respectively.

To conclude, being relatively good in detecting different shift values, CUSUM is still better than EWMA and XBAR in detecting small shifts whereas XBAR is better than EWMA and CUSUM in detecting large shifts. The reason of that is that CUSUM chart depends on the entire history of the process making the small shift detection easier. However Xbar depends only on the last subgroup of the data process improving its ability to detect large shifts. However, EWMA is a weighted sum of the recent history. Given these conclusions, we are motivated to find a combination of these different CCs. Because each CC has specific advantage and knowledge. Our aim is to benefit from all these advantages and competencies in detecting several shift ranges. Also, another aim of our proposal would be to ease the analysis of monitoring process through the use of dynamic learning control model.

8.4 Classification of the time varying shift for control

Time varying shift detection presents a serious issue in SPC because it requires a dynamic learning monitoring model as well as a re-designing identification system. Although many machine learn-

ing methods were integrated into SPC charts, in order to facilitate the shift identification and to assign a new competence of shift cause identification, the detection of time varying shift in nonstationary environment is still a common issue in SPC. In fact, many techniques use machine learning to improve the identification of special causes of process shift are proposed in the literature.

[Yang and Liao, 2015] and **[Wu and Yu, 2010]** have proposed the use of an hybrid ensemble learning model to monitor both the mean and variance simultaneously. A combination of the prediction of several artificial neural networks was proposed by **[Men et al., 2014]** to improve the wind speed and power forecast in the context of a weather prediction application.

Machine learning algorithms were proposed by **[Cheng and Cheng, 2008]** when using a neural network and a support vector machine to detect a shift in the mean and to determine variable causes of the control. One of the shortcomings of these methods is that learning models were separately applied. Also, the used learning method is not adapted to the process with time varying shifts being based on a fixed learning approach. Many other works have handled SPC shift monitoring problems but without considering the time varying shift and the dynamic nature of the required learning approach to be applied.

Accordingly, all previous methods did not use a dynamic learning model for re-learning and re-designing the time varying shift level. Also, ensemble methods which are designed to cope with shift and concept drift detection were never applied in SPC methods. That's why, our aim is to propose a method that has at least two advantages: (1) to use dynamic classification methods that can learn the shift and adapt the model to the time varying magnitude of the process change, (2) an ensemble method for non stationary environment to aggregate decision from several CCs used simultaneously (3) a method that is able to both detect small and large shifts.

In fact, this research is based on a general set up that works with little number of required information. The most important and only information needed is online data with zeros and ones as class labels, where 0 indicates that the batch or the instance has been correctly classified whereas 1 indicates that the batch or the instance has been wrongly classified. The properties we have to comply with are described in the following points: (a) the data can be in forms of instances or batches, (b) the required information is the classification error rates, (c) the change to be detected can be sudden or gradual, (d) specific or ensemble method can be used and (e) attributes charts or variable charts can be used.

8.5 Dynamic ensemble CC model

8.5.1 Data set assessment of the ensemble method

In order to minimize the risk of choosing a non adequate CC, as well as to enhance the detection of concept drift, an ensemble of CCs is used instead of choosing the best one. The idea of using adaptive techniques to detect concept drift is inspired from machine learning techniques such as STAGGER of **[Schlimmer and Granger, 1986]**, DWM of **[Kolter and Maloof, 2007]** and

Algorithm 8.1: Combined chart model based on Dynamic weighted majority algorithm for drift detection.

1: **Inputs** p: Initialize number of CCs to monitor the process.
β: parameter for decreasing classifier's weight when state decision is correct.
η: parameter for increasing classifier's weight when state decision is incorrect.
θ: threshold for removing classifiers from the ensemble
2: **Step1**: **Data Pre-processing**
1.1: Apply p chart for each batch of data
1.2: Store the charting statistics of each chart in vectors $X_1, X_2, \ldots, X_p$.
1.3: Concatenate vectors X_n in a matrix of p columns: this represents the combined data D.
3: **Step2**: **Apply DWM-WIN algorithm to combine chart's decisions.**
2.1 Initialize classifier's weights.
2.2 Apply DWM to the new generated data D
2.3 Compute the error rates of DWM-WIN based on D

[Kolter and Maloof, 2005a], DWM-WIN of **[Mejri et al., 2013]**, SVM of **[Ahmed et al., 1999]** and IFCS of **[Schlimmer and Granger, 2011]**. An adaptive combined chart is developed based on a weighted majority vote over the different classifers' decisions of the CCs for each instance of the batch. We use the DWM of **[Kolter and Maloof, 2007]** and DWM-WIN of **[Mejri et al., 2013]** to get this class prediction about the learned shift. Algorithm (8.1) defines these steps.

In Step 1, the dataset of the DEC chart model is based on the combination of the charting statistics of k individual CCs. The proposed method is suitable to be applied to all CCs types. Each charting statistic is presented in a vector X_k. Then, the different vectors are concatenated and stored in a matrix called X_k for k = 1,..., p given by

$$X_k = \begin{pmatrix} x_{1k} \\ x_{2k} \\ \vdots \\ x_{pk} \end{pmatrix} \tag{8.1}$$

where p is the number of CCs, X_k is the charting statistics of each CC and k is the range of the CC. For three CCs, the vectors are:

$$X_{Shewhart} = \begin{pmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{k1} \end{pmatrix}, X_{Cusum} = \begin{pmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{k2} \end{pmatrix}, X_{ewma} = \begin{pmatrix} x_{13} \\ x_{23} \\ \vdots \\ x_{k3} \end{pmatrix} \tag{8.2}$$

Equations (8.1), (8.2), (8.3), (8.4), (8.5) and (8.6) describe the different sub-steps of the pre-processing data phase.

For a combination of 3 CCs, the new data will be

$$X^3 = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ \vdots & \vdots & \vdots \\ x_{k1} & x_{k2} & x_{k3} \end{pmatrix} \tag{8.3}$$

More generally, the dataset representing the combination of p charts is:

$$X^p = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p} \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mp} \\ x_{m1+1} & x_{m2+1} & x_{m3+1} & \dots & x_{mp+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{k1} & x_{k2} & x_{k3} & \dots & x_{kp} \end{pmatrix} \tag{8.4}$$

The next pre-processing step consists on applying DWM algorithm to the generated data in order to monitor its classification error rates and to detect the shift in the data and this is done through a simultaneous use of more specific knowledge about the data by using the charting statistics information and the DWM as a technique of concept drift detection. We simultaneously base our experiments on data with different levels of shift in the mean aiming to prove the detection ability of the combined CC versus single charts in detecting concept drift during the classification process. To apply DWM, class labels have to be defined. Thus, before the shift occurs in the process we assume that the class label is 0. Then, when a shift occurs, the class label is 1. For p variables, if the shift occurs in observation $\frac{k}{2}$+1 the class label will be as follows:

$$Classlabel = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{8.5}$$

Finally, the new data is:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p} & 0 \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p} & 0 \\ x_{m1} & x_{m2} & x_{m3} & \dots & x_{mp} & 0 \\ x_{m1+1} & x_{m2+1} & x_{m3+1} & \dots & x_{mp+1} & 1 \\ x_{m1+2} & x_{m2+2} & x_{m3+2} & \dots & x_{mp+2} & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{k1} & x_{k2} & x_{k3} & \dots & x_{kp} & 1 \end{pmatrix} \tag{8.6}$$

DWM is called to detect the concept drift. More precisely, each classifier will initially have a weight of one. The weight of classifiers will be decreased if the classifier predicts incorrectly the state of the process based on the ensemble of chart's statistics and increased if the classifier's global prediction is correct. This step will be applied for the three CCs until having three final weights. If the weight of each CC is under the threshold θ, the classifier will be removed from the ensemble, else it will be maintained. Then, a global prediction vector will be computed from the remaining ensemble of classifiers. When this global prediction of classifiers based on the CC decisions is wrong, a new classifier is added. The same reasoning would continue until finding the best combination of classifiers which detects if the process is in control or out of control for the different batches of the process. The size of the batch affects the detection of the shift via the misclassification error rates. A shift could either detected or absorbed by the system. In fact, for large batch sizes, the shift is detected and considered while updating the weights inside one batch. However, for small batch sizes, the shift induces a high error which is expected to be more obvious with the monitoring method.

8.6 Experiments

8.6.1 Simultaneous small and large shift detection

DEC is a shift detection method based on dynamic classification error rates analysis. In this section, we further explain this method and illustrate it with a small experiment to show the robustness of DEC model in detecting a shift and updating itself. This method is at least as good as or better than the best single charts. Two illustrative examples based on two datasets, with small shift in the mean equal to 0.25 and a large shift of 3.5, are simulated to illustrate the proposed model. The dataset consists of 30 observations where the shift in the mean occurs in observation 11. Therefore, the first 10 observations are simulated based on $N(0.5,0.5^2)$ in both datasets and the second 20 observations are simulated based on a $N(0.75,0.5^2)$ and $N(4, 0.5^2)$ in the first and the second example respectively. First data is denoted D_1 and it is shown in Table (8.2) and second data is denoted D_2 and is shown in Table (8.3). The charting statistics of EWMA, CUSUM, XBAR and DEC charts are X_{1i}, X_{2i}, X_{3i} and Z_i respectively. The upper and lower control limits of EWMA, CUSUM, XBAR and DEC charts are (LCL_1, UCL_1), (LCL_2, UCL_2), (LCL_3, UCL_3) and (LCL_z,

UCL_z) respectively. Tables (8.2) and (8.3) summarize the results for the different methods using the same random generation numbers denoted "seed".

For the first dataset, EWMA and CUSUM chart can detect the shift that occurred in observation 11 at time t_{12} and t_{10}. However, XBAR detects the small shift in D_1 in observation 20. One reason of XBAR's delay to detect the small shift is that its charting statistic is based on independent subgroup of the dataset which decreases the ability of a quick small shift identification. For DEC chart, the shift is detected by the increase in the misclassification error rates. Then, the DWM algorithm needs some time to update the internal knowledge after the shift detection by removing and adding of experts as well as weight adjusting and other steps. All these steps require some time before reaching a certain stability which can be shown when the error goes back to 0. This is what explains how the shifts are detected and how the errors is failing back to 0 after a while. Furthermore, thanks to its prediction ability our proposed method has an immediate detection ability explained by the use of classification methods which eases the shift prediction from observation 10. Accordingly, DEC chart is as good as CUSUM chart but much earlier than EWMA and XBAR in detecting small shift range.

For the second dataset with large shift in the mean, a shift in the last 20 observations is also simulated. According to Table (8.3), DEC model is as good as Xbar chart to identify such shift. It predicts, identifies the shift at observation 10 thanks to the learning classification methods aggregating the decisions of the ensemble of charts' decision, EWMA detects it at t_{11}, CUSUM at t_{11} and XBAR at t_{10}. As expected, XBAR has an early detection ability compared to EWMA and CUSUM to detect such shift in D_2.

This advantage is first explained by the use of the misclassification error rates as charting statistics of the monitoring process in DEC chart. Second reason of the immediate detection is the use of the Time varying shift detection method based on TACL chart which makes the shift detection faster, updates the UCLs and LCLs and accordingly decreases the FPs and the FNs rates during the shift learning and identification process.

8.6.2 Improvement of the misclassification error rates

First, we focus on a shift simulation analysis based on several values of shift parameters. In Figure (8.3), the error rates of EWMA chart for shift values parameters are plotted based on a $N(\mu = 0, \sigma = 1)$ of 400 observations with a shift in the middle. It shows that the change in the error rates of EWMA chart is clearly impacted by the change of the shift level, except the change point of the shift detection which is practically the same. The relation between the speed of the adaptation of the error process after the shift detection and the shift level is clear. Changes in batches 13, 15 and 16 are random change points which are detected with different levels of errors impacted by the level of the shift.

Because the reaction of the chart to the shift is highly informative about the robustness of the method to detect the changes and to adapt itself after the shift detection, we conduct a comparative

Table 8.2: Illustrative examples of EWMA, CUSUM, XBAR and ensemble chart model for small shift in the mean $\delta = 0.25$ in $N(0.5, 0.5^2)$. The first 10 observations follows a $N(0.5, 0.5^2)$ and the last 20 observations follows a $N(\text{shift} + 0.5, 0.5^2)$ using the same set.seed (random number generation). For EWMA, $\lambda = 0.2$ and $L = 3$ are used and for CUSUM, $k = 1$ and $h = 3$.

n	**EWMA**			**CUSUM**			**XBAR**			**DEC**		
	X_{1i}	LCL	UCL	X_{2i}	LCL	UCL	X_{3i}	LCL	UCL	Z_i	LCL	UCL
1	0.472	0.160	0.609	0.660	-2	2	-	0.432	0.787	-	-	-
2	0.462	0.090	0.672	0.260	-2	2	-	0.432	0.787	0	0	0
3	0.603	0.063	0.706	1.850	-2	2	-	0.432	0.787	-	-	-
4	0.518	0.040	0.720	0.790	-2	2	-	0.430	0.787 2	0	0	0
5	0.423	0.031	0.730	0	-2	2	0.605	0.432	0.787	-	-	-
6	0.364	0.241	0.740	0	-2	2	-	0.432	0.787	0	0	0
7	0.278	0.0190	0.750	0	-2	2	-	0.432	0.787	-	-	-
8	0.289	0.016	0.750	0	-2	2	-	0.432	0.787	0	0	0
9	0.203	0.014	0.750	0	-2	2	-	0.432	0.787	-	-	-
10	0.476	0.013	0.750	**2.670**	**-2**	**2**	0.480	0.432	0.787	**1**	**0**	**0**
11	0.598	0.012	0.750	4.040	-2	2	-	0.432	0.787	-	-	-
12	**0.780**	**0.011**	**0.750**	6.660	-2	2	-	0.432	0.787	1	-0.841	1.840
13	0.720	0.011	0.750	8.361	-2	2	-	0.432	0.787	-	-	-
14	0.920	0.011	0.750	7.210	-2	2	-	0.432	0.787	1	-0.840	1.84
15	0.94	0.011	0.750	10.220	-2	2	0.692	0.432	0.787	-	-	-
16	0.74	0.011	0.750	11.450	-2	2	-	0.432	0.787	1	-0.840	1.840
17	0.804	0.011	0.750	9.750	-2	2	-	0.432	0.787	-	-	-
18	0.709	0.0110	0.750	11.040	-2	2	-	0.432	0.787	1	-0.840	1.84
19	0.590	0.011	0.750	10.380	-2	2	-	0.432	0.787	-	-	-
20	0.690	0.011	0.750	9.180	-2	2	**0.880**	**0.432**	**0.78**	0	-0.840	1.840
21	0.790	0.011	0.750	10.560	-2	2	-	0.432	0.787	-	-	-
22	0.680	0.011	0.750	12.230	-2	2	-	0.432	0.787	0	-0.840	1.84
23	0.570	0.011	0.750	11.390	-2	2	-	0.432	0.787	-	-	-
24	0.600	0.011	0.750	10.190	-2	2	-	0.432	0.787	0	-0.840	1.84
25	0.680	0.011	0.750	10.630	-2	2	0.420	0.432	0.787	-	-	-
26	0.480	0.011	0.750	11.82	-2	2	-	0.432	0.787	0	-0.840	1.840
27	0.469	0.011	0.750	9.370	-2	2	-	0.432	0.787	-	-	-
28	0.532	0.011	0.750	8.960	-2	2	-	0.432	0.787	0	-0.840	1.840
29	0.503	0.011	0.750	9.530	-2	2	-	0.432	0.787	-	-	-
30	0.508	0.011	0.750	9.030	-2	2	0.649	0.432	0.787	0	-0.840	1.840

Table 8.3: Illustrative exmaples of EWMA, CUSUM, XBAR and ensemble chart model for large shift in the mean of 3.5 in $N(1, 1)$. The first 10 observations follows a $N(0.5, 0.5^2)$ and the last 20 observations follows a $N(0.5 + \text{shift}, 0.5^2)$ using the same set.seed (random number generation). For EWMA, $\lambda = 0.2$ and L =3 are used and for CUSUM, $k = 1$ and $h = 3$.

n	**EWMA**			**CUSUM**			**XBAR**			**DEC**		
	X_{1i}	LCL	UCL	X_{2i}	LCL	UCL	X_{3i}	LCL	UCL	Z_i	LCL	UCL
1	0.472	0.160	0.609	0.084	-2	2	-	2.450	3.101	-	-	-
2	0.460	0.090	0.672	0	-2	2	-	2.450	3.101	0	0	0
3	0.603	0.063	0.706	0	-2	2	-	2.450	3.101	-	-	-
4	0.518	0.043	0.726	0	-2	2	-	2.450	3.101	0	0	0
5	0.420	0.031	0.738	0	-2	2	2.550	2.450	3.101	-	-	-
6	0.360	0.024	0.740	0	-2	2	-	2.450	3.101	0	0	0
7	0.270	0.019	0.746	0	-2	2	-	2.450	3.101	-	-	-
8	0.289	0.0160	0.751	0	-2	2	-	2.450	3.101	0	0	0
9	0.203	0.016	0.754	0	-2	2	-	2.450	3.101	-	-	-
10	0.470	0.014	0.756	1.580	-2	2	**2.350**	**2.450**	**3.101**	**1**	**0**	**0**
11	**1.240**	**0.0132**	**0.757**	**8.700**	**-2**	**2**	-	2.450	3.101	-	-	-
12	1.950	0.012	0.750	16.740	-2	2	-	2.450	3.101	1	-0.090	1.690
13	2.450	0.011	0.758	24.090	-2	2	-	2.450	3.101	-	-	-
14	2.640	0.011	0.758	29.320	-2	2	-	2.450	3.101	1	-0.090	1.690
15	3.106	0.011	0.759	37.650	-2	2	2.642	2.450	3.101	-	-	-
16	3.340	0.011	0.759	44.660	-2	2	-	2.450	3.101	1	-0.090	1.690
17	3.310	0.011	0.759	49.470	-2	2	-	2.450	3.101	-	-	-
18	3.509	0.011	0.759	56.520	-2	2	-	2.450	3.101	1	-0.090	1.690
19	3.52	0.011	0.759	62.122	-2	2	-	2.450	3.101	-	-	-
20	3.49	0.011	0.759	67.305	-2	2	2.830	2.450	3.101	1	-0.090	1.690
21	3.66	0.011	0.759	74.420	-2	2	-	2.450	3.101	-	-	-
22	3.810	0.011	0.759	81.750	-2	2	-	2.450	3.101	1	-0.090	1.690
23	3.75	0.011	0.759	87.210	-2	2	-	2.450	3.101	-	-	-
24	3.680	0.011	0.759	92.390	-2	2	-	2.450	3.101	0	-0.090	1.690
25	3.740	0.011	0.759	98.810	-2	2	3.022	2.450	3.101	-	-	-
26	3.840	0.011	0.759	105.790	-2	2	-	2.450	3.101	0	-0.090	1.690
27	3.650	0.011	0.759	110.040	-2	2	-	2.450	3.101	-	-	-
28	3.660	0.011	0.759	115.820	-2	2	-	2.45	3.101	0	-0.090	1.690
29	3.730	0.011	0.759	122.330	-2	2	-	2.450	3.101	-	-	-
30	3.710	0.011	0.759	128.304	-2	2	-	2.450	3.101	0	-0.090	1.690

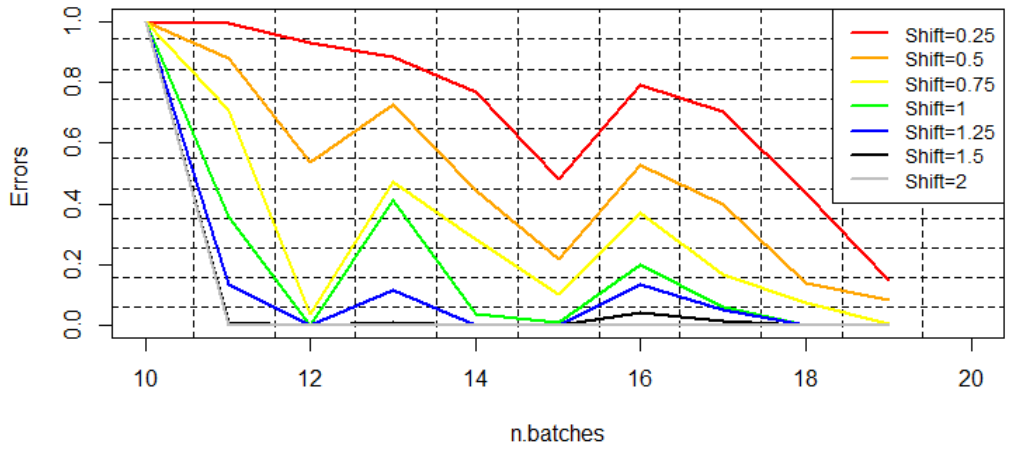

Figure 8.3: Shift comparison based on ewma misclassification error rates.

study between the different variants of DEC chart and the individual charts. Thus, we compare results obtained by the different variants of DEC chart denoted: (1) EWMA-CUSUM, (2) EWMA-Xbar, (3) CUSUM-Xbar and (4) XBAR-CUSUM-EWMA with the application of individual CCs: EWMA, CUSUM and Xbar. The combined CCs were implemented based on 400 observations with a mean shift in the middle using a number of batches of 5, 10, 20, 30 and 40. The procedure used by DWM algorithm to divide the data in different batches, train on the $n-1$ recent batches and test on the last batch is the same procedure used for generating a k fold cross validation which consists on portioning the data into n subsets, performing the training in one subset and test on the other subsets. So, the number of batches is equal to the number k used to generate k fold cross validations. Indeed, we compute the mean over 10 runs for each method. By using this procedure, we abstain sampling biases. Furthermore, we use other measures of performance such as F-measure, recall, FPs ad FNs and statistically compare the difference between the methods.

DEC chart is configured with the following parameters: $\beta = 0.449$, $\eta = 1.13$ and $\theta = 0.01$. These values were selected after applying GA with a population size = 50. For each method 10 runs were performed for each method using each time a different random seed and providing results by the mean of these iterations. The task of DEC chart is to mine the charting statistics of the different CCs. To treat this problem, DEC proposes a classification task solution. It combines many decisions at each time step based on DWM-WIN algorithm to increase the probability of correctly classifying a change point.

Here, we track the degree of diversity of the CC ensemble. Thus, we analyze the capacity of the combined CC model based on a normal distribution with $\mu = 0$ and $\sigma = 1$ by comparing it with the individual chart model in terms of misclassification error rate of DWM-WIN. For the

EWMA chart, many values of the parameter λ are used in the literature. In general, values of this parameter are between 0 and 1 as stated by **[Steiner, 1999]**. In this experiment, $\lambda = 0.2$ and $L = 3$ are used with the notation that these parameters can be optimized. For CUSUM chart we use the decision boundary $h = 4$, the reference value $k = 1$ and the target value representing the actual process mean. $T = 1$. For XBAR chart used in this simulation, we use the Xbar'one chart function from the qcc R package which computes statistics required by the xbar chart for one at-time data. The reason of this choice is that we need to combine vectors of charting statistics of equal size. This can only be obtained by the Xbar'one chart where the number of subgroups is equal to the dataset size as in CUSUM and EWMA charts.

In Figure (8.4), the horizontal axis represents the shift level and the vertical axis the error rates of the classification ensemble method used to combine the different chart models. We base our analysis on 6 shift levels. The analysis is as follows:

Shift = 0.25: The differences between methods are more pronounced for small shift values. Although all other individual chart models as well as the combined ones: EWMA-CUSUM, EWMA-XBAR and XBAR-EWMA charts begin with relatively high errors, EWMA-CUSUM-XBAR starts with low error rates by perfectly coping with the shift level 0.25. In fact, the ensemble chart model begins with misclassification error rates between 0.055 and 0.12 for all different number of batches situations, however it is greater than 0.4 for individual charts. Thus, our proposed CC outperforms other CCs and is able to track small shifts when considering the statistics information of more than one chart.

Shift = 0.5: When the shift level is 0.5, errors of the combined three charts increase but they are still less than 0.22. On the other hand, all combined two chart models: EWMA-CUSUM, EWMA-XBAR and CUSUM-XBAR perform better than individual charts for all batch size situations. For example, for $N = 5$, these methods greatly outperform individual charts. This is due first to the advantages of the ensemble method technique over the individual ones. Second, this performance is also due to the fact that statistics used for each CC are very informative about the quality control of process behavior. Also, the DWM-WIN used in the combined chart models allows this technique to easily detect shifts in the process and to update the internal knowledge announced in the CC about this shift.

Shift = 1: For this shift level, errors of individual charts decrease compared to smaller shift level situations because all these CCs are more sensitive to detect large shifts easily, except the CUSUM which is based on accumulating changes in one direction and is more sensitive to small shifts. Combined models are still more robust to react to the shift better than individual charts. XBAR, CUSUM and EWMA do not show this effect on detecting shifts in processes because the internal information does not allow it. More rigorously, the combined chart model performs better

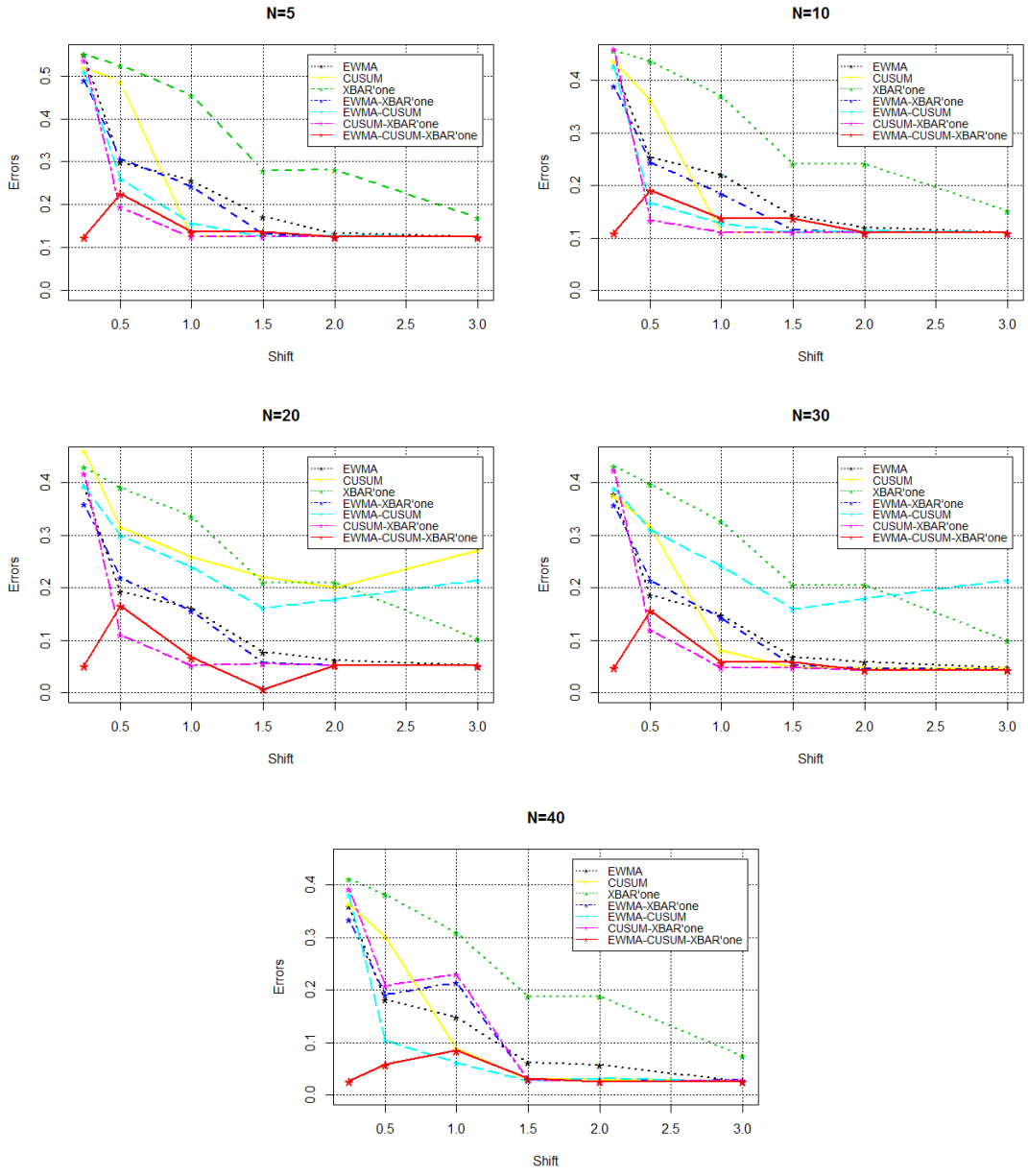

Figure 8.4: Comparison of the misclassification error rates of DWM-WIN monitoring the different combined control chart models versus the individual charts for a variety of shift levels.

than other individual charts, in particular XBAR-CUSUM and EWMA-CUSUM are the best ones for $N = 10$ and $N = 30$. This is due to the fact that CUSUM chart is more sensitive to small shifts and EWMA and XBAR perform better with large shift cases as explained earlier. Thus, combining CUSUM with XBAR and with EWMA outperforms the decision of only one chart model in detecting a shift of 1.

Shift = 1.5: All CC error rates decrease when the shift level increases to 1.5. In particular, EWMA-CUSUM-XBAR chart errors are between 0.05 and 0.12 whereas other charts reach error rates of 0.2 and even more for $N = 5$. An error rate of 0.28 is obtained with XBAR chart. More precisely, XBAR-CUSUM shows a good reaction capacity to shift detection for all batch size situations. This is due to the combined effect of the two charts and the diversity of the internal knowledge given by the two different charts, one is sensitive to very small shifts and the other is more sensitive to large shifts.

Shift = 2 Combined EWMA-CUSUM-XBAR chart outperforms other CCs for moderate and large shifts. In fact, it has a maximum error of 0.12, individual charts reach 0.28, two combined models reach 0.18 for $N = 30$.

Shift = 3: Similar results as for shift level = 2.

Experimental results show that combined EWMA-CUSUM-XBAR chart outperforms individual CCs. When the diversity of combined CCs increases, the misclassification error rates decrease accordingly. When combining three CCs based on a dynamic ensemble method for concept drift, the accuracy of the CC improves. As shown in Figure 8.4, EWMA-CUSUM-XBAR chart depicts smaller values of misclassification error rates better than the combined two chart model. Interestingly, the diversity in combining CCs presents an important factor to improve the classification accuracy rate of the proposed chart in both monitoring small and large shifts in the mean.

8.7 Comparison of DEC chart with combined SFEWMA-X chart

In this section we focus on comparing the proposed dynamic chart DEC with one of the combined charts from the literature called Single Featured EWMA chart of **[Liu and Tien, 2010]**. Like DEC chart, SFEWMA-X was proposed in the aim to both monitor small and large shifts in the process. Indeed, we choose this method among others since it is based on an only one statistic and CLs. It transforms the EWMA statistics to the same magnitude of XBAR chart by the use of a rescale parameter in order to use the same CLs for EWMA and XBAR which is similar to our proposed method in unifying the charting statistic of 3 CCs and more. In their article, **[Liu and Tien, 2010]** compared SFEWMA-X only to individual charts. Here, we also compare SFEWMA-X to individual charts and also the proposed DEC chart. Comparisons were conducted in terms of

errors and in terms of different performance measures. Figure (8.5) performs comparison in terms of error rates of DEC chart and SFEWMA-X chart for the different shift sizes.

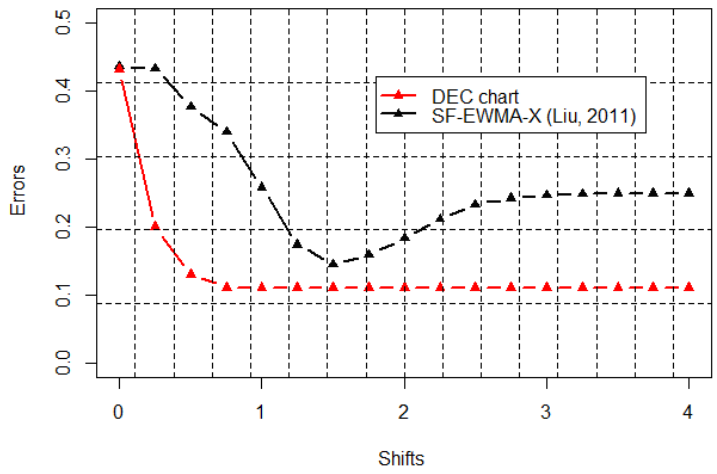

Figure 8.5: Performance evalutioon of proposed DEC chart model versus combined SFEWMA-X chart based on $N(\mu = 1, \sigma^2 = 1)$ using 400 observations.

The two methods begin with equal performance when the dataset doesn't contain any shift. Then, as the shift increases, DEC's error decreases until achieving a stable level of error equal to 0.11 for shift ranging from 0.75 to 4. Concerning SFEWA-XBAR chart, the error is decreasing until a shift of 1.5 achieving 0.13 of errors. Then, it increases again to achieve 0.25 for shift 2.5 until 4. Thus, the performance of dynamically combining three CCs based on DWM algorithm outperforms the SFEWMA-X chart. The reason for that is that first, DEC chart uses more statistical information than SFEWMA-X chart. Second, the dynamic combination through the use of DWM algorithm is much more informative about the shift detection points than the use of the rescaled parameter proposed to combine EWMA and XBAR in SFEWMA-X. Indeed we note that the decrease in the error is impacted by the shift level. This is explained by the relation between the knowledge about the shift level and the classification errors. In fact, a clear knowledge about the shift, obtained by higher levels, impacts the classification errors. Thus, the clear is the knowledge representation of the shift can be observed, the smaller is the classification errors. This reasoning is confirmed by **[Pasman, 2015]** when they make the high relation between the knowledge-based levels and the error of classification.

To go further in our analysis, we want to understand the cause of high errors in SFEWMA-X compared to DEC chart. Thus, we concentrated on the behavior of the two charts for shifts smaller

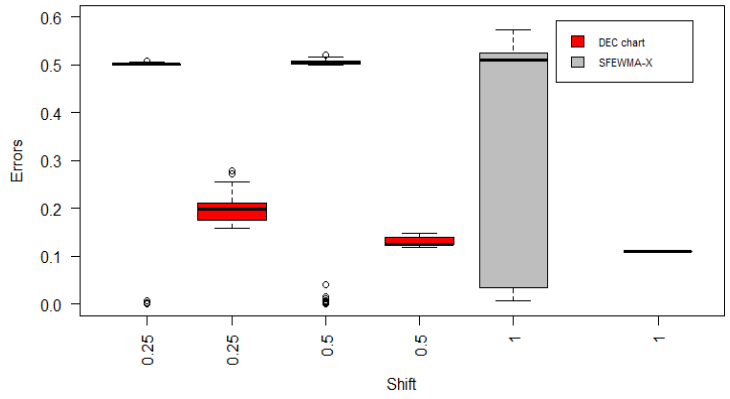

Figure 8.6: Boxplot comparing the mean over Type I and Type II errors of DEC chart and SFEWMA-X charts based on 100 runs.

than 1. Figure (8.6) shows the high differences in errors between the two methods based on the mean over the Type I and Type II errors.

As noticed, the variance in SFEWMA-X is higher than in DEC chart for shift of 1. Compared to SFEWMA-X chart, the DEC chart is more stable in results and procures smaller error variance.

To go further in our analysis, we compare DEC chart with SFEWMA-X as well as individual CUSUM, XBAR and EMWA charts in terms of accuracy, recall and FP measures. Figures (8.7) to (8.9) show the comparisons of performances of the different methods for small-moderate, large and all shifts.

We mention that the performance of each method could be highlighted based on one measure and not the others. Each measure is sensitive to evaluate one or more methods but not all of them. That's why, instead of error rates, we use accuracy, FPs and FNs. For small and moderate shifts as shown in Figure (8.7), DEC chart outperforms SFEWMA-X and individual charts in terms of accuracy, recall and FP rates. SFEWMA-X is competitive with respect to DEC chart and outperforms individual EWMA and XBAR in terms of recall. The performance of CUSUM chart is not highlighted in this Figure because of the not good performance of this type of chart in detecting the moderate shift which is included in the computations done in these boxplots.

For large shifts, the performance of DEC chart is also maintained. It optimizes the different measures compared to other methods. The performance of SFEWMA-X chart is highlighted in terms of recall measure when a competitive recall measure is achieved. Concerning the comparison between the individual charts in detecting a large shift range, results show that as expected, XBAR chart outperforms CUSUM and EWMA charts in terms of Accuracy, Recall and FPs. The

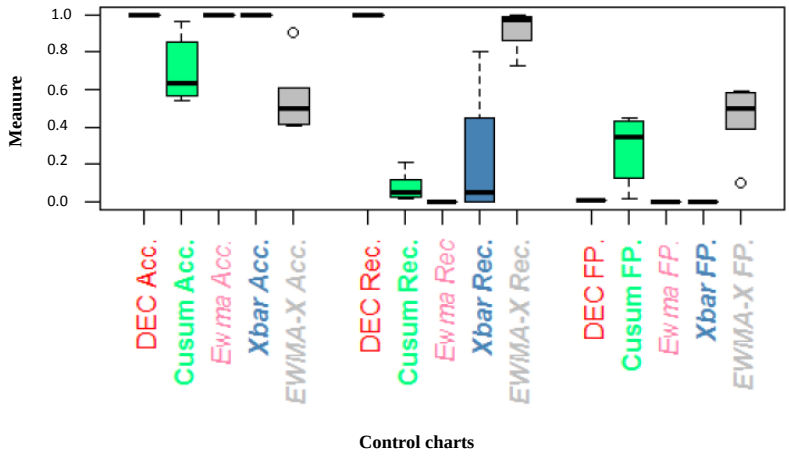

Figure 8.7: Performance evaluation of DEC model compared to individual EWMA, CUSUM and Xbar in terms of Accuracy, Recall, FP and FN rates: small-moderate shifts.

performance of CUSUM is not shown because of the moderate shift included in the computations done in Figure (8.8). DEC chart's performance is noticed through the different measures and it shows a high robustness and stability to the shift detection than the combined SFEWMA-X chart as well as show other individual charts.

For more general results over the different shift sizes, Figure (8.9) illustrates a summary. SFEWMA-X outperforms individual EWMA and XBAR and is as good as DEC chart in terms of Recall. However, this effect is not shown in terms of accuracy and FP rates. Being already very performant to shift detection, this method assumes that the class labels are unknown. In the next chapter, we propose a new combined chart without the assumption of known classes in order to make the application in real word data possible.

8.8 Conclusion

The proposed CC does not only exhibit superior robustness to individual EWMA, CUSUM and XBAR for some performance measures but also presents a new heuristic for shift learning and monitoring in nonstationary environment. Another advantage of the proposed CC is that it is suitable for any other CC types.

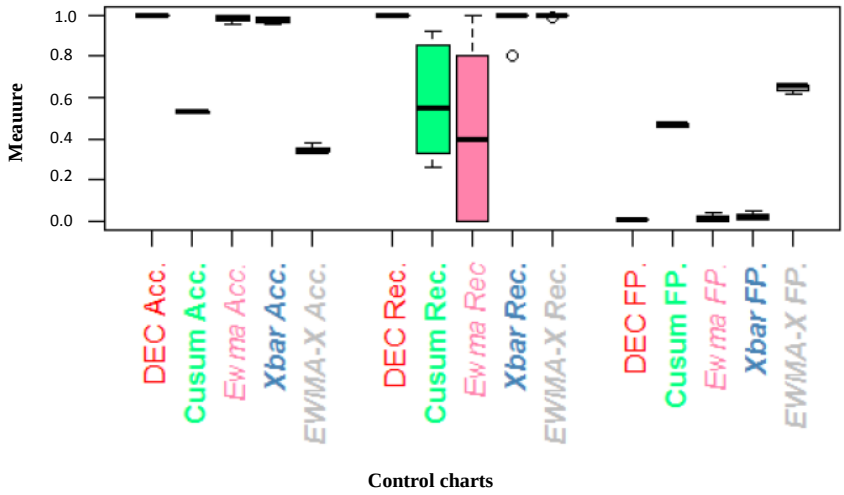

Figure 8.8: Performance evaluation of DEC model compared to individual EWMA, CUSUM and XBAR in terms of Accuracy, Recall, FP and FN rates: large shifts.

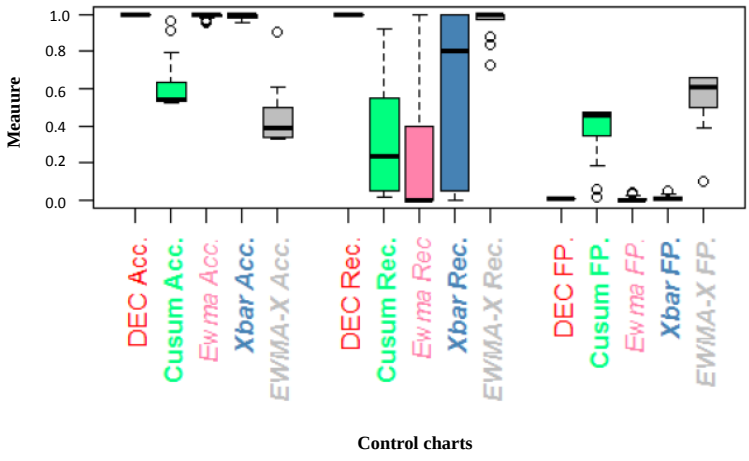

Figure 8.9: Performance evaluation of DEC model compared to individual EWMA, CUSUM and Xbar in terms of Accuracy, Recall, FP and FN rates: all shifts.

Chapter 9

Dynamic Weighted Majority CCs

Dynamics are fundamental properties of batch learning processes. Recently, monitoring dynamic processes has interested many researchers due to the importance of dealing with time-changing data stream processes in real world applications.

In this chapter, a DWM based identification model is proposed for monitoring small, large as well as covariate shift in non-stationary processes. The proposed method applies DWM ensemble method to aggregate the decision of different CCs which aim to improve single charts' performance and to reduce the risk of choosing a non adequate chart. The dynamic process model consists of an online monitoring of all shift ranges within a classification of shifts over time. This technique is unique since most methods use a learning based model without classifying the shift. Although some few works have been proposed for classifying the type of the shift, no one developed a classification based learning shift model which provides additional knowledge about the process behavior using an ensemble of charts instead of individual ones. Indeed, in order to improve the shift adaptation mode, a prediction of class label is used and greatly helps the model to classify the shift during the changing of the process toward the approximated right direction. The new proposed ensemble chart not only has the ability to deal with complex real datasets but also presents a concrete shift identification method based on classification learning technique of changes in non-stationary processes. This chapter is organized as follows: Section 1 describes the Dynamic Weighted Majority Control (DWMC) chart. Section 2 explains the methodology and the experiments. Finally, Section 3 concludes this chapter.

9.1 DWM ensemble-based CCs

Similarly to the general framework of learning classifier systems (LCS) used by **[Butz, 1995]** in machine learning, we propose a learning CC system (LCCS) for SPC. The LCS were first applied to *X* Classifier Sysytem (XCS) of **[Wilson, 1995]** by using an accuracy based fitness function for executing this mechanism with GA. However, this learning system can be extended

to other systems following the same framework. **[Asensio et al., 2014]** have recently used the LCS mechanism to propose a supervised neural constructivist system (SNCS). Their system uses an ensemble of multilayer perceptron (MLP) classifiers to learn from data stream where they have improved the quick adaptation capacity. In what follows, we first present our proposed DWM chart as a general learning chart system model in which the representation is quasi similar to LCS in terms of knowledge representation, knowledge discovery and feedback to the environment. The scheme representation for CCs instead of classifiers is shown in Figure (9.1).

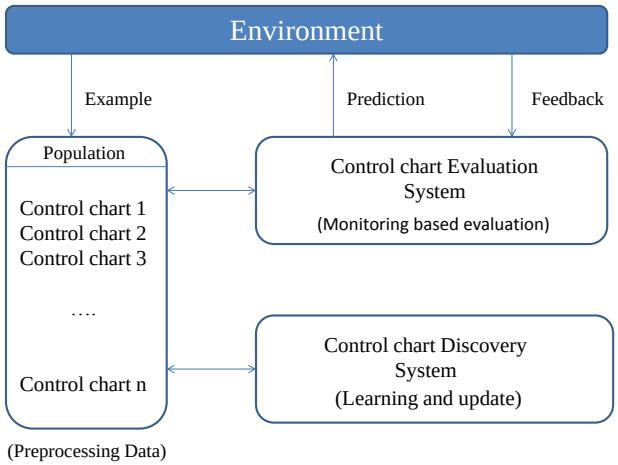

Figure 9.1: DWMC-based-LCCS mechanism.

DWMC chart develops an ensemble of CCs during its interaction with the environment. Our proposal is different from the basic LCS mechanism by the fact that it is a CC based learning method instead of classifier based learning model. Second, DWMC chart model presents an incremental learning model which dynamically adapts the charts to the changes in concepts. The new proposed method is adequate for real data application where the class labels are unknown thanks to a one step ahead prediction of the labels based on TACL.

Next, the different steps of DWMC chart based LCCS model are detailed.

9.1.1 CC knowledge representation

First of all, the simulated dataset is divided into n different batches. For $n > 1$, $n-1$ batches are used for training and the n^{th} batch is used for testing. During the first batch, TACL is applied to predict the unknown class label. TACL's role is to provide some initial information about the unknown class labels which helps the DWMC chart to apply the learning process based on DWM algorithm. The learning process is generated toward the same direction of the predicted class labels

$\widehat{y}$. Learning means updating the weights of the CCs in a way to minimize the errors between the global prediction of the ensemble and the real classes. To predict the class labels with TACL chart, 40 % of the batch size is used for training and the rest is used for testing. First, the CLs are kept fixed during the training phase whereas in the testing phase they are adjusted each time a number T of alarms are detected based on TACL procedure (line 2 of Algorithm (9.1)). Then, the output labels are determined as follows: when TACL signals an alarm, a value equal to 1 is assigned to the vector of labels, else a value equal to 0 is attributed (see lines 9, 10, 11 of Algorithm (9.1)). This mechanism is applied to all instances of the first batch.

In the second step, DWM chart uses an ensemble of different CCs chosen in a way to maximize the probability of detecting different shift ranges. For example, one can choose CUSUM, EWMA or XBAR chart as single elements of the ensemble. We can then vary the parameters of each single chart to have more variants of the ensemble. In the same way of detecting the state of the process in TACL, each CC's output is determined as follows: during the different instances of the batch, if the CC detects an out of control signal, the assigned chart's output is 1, otherwise the assigned chart's output is 0. Figure (9.2) illustrates the constructed data based on control charts and the true classes of the data. We note that these true classes are used only to evaluate the whole method but for the learning we only use the predicted classes $\widehat{y_i}$ as shown in red vertical lines in Figure (9.3). In this regard, for each new instance in the classification matrix (see line 12 of Algorithm (9.1)), each CC will have an assigned output to be compared to the predicted labels and continue the learning.

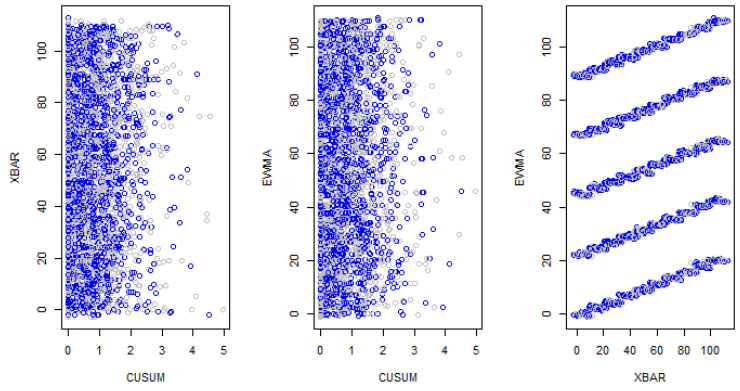

Figure 9.2: Illustration of simulated dataset after preprocessing, data is constructed based on features with EWMA, CUSUM and XBAR statistics, blue points represent the in control observations and grey points represent the out of control data.

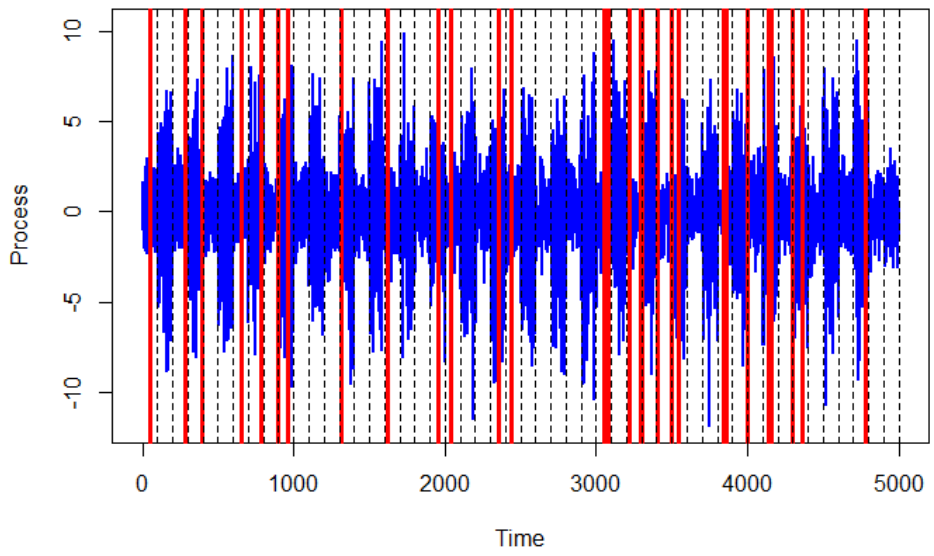

Figure 9.3: Ilustration of predicted class labels with TACL chart, red vertical lines present the predicted class labels which are out of control and the black dashed lines present the true out of control classes.

9.1.2 CC knowledge learning

In this state, a learning mechanism based on a weight adjustment procedure as well as an adaptation of the ensemble's elements inside the pool is applied. When the CC's output is equal to the predicted label $\widehat{y}_i$, its weight is increased. However when a CC's output is different from the predicted class label, the CC's weight is decreased. This procedure will continue during the whole first batch.

At the end of the batch, if a CC's weight was decreased many times until becoming smaller than a fixed threshold θ^c, then this chart is removed from the ensemble (line 20, 21 and 22 of Algorithm (9.1)). This is due to the fact that this CC doesn't contribute to the global performance of the algorithm during the learning process. In this regard, only CCs with weight larger than the threshold are maintained because of their good contribution to the ensemble's learning performance. Then, the global performance after the first batch is determined with a majority vote based on the weights of charts as well as their "vote" which is equal to 0 or 1 (line 23 of Algorithm (9.1)). The CC with the highest weighted majority vote is chosen as the global prediction in the monitored batch. Then DWMC chart compares the global prediction Λ^c with $\widehat{y}_i$. If they are equal, then it runs to the next batch, however, if are not, it adds a new CC (lines 24, 25 and 26 of Algorithm (9.1)).

Algorithm 9.1: Optimized DWM-WIN algorithm.

input : N: number of initialized charts
t: size of the batch
$\overrightarrow{\sigma}$: total sum of weighted predictions of each class
Λ: Global prediction of the ensemble
λ: local prediction of each individual classifiers
η, β, θ: parameters of DWM-WIN.
n^*: number of batches used to test the ensemble.

output: Concept drift detection points

for *(each batch)* **do**
  Step 1: Estimate the class label with TACL chart Step 2: Training Phase **for** *Preprocessing Data* **do**
    1. Apply N BaseCharts *on the first batch of data*;
    2. Compute the charting Statistics, the UCL and the LCL of each chart
    3. Detect the outliers detected with each CC
    4. Transform the monitoring process into a binary classification issue
    5. Assign 1 if the process is in control and 0 if the process out of control
    6. Construct the classification matrix data: $D_k = (\lambda_i, y_i)$ where λ_i is the prediction of each CC and y_i is the prediction of the unknown class label with TACL computed in Step 1.
  end
  Step 3: Control based on DWM chart
  for *i in 1: size of batches* **do**
    for *j in 1: N* **do**
      if $\lambda_{i,j} = y_i$ **then**
        $w_i = w_i \cdot \eta$ else $w_i = w_i \cdot \beta$
      end
    end
    $\sigma_\lambda = \sigma_\lambda + w_j$
    Normalize Weights
    if $w_i < \theta$ **then**
      remove one control chart j
    end
    Define the $\Lambda = argmax_j(\sigma_j)$
    if $\Lambda \neq y_i$ **then**
      AddNewChart with a weight $W_{i+1} = 1$
    end
  end
end
Go to the next batch

In this state, the adding of the CC can be done using different ways. Actually, we randomly choose a new CC from one ensemble constituted of three different charts but with different parameters. A weight of 1 is assigned to the new chart, then a normalization is applied to the weight of the charts remaining in the ensemble and the new one in order to avoid letting the new chart dominate the other ones (line 29). The same reasoning is applied to each new arriving batch. At each time step, the weight of the CCs is updated in an incremental way during one batch. The combination of the CCs at each batch is modeled in a way to optimize the CC based learning process.

9.1.3 CCs knowledge evaluation

CCs are evaluated at each time step and a local prediction is computed at each time for each CC. However, the global prediction is computed at the end of each batch of data. Evaluation is based on the error rates of type II and I and is computed as follows:

$$\text{Globalerror} = \text{error type I} + \text{error type II}$$

9.2 Methodology and experiments

This section investigates the performance of DWMC chart when monitoring and classifying out of control alarms of different shifts in non-stationary environment. All generated datasets for this experiment are presented in Figure (9.4) and their corresponding equations are detailed in Table (9.1).

As it is shown in Table (9.1), we first generate the results based on one dataset with abrupt change (D_1) to analyze the reaction of the ensemble chart model to such shift. Then, we apply it to the jumping mean dataset of **[Liu et al., 2013]** (D_2) and to three generated variants of this dataset by varying the parameters of the AR model of the original data, varying the noise level and finally by adding non-linearity issues ((D_3, D_4, and D_5 respectively). Moreover, we generated results based on the scaling variance dataset (D_6).

9.2.1 Change point identification based DWMC chart

The methodology used consists of dividing the data into n batches. Then 40 % of the data was used for training and the rest for testing. For all the experiments, we use the parameters set at: number of initialized charts $N = 3$, $\beta^c = 0.5$, $\theta^c = 0.2$, number of batches = 20 and 250 and for TACL, $\lambda = 0.1$ and $T = 1$.

We explore the ability of DWMC chart model to detect the change in the process behavior. In fact, the change point detected by DWMC chart model can rapidly be identified by the increase of the error rates each time a change point is identified. More specifically, plot (a) in Figure (9.5) shows the abrupt dataset process with the corresponding change point detection with DWMC chart. The process consists of 5000 instances with a shift in observation 2500. Thus, we choose

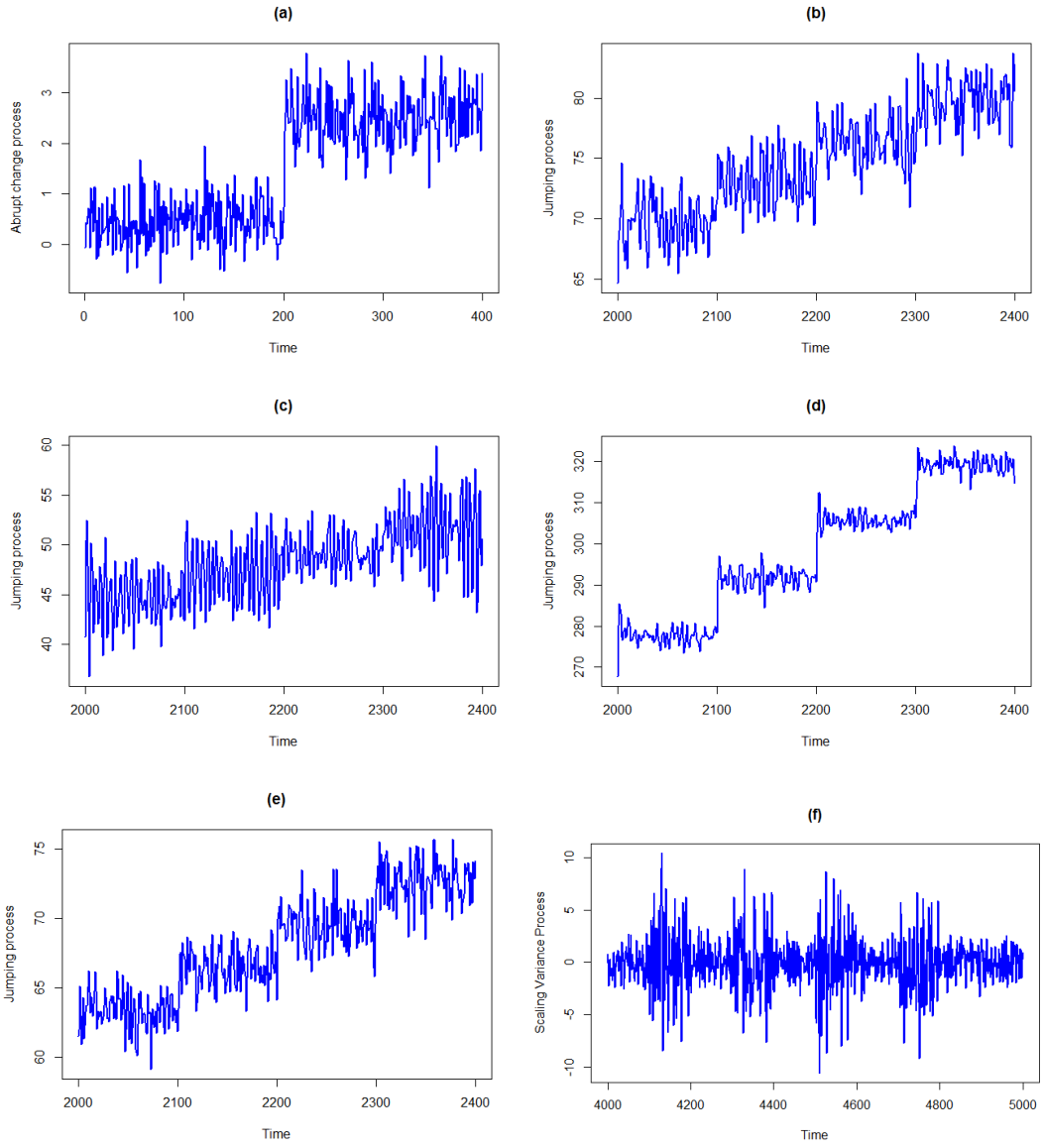

Figure 9.4: Jumping datasets with covariate shift where (a), (b), (c), (d), (e) and (f) represent respectively abrupt change process, jumping process, jumping process with modified parameters, jumping process with varying noise level, jumping process with nonlinearity and Scaling variance datasets.

to select 500 observations from the middle of the process of this dataset. We see that the abrupt change is easily detected by the proposed ensemble chart. For the other datasets, D_2, D_3, D_4, D_5 and D_6, we refer to **[Liu et al., 2013]** in presenting the last 5 change point detections. In fact, while using the jumping mean dataset, the authors mentioned that the 10 last change points are purposely

more significant than earlier ones in this dataset type. As we can see, the ensemble chart model has very good performance in identifying the different shift points. This is due to the learning procedure used by DWM algorithm which highlights the ability of learning the concept drift when it occurs during the process. Also, the use of the TACL chart as a class label prediction method is another factor which facilitates the change identification. The use of both heuristics allows the DWMC chart method to have a very good detection of the different types of data dynamics. Figure (9.5) illustrates the change point time detection of DWMC chart for the different behaviors of the dataset in the different variants of concept changes.

Table 9.1: Simulated datasets used in the experiments.

	Data Description	Equations
D_1	Dataset with abrupt change in the mean	Normal distribution process with a shift in the mean defined as follows: $\mu_1 = \mu_0 + \delta\ \sigma_0$, where δ is the shift size, μ_1 is the shifted mean, μ_0 and σ_0 are the initial mean and standard deviation of the process .
D_2	Covariate shift based on a samples of 5000 with different speeds **[Liu et al., 2013]**	$x(t) = \mathbf{0.6} \cdot (t\text{-}1)\text{-}\mathbf{0.5} \cdot (t\text{-}2) + \xi(t)$, Every 100 time steps, the noise mean at time t is: $\xi(t) \backsim N(\mu_N, \sigma)$: $\sigma = 1.5$ and If $N = 1$, $\mu_N = 0$, If $N = 2, \ldots, 49$, $\mu_N = \mu_{N-1} + N/\mathbf{16}$
D_3	Modified D_2	$x(t) = \mathbf{0.4}(t-1) - \mathbf{0.8}(t-2) + \xi(t)$
D_4	Jumping dataset with varying noise levels	$x(t) = \mathbf{0.6} \cdot (t\text{-}1)\text{-}\ \mathbf{0.5} \cdot (t\text{-}2) + \xi(t)$, Every 100 time steps, the noise mean at time t is: $\xi(t) \backsim N(\mu_N, \sigma)$: If $N = 1$, $\mu_N = 0$, If $N = 2, \ldots, 49$, $\mu_N = \mu_{N-1} + N/\mathbf{4}$
D_5	Jumping dataset with nonlinearity issues	$x(t) =$ $0.6\sqrt{\lvert t-1\rvert} - 0.5\sqrt{\lvert t-2\rvert} + \xi^{'}(t)$ where $\xi^{'}(t) \backsim N(\mu_N^{'}, \sigma)$: $\sigma = 1.5$ and If $N = 1$, $\mu_N^{'} = 0$, If $N = 2,\ldots, 49$, $\mu_N^{'} = \mu_{N-1}^{'} + N/\mathbf{4}$
D_6	Scaling Variance dataset **[Liu et al., 2013]**	$x(t) = \mathbf{0.6} \cdot (t\text{-}1)\text{-}\mathbf{0.5} \cdot (t\text{-}2) + \xi(t)$, Every 100 time steps, the noise standard deviation at time t is: $\xi(t) \backsim N(\mu^{'}, \sigma_N)$, where $\mu^{'} = 0$ and If $N = 1,3, \ldots, 49$, $\sigma_N = 1$, If $N = 2, 4, \ldots, 48$, $\sigma_N = log(e + N/4)$

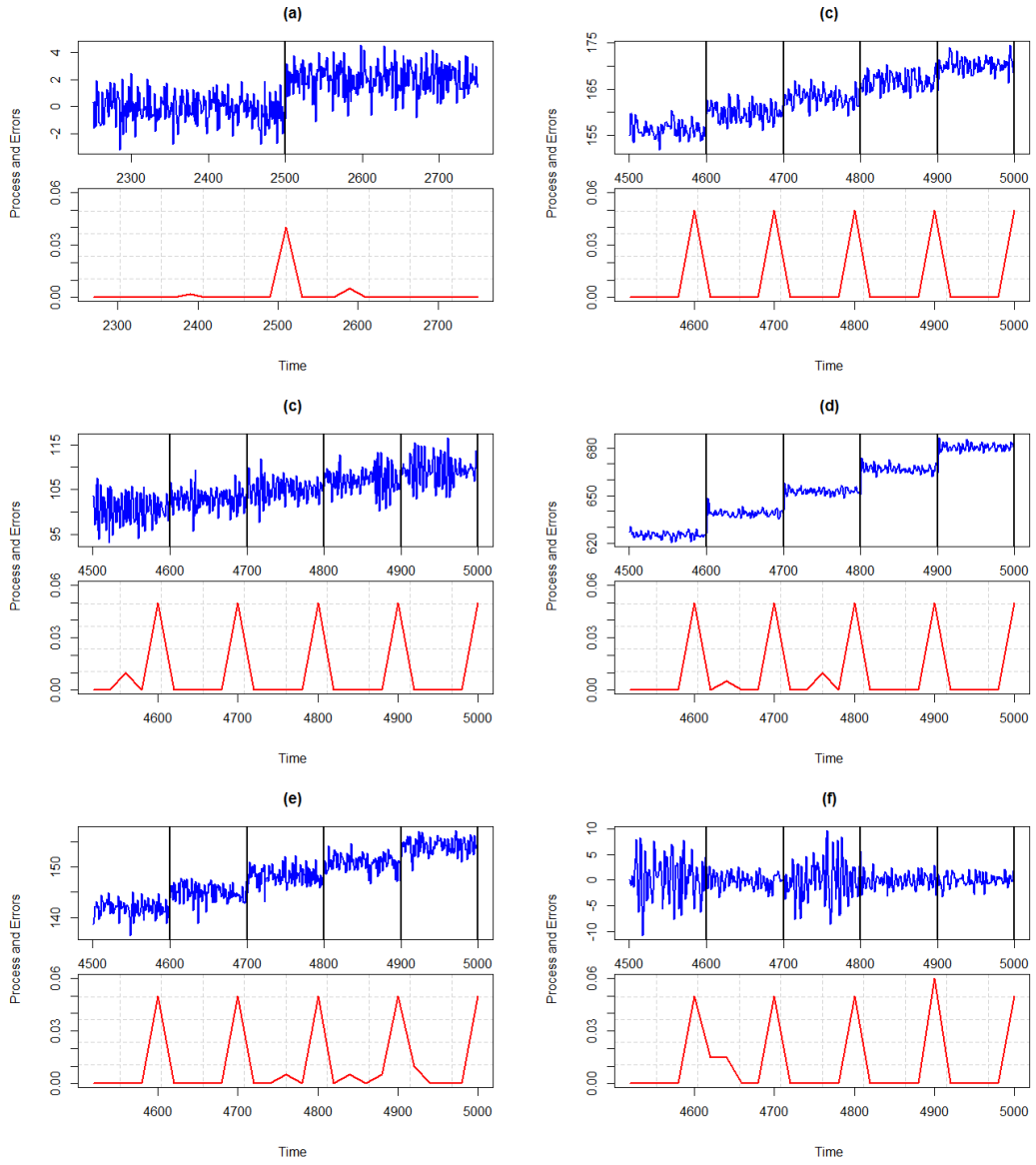

Figure 9.5: Illustration of time changing process with black vertical lines marking the true alarms (upper graphs) and the change point detection with DWM- based LCCS (lower graphs).

Next, we compare the performance of DWMC chart with individual chart models.

9.2.2 reduction of the error rates from the single chart to the ensemble of charts

Dataset with abrupt shift For D_1 dataset with abrupt change, we used number of batches=20 and 100 runs. Results representing the error mean in terms of number of batches in Figure (9.6) show that the ensemble outperforms the different charts in detecting the different simulated shift

range values. Although all charts have shown an increase during the time of the shift, the error of the DWMC chart is reduced compared to individual charts. Table (9.2) presents the mean and the standard deviation of the errors and the percentage of improvement of DWMC chart compared to EWMA, CUSUM and XBAR. For D_1 dataset, these charts have been improved with 75 %, 50 % and 39 % respectively by the DWMC chart. Results over the different datasets were tested with ANOVA test and the difference between the algorithms are statistically significant with an F statistic of 3.0563 and a p-value of 0.052.

For the rest of the datasets, we used number batches = 250 and 100 runs.

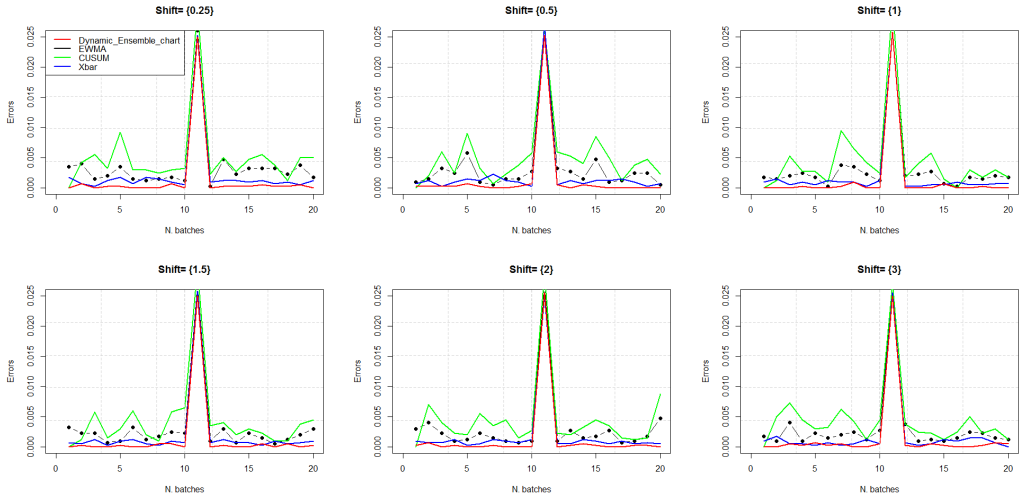

Figure 9.6: Error rates of DWMC chart based on D_1 dataset with abrupt change in the mean in terms of 20 batches and 100 runs.

Jumping dataset In D_2 dataset, consisting of the jumping mean dataset with a change every 100 observations, the errors of DWMC, CUSUM and XBAR are closely similar whereas the errors of EWMA are slightly higher. Concerning the reaction of the different methods to the shift which happens every 5 batches as shown in the first left plot of Figure (9.7), DWMC shows a high robustness to the problem of concept drift and a peak at each change characterizing the quick ability to detect changes in concepts. CUSUM and XBAR are also good and show better reaction capacity than EWMA in detecting the changes. For D_2 dataset, the performance of EWMA and CUSUM charts is improved by 84 % and 2.7 % respectively but no improvement is achieved for XBAR as shown in Table (9.2).

Jumping dataset with modified parameters Concerning D_3, DWMC chart shows a good robustness to changes in parameters and a peak at each time change characterizing the quick ability to correctly identify changes in concepts. CUSUM, XBAR are also good in maintaining the same reaction for different parameters based model. However, the error of EWMA in D_3 is somehow

smaller than in D_2 for these new parameter settings. EWMA is more sensitive to the change in the process parameters than other charts. As shown in Table (9.2), DWMC chart has improved EWMA and XBAR by 56.5 % and 32.8 % respectively but no improvement is achieved against CUSUM.

Jumping dataset with varying noise levels For D_4 dataset, a noise is added to the initial jumping dataset. DWMC, CUSUM and XBAR charts behave with high robustness being unaffected by the distorting effect of noise. EWMA has a small reaction so that the improvement of DWMC chart compared to EWMA has increased to 86.6 %. This is due to the sensitivity of the computed weighted charting statistics to the noise.

Jumping dataset with non-linearity issues In D_5, the dataset with nonlinearity issues, DWMC, XBAR and CUSUM behave with high robustness, being almost unaffected by the effect of non-linearity. This behavior is more noticeable with DWMC. In this dataset, DWMC has achieved an improvement of 65 % whereas no improvement is achieved compared to CUSUM and XBAR. Robustness against non-linearity issues are a desirable feature for CCs.

Scaling Variance dataset Lastly, in D_6 dataset with scaling variance, EWMA and CUSUM have higher errors compared to XBAR and DWMC in D_2. DWMC's improvement is 84.9 %, 31 % and 89 % compared to EWMA, CUSUM and XBAR charts. Boxplot presented in Figure (9.9) shows that the differences in errors comes from EWMA. The first benefit of the ensemble based chart is that it presents a precise change point detection method and second it outperforms the cases where one specific CC is a bad choice.

Furthermore, Figure (9.8) shows that DWMC chart is better or as good as the best individual chart for the different data sets. For D_1 dataset, DWMC chart is as good as XBAR and better than CUSUM and EWMA chart. For D_2 dataset, DWMC chart is as good as CUSUM and better than EWMA and XBAR charts. For the Jumping mean dataset with modified parameter, DWMC chart is as good as CUSUM and better than EWMA and XBAR. Furthermore, DWMC chart is more robust to the effect of noise than other individual charts. For the effect of nonlinearity, DWMC chart is as good as CUSUM and better than EWMA and XBAR charts. In scaling variance dataset, DWMC chart is as good as XBAR and better than EWMA and CUSUM.

9.3 Friedman test

DWMCC versus individual control charts: For the abrupt change and the jumping datasets the difference between the methods is statistically significant based on an ANOVA test with a p-value $< 2.2 \cdot 10^{-16}$. In Jumping dataset with modified parameters and Jumping dataset with varying noise levels, the difference between methods is also significant with a p-value = $8.615 \cdot 10^{-15}$ and $< 2.2 \cdot 10^{-16}$ respectively. For the Jumping dataset with non-linearity issues, results show

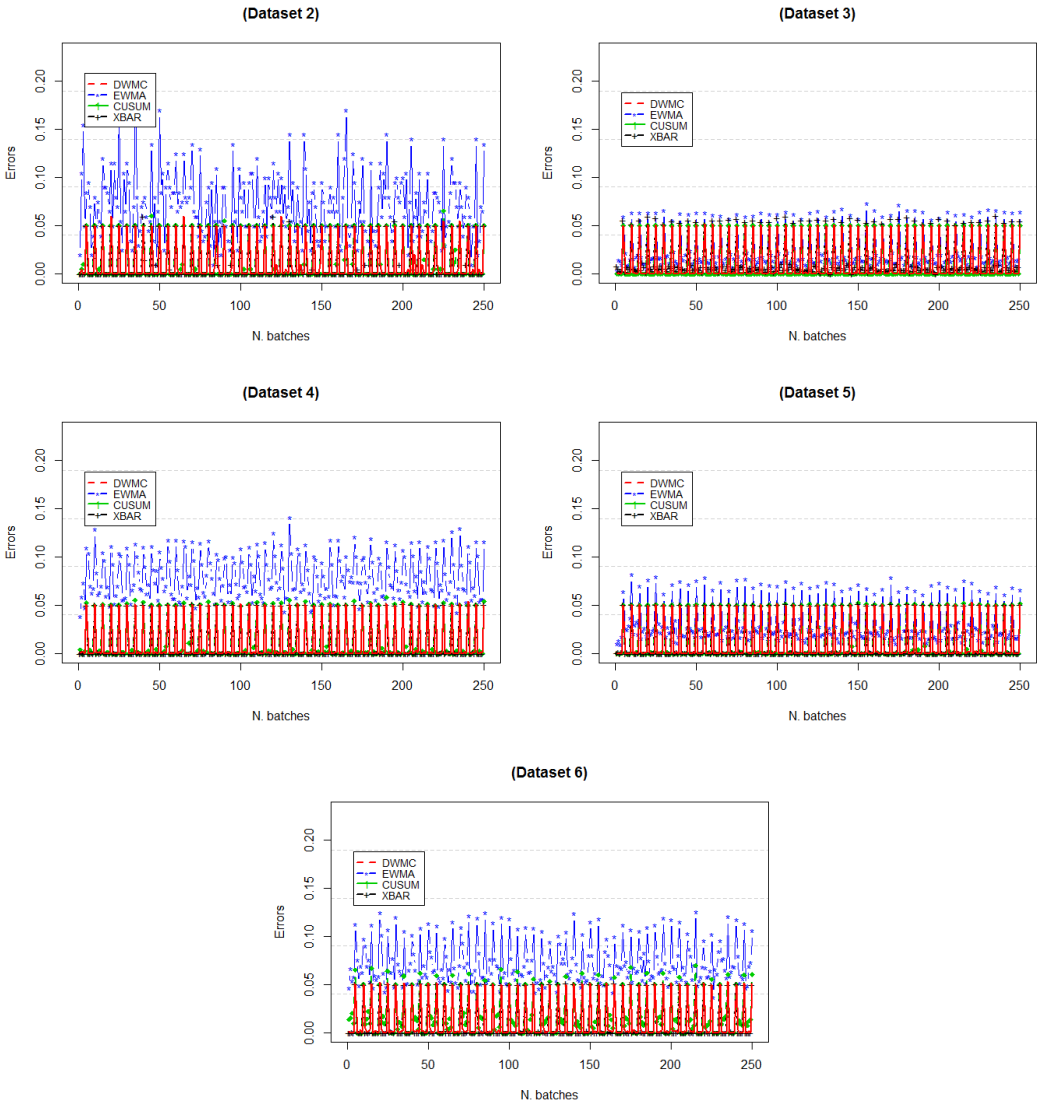

Figure 9.7: Error rates of DWMC and individual charts in D_2, D_3, D_4, D_5 and D_6.

that the difference between methods is significant based on a p-value = $3.064 \cdot 10^{-6}$. Finally, methods perform differently based on the Scaling Variance dataset, this is shown based on the p-value which is $< 2.2 \cdot 10^{-16}$.

DWMCC versus the best individual chart: For more advanced analysis, we apply the ANOVA test to the proposed method and the best control chart in each dataset. Based on Friedman's test we accept the null hypothesis that the two methods perform similarly. This result confirms that the ensemble chart model is as good as the best control chart and more interestingly is that it cannot be in any case more worse than the best individual chart.

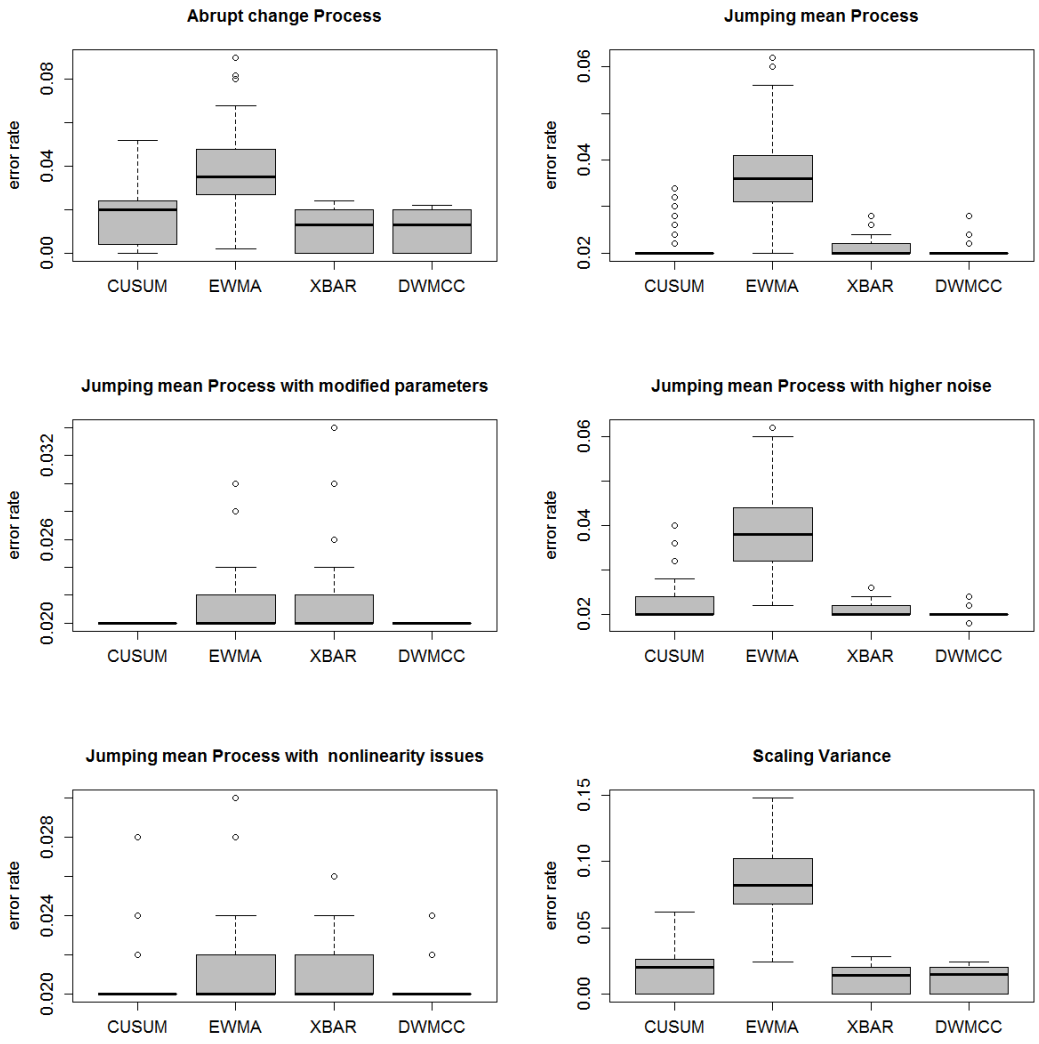

Figure 9.8: Boxplot representing the error rates of DWMCC chart and individual ones based on D_1, D_2, D_3, D_4, D_5 and D_6.

DWMCC versus the worst individual chart: We also compare the proposed method with the worst individual control chart. Based on a p-value of 0.0045 at 1% level, the Friedman test rejects the null hypothesis that the DWMC chart performs as good as the worst individual chart. Hence, indeed that our proposed ensemble chart is better or as good as the best individual chart, it is never more worse than the the worst individual chart. Using the ensemble chart model not only allows maintaining or improving the performance of the best individual chart as well as it prevents from using a non adequate control chart.

Table 9.2: The mean and standard deviation of the error values of DWMC based LCCS, EWMA, CUSUM and XBAR charts.

Datasets	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
DWMC mean	0.001275	0.0107	0.0103	0.01063	0.0105	0.0109
DWMC sd	(0.00558)	(0.02)	(0.02)	(0.02)	(0.02)	(0.0199)
EWMA mean	0.00518	0.071	0.0237	0.0795	0.03085	0.0726
EWMA sd	(0.00313)	(0.021)	(0.0199)	(0.0222)	(0.0205)	(0.0219)
CUSUM mean	0.00255	0.011	0.01	0.011	0.0104	0.0158
CUSUM sd	(0.055)	(0.0201)	(0.02)	(0.0202)	(0.02)	0.0209
XBAR mean	0.0021	0.0103	0.01533	0.0103	0.0105	0.01038
XBAR sd	(0.0057)	(0.02)	(0.02)	0.02	(0.02)	(0.02)
DWMC vs EWMA	**75** %	**84** %	**56.5** %	**86.6** %	**65** %	**84.9** %
DWMC vs CUSUM	**50** %	**2.7** %	-3 %	**3.36** %	-0.0096 %	**31** %
DWMC vs XBAR	**39** %	-3.8 %	**32.8** %	-3.2 %	**0**	**89** %

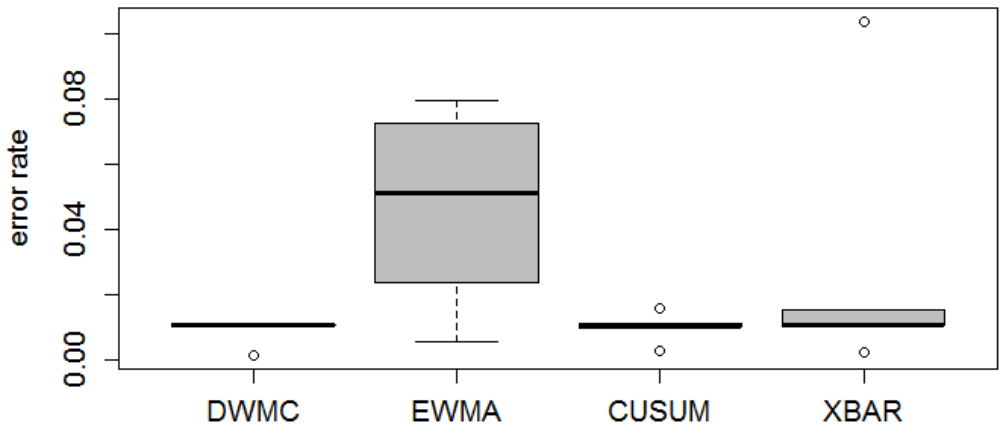

Figure 9.9: Boxplot comparing the mean over the different datasets of the error of the combined method and individual ones.

9.4 Conclusion

For controlling nonstationary processes with different issues of the data stream, a DWMC is proposed. It combines individual charts based on ensemble methods with a batch learning process to monitor small and large shifts simultaneously and to cope with different forms of concept drift. The proposed method consists of three steps: first, transforming the task of determining the state of the process into a classification problem by treating CCs as attributes of the data where the drift has to be predicted. Second, DWM-WIN mechanism is applied to learn the shift and to combine the decision of the different individuals. Third, the learning process is directed within a one ahead step prediction based on TACL chart. The proposed model does not only exhibit superior robustness to individual charts but also presents a new heuristic for shift learning and batch monitoring in nonstationary environment with a quick adaptation to complex issues of the data.

Chapter 10

Summary and Further Research

In this dissertation we were interested in improving DWM algorithm using online machine learning to improve SPC methods performance in non stationary environment. The most important results of this dissertation are provided in Section 1. Outlooks developed to enhance and extend this research are discussed in Section 2.

10.1 Summary

Part I of this thesis investigates the optimization of DWM method as well as the analysis of its robustness against many issues. Chapter 2 is devoted to DWM-WIN method which improves the DWM by considering the stability of the classifiers in the ensemble as well as their age and history during the learning process as a new criterion for finding the best ensemble of classifiers which copes the best with the learned data. Results assessed in Chapter 3 show that the use of these new criterion can effectively improve the performance and the speed of DWM algorithm. In order to deal with the problem of parameter selection, a GA was used in Chapter 4 to find the best values that should be chosen. The advantage of the DWM-WIN-GA method is to automate the parameter selection inside the DWM-WIN algorithm. Then, Chapter 5 is dedicated to the assessment of DWM-WIN in SEA dataset. The impact of permutation, nonlinearity and noise among other issues were evaluated based on SEA dataset with concept drift. Results show that DWM-WIN has a very good robustness to concept drift and that an ensemble of naiveBayes is better than an ensemble based on rpart or kknn and that the noise level involves a significant difference between the methods.

In part II, we investigate two new time adjusting CCs for shift detection in non stationary processes. In Chapter 6, we first propose TACL chart. The advantage of this CC is that it updates its statistics as well as the CLs after a new detected concept of newly arriving dataset. Also, we proposed another improved version of TACL, the TS-TACL chart, which reduces the number of false

detections identified in the first stage using the Kolmogorov Simirnov test of stationarity during stage *II* to validate the detected out of control in Stage *I*. Results evaluated in Chapter 7 show that TACL and TS-TACL can effectively detect concept drift in nonstationary environment. In dataset with abrupt change, TS-TACL reports results as the best chart in terms of FN and Recall. The best Accuracy measure was obtained by TACL and TS-TACL in jumping mean dataset. In SEA dataset with problem of concept drift, both proposed charts have shown better ability to reduce the false detections and to improve the recall measure. Additionally, we proposed learning from the error rates of DWM-WIN method. The advantage of this procedure is to benefit from the additional knowledge of the misclassification error rates of DWM-WIN which improve the shift identification when monitoring with TACL and TS-TACL.

Part III of this dissertation deals with the combination of CCs with the DWM-WIN method. In Chapter 8, the "DEC" chart was first proposed by aggregating the decisions of three different CCs based on DWM method. Results show that the use of the ensemble techniques in CCs outperforms the use of single CCs as well as other combined charts when several shift levels have to be monitored in terms of different measures thanks to the fact that the use of DWM in CCs allows an identification of the shift learning process. In order to enable the application of combined charts to real world data, a new model based on both DWM-WIN as well as TACL chart was proposed. In fact, DWM was used in learning while monitoring with CCs by applying exactly the same mechanism dealing with classifiers in the DWM to CCs. Results show that using this new heuristic of treating CCs as classifiers by adding, removing and dynamic weighting of CCs based on their performance leads to a better shift identification ability. Furthermore, a solution for predicting the unknown class labels representing the state of the process was done using TACL chart. Results show that the dynamic weighting majority chart model leads to the best model to deal with the non stationarity of the data compared to models from the most successful individual SPC techniques. The novelty of Chapter 9 is that it presents the first enhancement of ensemble methods in CCs as well as that this method was explored by the use of TACL chart to optimize the handling of online dynamic real non stationary data with concept drift in a new context of a shift learning monitoring process.

10.2 Outlooks

The research conducted in this dissertation has emphasized several topics on which an enhancement would be beneficial.

The results shown for DWM-WIN might be further investigated. A reinforcement of the learning by adding adequate new classifiers in the ensemble of DWM-WIN would be very interesting. Moreover, further research about the performance on other datasets for concept drift could be investigated. There are also several possible developments and applications of the research results about time adjusting CCs. One extension is to investigate the relation between the CLs adjust-

ment and the recurring concept identification. This could be done by either checking the tradeoff between CC's stability and the frequency of recurring concepts occurrence or by adding new criterion in the adjustment condition. Indeed, due to the importance of some parameters in the time adjusting charts, self configuration of the parameters of the adaptive charts using optimization methods is very important. Furthermore, the new DWMC chart presenting the first enhancement of ensemble methods in SPC open up lines for future work. An interesting research related to this that might be investigated is to use a larger and diverse set of CCs to be combined and also to optimize the choice of the selected charts to be added in the ensemble. This can be done by one of the optimization methods such as Particle swarm optimization or Model based optimization. The stability of the CCs in the ensemble as well as their age and their learning history presents an interesting research where more interpretability can be investigated. Another interesting enhancement is to apply the DWM CC for many features. Accordingly, instead of applying many CCs to one dataset and combining the decision over the different charts, one can apply one CC to different features of the data, then combine the decision over the different features decisions. This would represent a new multivariate application of ensemble methods in SPC.

Chapter 11

Appendix

	TACL					SD-EWMA				
	Acc.	FP	FN.	F.M	Recall	Acc	FP	FN	F.M	Recall
Perm.										
Perm 1	0.98	0.01	0.166	0.833	0.833	**0.96**	0.032	0	0.8	1
Perm 2	0.97	0.031	0	0.8	1	**0.94**	0.043	0.166	0.66	0.833
Perm 3	0.95	0.042	0.166	0.66	0.833	0.95	0.0322	0.166	0.71	0.833
Perm 4	0.95	0.032	0.166	0.714	0.833	0.96	0.021	0.166	0.76	0.833
Perm 5	0.97	0.021	0.166	0.769	0.833	0.96	0.021	0.166	0.76	0.833
Perm 6	0.97	0.021	0.166	0.769	0.833	0.96	0.021	0.166	0.76	0.833
Perm 7	0.969	0.021	0.166	0.769	0.833	0.96	0.021	0.166	0.76	0.833
Perm 8	0.97	0.021	0.166	0.769	0.833	0.96	0.021	0.166	0.76	0.833
Perm 9	0.969	0.021	0.166	0.76	0.833	0.94	0.043	0.166	0.66	0.833
Perm 10	0.97	0.021	0.166	0.76	0.833	0.95	0.032	0.166	0.71	0.833
Perm 11	0.97	0.021	0.166	0.76	0.833	0.95	0.043	0	0.75	1
Perm 12	0.97	0.021	0.166	0.769	0.833	0.96	0.032	0	0.8	1
Perm 13	0.97	0.021	0.166	0.769	0.833	0.96	0.032	0	0.8	1
Perm 14	0.97	0.021	0.166	0.769	0.833	0.95	0.043	0	0.75	1
Perm 15	0.97	0.021	0.166	0.769	0.833	0.95	0.043	0	0.75	1
Perm 16	0.97	0.021	0.166	0.769	0.833	0.94	0.053	0	0.7	1
Perm 17	0.97	0.021	0.166	0.769	0.833	0.96	0.032	0	0.8	1
Perm 18	0.97	0.021	0.166	0.769	0.833	0.96	0.032	0	0.8	1
Perm 19	0.97	0.021	0.166	0.769	0.833	0.95	0.032	0.166	0.71	0.833
Perm 20	0.97	0.021	0.166	0.769	0.833	0.95	0.032	0.166	0.71	0.833
Perm 21	0.97	0.021	0.166	0.769	0.833	0.95	0.043	0	0.75	1

Perm 22	0.97	0.021	0.166	0.76	0.833	0.95	0.032	0.166	0.71	0.833
Perm 23	0.97	0.021	0.166	0.76	0.833	0.95	0.032	0.166	0.71	0.833
Perm 24	0.97	0.021	0.166	0.769	0.833	0.97	0.021	0	0.75	1
Mean	**0.967**	**0.03**	**0.159**	**0.764**	**0.839**	**0.953**	**0.03**	**0.086**	**0.743**	**0.909**
Sd	**0.007**	**0.038**	**0.033**	**0.029**	**0.034**	**0.07**	**0.009**	**0.084**	**0.04**	**0.084**

Table 11.1: *Results of monitoring with TACL and SD-EWMA charts based on error rates of DWM-WIN based kknn algorithm based on 100 runs, noise = 10%*

	TS-TACL					TSSD-EWMA				
	Acc.	FP	FN.	F.M	Recall	Acc	FP	FN	F.M	Recall
Perm.										
Perm 1	0.99	0	0.166	0.909	0.833	0.98	0.0107	0	0.92	1
Perm 2	0.99	0	0	0.923	1	0.98	0	0.166	0.909	0.833
Perm 3	0.98	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 4	0.99	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 5	0.99	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 6	0.99	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 7	0.99	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 8	0.99	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 9	0.99	0	0.166	0.909	0.833	0.97	0.0107	0.166	0.833	0.833
Perm 10	0.99	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 11	0.99	0	0.166	0.909	0.833	1	0	0	1	1
Perm 12	0.99	0	0.166	0.909	0.833	1	0	0	1	1
Perm 13	0.99	0	0.166	0.909	0.833	1	0	0	0	1
Perm 14	0.99	0	0.166	0.909	0.833	1	0	0	1	1
Perm 15	0.99	0	0.166	0.909	0.833	1	0	0	1	1
Perm 16	0.99	0	0.166	0.909	0.833	0.98	0.01	0	0.92	1
Perm 17	0.99	0	0.166	0.909	0.833	1	0	0	1	1
Perm 18	0.99	0	0.166	0.909	0.833	1	0	0	1	1
Perm 19	0.99	0	0.166	0.909	0.833	0.989	0	0.166	0.909	0.833
Perm 20	0.99	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 21	0.99	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 22	0.99	0	0.166	0.909	0.833	0.98	0	0.166	0.909	0.833
Perm 23	0.99	0	0.166	0.909	0.833	0.97	0.0107	0.166	0.833	0.833
Perm 24	0.99	0	0.166	0.909	0.833	1	0	0	1	1
Mean	**0.99**	**0**	**0.159**	**0.909**	**0.839**	**0.985**	**0.0017**	**0.092**	**0.89**	**0.906**
Sd	**0**	**0**	**0.033**	**0.0028**	**0.034**	**0.01**	**0.004**	**0.084**	**0.188**	**0.084**

Table 11.2: Results of monitoring with TS-TACL and TSSD-EWMA charts based on error rates of DWM-WIN based kknn algorithm based on 100 runs, noise = 10%

	TACL					SD-EWMA				
	Acc.	FP	FN.	F.M	Recall	Acc	FP	FN	F.M	Recall
Perm.										
Perm 1	0.959	0.043	0	0.75	1	**0.95**	0.043	0	0.75	1
Perm 2	0.959	0.043	0	0.75	1	**0.94**	0.053	0	0.705	1
Perm 3	0.959	0.043	0	0.75	1	**0.95**	0.043	0	0.75	1
Perm 4	0.959	0.043	0	0.75	1	**0.97**	0.021	0	0.85	1
Perm 5	0.959	0.043	0	0.75	1	**0.95**	0.032	0.166	0.714	0.833
Perm 6	0.959	0.043	0	0.75	1	**0.93**	0.053	0.166	0.625	0.833
Perm 7	0.959	0.043	0	0.75	1	**0.95**	0.043	0	0.75	1
Perm 8	0.959	0.043	0	0.75	1	**0.95**	0.043	0	0.75	1
Perm 9	0.959	0.043	0	0.75	1	**0.95**	0.043	0	0.75	1
Perm 10	0.959	0.043	0	0.75	1	**0.89**	0.107	0	0.54	1
Perm 11	0.959	0.043	0	0.75	1	**0.9**	0.053	0	0.705	1
Perm 12	0.959	0.043	0	0.75	1	**0.94**	0.053	0	0.705	1
Perm 13	0.959	0.043	0	0.75	1	**0.94**	0.053	0	0.705	1
Perm 14	0.959	0.043	0	0.75	1	**0.94**	0.053	0	0.705	1
Perm 15	0.959	0.043	0	0.75	1	**0.94**	0.053	0	0.705	1
Perm 16	0.959	0.043	0	0.75	1	**0.94**	0.053	0	0.705	1
Perm 17	0.959	0.043	0	0.75	1	**0.97**	0.021	0	0.85	1
Perm 18	0.959	0.043	0	0.75	1	**0.97**	0.021	0	0.85	1
Perm 19	0.959	0.043	0	0.75	1	**0.98**	0.01	0	0.923	1
Perm 20	0.959	0.043	0	0.75	1	**0.95**	0.032	0.166	0.714	0.833
Perm 21	0.959	0.043	0	0.75	1	**0.95**	0.032	0.166	0.714	0.833
Perm 22	0.959	0.043	0	0.75	1	**0.97**	0.01	0.166	0.833	0.833
Perm 23	0.959	0.043	0	0.75	1	**0.96**	0.032	0	0.8	1
Perm 24	0.959	0.043	0	0.75	1	**0.97**	0.021	0	0.85	1
Mean	**0.959**	**0.043**	**0**	**0.75**	**1**	**0.949**	**0.04**	**0.055**	**0.747**	**0.965**
Sd	**0**	**0**	**0**	**0**	**0**	**0.18**	**0.02**	**0.116**	**0.082**	**0.069**

Table 11.3: *Results of monitoring with TACL and SD-EWMA charts based on error rates of DWM-WIN based naive Bayes algorithm based on 100 runs, noise = 10%*

	TS-TACL					TSSD-EWMA				
	Acc.	FP	FN.	F.M	Recall	Acc	FP	FN	F.M	Recall
Perm.										
Perm 1	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 2	0.989	0.01	0	0.92	1	**0.97**	0.021	0	0.85	1
Perm 3	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 4	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Perm 5	0.989	0.01	0	0.92	1	**0.98**	0	0.166	0.909	0.833
Perm 6	0.989	0.01	0	0.92	1	**0.97**	0.01	0.166	0.833	0.833
Perm 7	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 8	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 9	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 10	0.989	0.01	0	0.92	1	**0.95**	0.01	0.5	0.6	0.5
Perm 11	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 12	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Perm 13	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Perm 14	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 15	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 16	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 17	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Perm 18	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Perm 19	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Perm 20	0.989	0.01	0	0.92	1	**0.98**	0	0.166	0.909	0.833
Perm 21	0.989	0.01	0	0.92	1	**0.98**	0	0.166	0.909	0.833
Perm 22	0.989	0.01	0	0.92	1	**0.98**	0	0.166	0.909	0.833
Perm 23	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Perm 24	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Mean	**0.989**	**0.01**	**0**	**0.92**	**1**	**0.984**	**0.0059**	**0.055**	**0.92**	**0.944**
Sd	**0**	**0**	**0**	**0**	**0**	**0.012**	**0.006**	**0.116**	**0.084**	**0.117**

Table 11.4: *Results of monitoring with TSTACL and TSSD-EWMA charts based on error rates of DWM-WIN based NaiveBayes algorithm based on 100 runs, noise = 10%*

	TACL					SD-EWMA				
	Acc.	FP	FN.	F.M	Recall	Acc	FP	FN	F.M	Recall
Perm.										
Perm 1	0.969	0.032	0	0.8	1	**0.96**	0.032	0	0.8	1
Perm 2	0.969	0.032	0	0.8	1	**0.93**	0.064	0	0.66	1
Perm 3	0.969	0.032	0	0.8	1	**0.95**	0.043	0	0.75	1
Perm 4	0.969	0.032	0	0.8	1	**0.95**	0.032	0.166	0.71	0.833
Perm 5	0.969	0.032	0	0.8	1	**0.95**	0.021	0.33	0.66	0.66
Perm 6	0.969	0.032	0	0.8	1	**0.93**	0.34	0.33	0.57	0.66
Perm 7	0.969	0.032	0	0.8	1	**0.95**	0.04	0	0.75	1
Perm 8	0.969	0.032	0	0.8	1	**0.94**	0.043	0.166	0.66	0.833
Perm 9	0.969	0.032	0	0.8	1	**0.969**	0.021	0.166	0.76	0.833
Perm 10	0.969	0.032	0	0.8	1	**0.96**	0.021	0.166	0.769	0.833
Perm 11	0.969	0.032	0	0.8	1	**0.95**	0.043	0	0.75	1
Perm 12	0.969	0.032	0	0.8	1	**0.95**	0.043	0	0.75	1
Perm 13	0.969	0.032	0	0.8	1	**0.95**	0.043	0	0.75	1
Perm 14	0.969	0.032	0	0.8	1	**0.95**	0.043	0	0.75	1
Perm 15	0.969	0.032	0	0.8	1	**0.95**	0.043	0	0.75	1
Perm 16	0.969	0.032	0	0.8	1	**0.95**	0.043	0	0.75	1
Perm 17	0.969	0.032	0	0.8	1	**0.95**	0.043	0	0.75	1
Perm 18	0.969	0.032	0	0.8	1	**0.95**	0.043	0	0.75	1
Perm 19	0.969	0.032	0	0.8	1	**0.959**	0.032	0.166	0.714	0.833
Perm 20	0.969	0.032	0	0.8	1	**0.95**	0.032	0.166	0.714	0.833
Perm 21	0.969	0.032	0	0.8	1	**0.96**	0.032	0	0.8	1
Perm 22	0.969	0.032	0	0.8	1	**0.95**	0.032	0.166	0.714	0.833
Perm 23	0.969	0.032	0	0.8	1	**0.95**	0.032	0.166	0.714	0.833
Perm 24	0.969	0.032	0	0.8	1	**0.949**	0.053	0	0.7	1
Mean	**0.969**	**0.032**	**0**	**0.8**	**1**	**0.949**	**0.038**	**0.082**	**0.726**	**0.916**
Sd	**0**	**0**	**0**	**0**	**0**	**0.0077**	**0.0099**	**0.109**	**0.05**	**0.111**

Table 11.5: *Results of monitoring with TACL and SD-EWMA charts based on error rates of DWM-WIN based on rpart algorithm based on 100 runs, noise = 10%*

	TS-TACL					TSSD-EWMA				
	Acc.	FP	FN.	F.M	Recall	Acc	FP	FN	F.M	Recall
Perm.										
Perm 1	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 2	0.989	0.01	0	0.92	1	**0.96**	0.032	0	0.8	1
Perm 3	0.989	0.01	0	0.92	1	**0.97**	0.021	0	0.85	1
Perm 4	0.989	0.01	0	0.92	1	**0.97**	0.01	0.166	0.833	0.833
Perm 5	0.989	0.01	0	0.92	1	**0.96**	0.01	0.33	0.72	0.66
Perm 6	0.989	0.01	0	0.92	1	**0.95**	0.021	0.33	0.66	0.66
Perm 7	0.989	0.01	0	0.92	1	**0.96**	0.021	0.166	0.769	0.833
Perm 8	0.989	0.01	0	0.92	1	**0.96**	0.021	0.166	0.769	0.833
Perm 9	0.989	0.01	0	0.92	1	**0.98**	0	0.166	0.909	0.833
Perm 10	0.989	0.01	0	0.92	1	**0.989**	0	0.166	0.909	0.833
Perm 11	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 12	0.989	0.01	0	0.92	1	**0.98**	0	0.166	0.909	0.833
Perm 13	0.989	0.01	0	0.92	1	**0.98**	0	0.166	0.909	0.833
Perm 14	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.923	1
Perm 15	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Perm 16	0.989	0.01	0	0.92	1	**1**	0	0	1	1
Perm 17	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Perm 18	0.989	0.01	0	0.92	1	**0.989**	0.01	0	0.923	1
Perm 19	0.989	0.01	0	0.92	1	**0.97**	0.01	0.166	0.833	0.959
Perm 20	0.989	0.01	0	0.92	1	**0.97**	0.01	0.166	0.833	0.833
Perm 21	0.989	0.01	0	0.92	1	**0.97**	0.01	0.166	0.833	0.833
Perm 22	0.989	0.01	0	0.92	1	**0.97**	0.01	0.166	0.833	0.833
Perm 23	0.989	0.01	0	0.92	1	**0.97**	0.01	0.166	0.833	0.833
Perm 24	0.989	0.01	0	0.92	1	**0.98**	0.01	0	0.92	1
Mean	**0.989**	**0.01**	**0**	**0.92**	**1**	**0.97**	**0.01**	**0.11**	**0.863**	**0.88**
Sd	**0**	**0**	**0**	**0**	**0**	**0.011**	**0.008**	**0.105**	**0.082**	**0.1**

Table 11.6: *Results of monitoring with TS-TACL and TSSD-EWMA charts based on error rates of DWM-WIN based on rpart algorithm based on 100 runs, noise = 10%*

Bibliography

[Ahmed et al., 1999] Ahmed, S. N., Huan, L., and Kay, S. K. (1999). Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 317–321, New York, NY, USA. ACM.

[Akorede et al., 2011] Akorede, M. F., Hizam, H., bin Aris, I., and Kadir, M. Z. A. A. (2011). Effective method for optimal allocation of distributed generation units in meshed electric power systems. *Transmission and Distribution*, 5:276–287.

[Aljahdali and Telbany, 2008] Aljahdali, S. H. and Telbany, M. E. S. E. (2008). Genetic algorithms for optimizing ensemble of models in software reliability, prediction. *ICGST AIML Journal*, 8:5–13.

[Asensio et al., 2014] Asensio, A. S., Puig, A. O., and Golobardes, E. (2014). Robust on line neural learning classifier system for data stream classification tasks. *Journal of Soft Computing*, 18(8):1441–1461.

[Asuncion and Newman, 2007] Asuncion, A. and Newman, D. J. (2007). Uci machine learning repository. University of California, Irvine, School of Information and Computer Sciences,.

[Bifet et al., 2010a] Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010a). Accurate ensembles for data streams: Combining restricted hoeffding trees using stacking. In 2^{nd} *Asian Conference on Machine Learning (ACML)*, pages 225–240, Tokyo, Japan. JMLR, Workshop and Conference Proceedings 13.

[Bifet et al., 2010b] Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010b). Moa: Massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604.

[Bischl and Richter, 2014] Bischl, B., M. L. and Richter, J. (2014). mlr: Machine learning in r.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Journal of Machine Learning*, 45:5–32.

[Breiman et al., 1984] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. 9780412048418. Wadsworth, Belmont, CA.

[Brillman et al., 2005] Brillman, J. C., Burr, T., Forslund, D., Joyce, E., Picard, R., and Umland, E. (2005). Modeling emergency department visit patterns for infectious disease complaints: results and application to disease surveillance. *BMC medical informatics and decision making*, 5:4.

[Budin, 1996] Budin, L., G. M. B. A. (1996). Traditional techniques of genetic algorithms applied to floating-point chromosome representations. In *KoREMA 96,* 41^{96} *Annual Conference*, pages 93–96, Opatija.

[Butz, 1995] Butz, M. V. (1995). Rule-based evolutionary online learning systems- a principal

approach to lcs analysis and design. volume 191.

[Cheng and Cheng, 2008] Cheng, C. S. and Cheng, H. P. (2008). Identifying the source of variance shifts in the multivariate process using neural networks and support vector machines. *Expert Systems with Applications*, 35(1-2):198–206.

[Daniel and Ping, 2000] Daniel, B. and Ping, C. (2000). Using the fractal dimension to cluster datasets. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 260–264, New York, NY, USA. ACM.

[Deckert, 2011] Deckert, M. (2011). Batch weighted ensemble for mining data streams with concept drift. In *Proceedings of the 19th International Symposium, Foundations of Intelligent Systems*, pages 290–299, Poland.

[Díaz et al., 2014] Díaz, A. O., del Campo-Ávila, J., Ramos-Jiménez, G., Blanco, I. F., Mota, Y. C., Hechavarría, A. M., and Morales-Bueno, R. (2014). Fast adapting ensemble: A new algorithm for mining data streams with concept drift. *The Scientific World Journal*, Article ID 235810, in press.

[Dimitris et al., 1997] Dimitris, G. S., Elias, C. S., and Michael, N. V. (1997). A new hybrid genetic algorithm for global optimization. *Nonlinear Analysis, Theory, Methods and Applications*, 30:4529–4538.

[Duda et al., 2000] Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. John Wiley and Sons, New York.

[Eberhart and Kennedy, 1995] Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *In Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. IEEE.

[Ewan, 1963] Ewan, W. D. (1963). When and how to use cusum charts. *Technometrics*, 5(1):1–22.

[Flaig, 2014a] Flaig, J. J. (2014a). Construction of combined charts based on combining functions. *Applied Mathematical Sciences*, 8(84):4187–4200.

[Flaig, 2014b] Flaig, J. J. (2014b). The shewhart ewma automatic control chart. *Global Journal of Researchers in Engineering*, 14(1):21–26.

[Fox, 2011] Fox, J., W.-S. (2011). *An R Companion to Applied Regression*. Second Edition, Sage.

[Gama and Gaber, 2007] Gama, J. and Gaber, M. M. (2007). *Learning from data streams: processing techniques in sensor networks*. 1st edn. Springer, Berlin.

[Gama et al., 2004] Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *In SBIA Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer Verlag.

[Gama et al., 2013] Gama, J., SebastiÃčo, R., and Rodrigues, P. P. (2013). On evaluating stream

learning algorithms. *Journal of Machine Learning*, 90(3):317–346.

[Gao et al., 2007] Gao, J., Fan, W., Han, J., and Yu, P. S. (2007). A general framework for mining concept drifting data streams with skewed distributions. In *Proceedings of the* 7^{th} *SIAM International Conference on Data Mining (SDM'07)*. AAAI Press, Menlo Park, CA.

[Garnett, 2010] Garnett, R. (2010). *Learning Data Stream with Concept Drift*. PhD thesis, University of Oxford.

[Georgiadis et al., 2010] Georgiadis, M. C., Banga, J. R., and Pistikopoulos, E. N. (2010). Dynamic process modeling: Combining models and experimental data to solve industrial problems. *Process Systems Engineering: Dynamic Process Modeling*, 7:208–219.

[Gibbons, 1999] Gibbons, R. D. (1999). Use of combined shewhart cusum cntrol charts for groud water monitoring applications. *GROUND WATER*, 37(5):682–691.

[Gimeno and Nave, 2009] Gimeno, R. and Nave, J. M. (2009). Genetic algorithm estimation of interest rate term structure. *Computational Statistics and Data Analysis*, 53:2236–2250.

[Hamza and Larocquea, 2005] Hamza, M. and Larocquea, D. (2005). An empirical comparison of ensemble methods based on classification trees. *Journal of Statistical Computation and Simulation*, 75(8):629–643.

[Hauskrecht, 2010] Hauskrecht, T. S. M. (2010). Learning to detect incidents from noisily labeled data. *Journal of machine learning*, 79:335–354.

[Herwartz and Xu, 2009] Herwartz, H. and Xu, F. (2009). A new approach to bootstrap inference in functional coefficient models. *Computational Statistics and Data Analysis*, 53:2155–2167.

[Johnson, 1961] Johnson, N. L. (1961). A simple theoretical approach to cumulative sum control chart. *Journal of the American Statistical Association*, 56(296):835–840.

[Johnson and Leone, 1962] Johnson, N. L. and Leone, F. C. (1962). Cumulative sum control charts: Mathematical principles applied to their construction and use, part i. *Industrial Quality Control*, 18(12):15–21.

[Jr and Chob, 2011] Jr, M. R. R. and Chob, G.-Y. (2011). Multivariate control charts for monitoring the mean vector and covariance matrix with variable sampling intervals. *Sequential Analysis: Design Methods and Applications*, 30(1):1–40.

[Kaibo and Fugee, 2008] Kaibo, W. and Fugee, T. (2008). An adaptive t^2 hart for monitoring dynamic systems. *Journal of Quality Technology*, 40(1):109–123.

[Kolter and Maloof, 2005a] Kolter, J. Z. and Maloof, M. A. (2005a). Dynamic weighted majority: A new ensemble method for tracking concept drift. In *International Conference on Data Mining (ICDM)*, pages 123–130. IEEE.

[Kolter and Maloof, 2005b] Kolter, J. Z. and Maloof, M. A. (2005b). Using additive expert ensembles to cope with concept drift. In *Proceedings of the* 22^{nd} *International Conference on*

Machine Learning (ICML 2005), pages 449–456, Bonn, Germany. ACM Press.

[Kolter and Maloof, 2007] Kolter, J. Z. and Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790.

[Kuncheva, 2009] Kuncheva, L. I. (2009). Using control charts for detecting concept change in streaming data. Technical Report BCS-TR-001-2009.

[Lavretsky and Wise, 2013] Lavretsky, E. and Wise, K. (2013). *Robust and Adaptive Control: With Aerospace Applications*. Springer Verlag London 2013, advanced textbooks in control and signal processing edition.

[Lessmann et al., 2006] Lessmann, S., Stahlbock, R., and Crone, S. F. (2006). Genetic algorithms for support vector machine model selection. In *International Joint Conference on Neural Networks Sheraton Vancouver Wall Centre Hotel*, pages 3063–3069.

[Li et al., 2010] Li, P. P., Wu, X., and Hu, X. (2010). Mining recurring concept drifts with limited labeled streaming data. In *Proceedings of the 2nd Asian Conference on Machine Learning, ACML, 2010, Tokyo, Japan, November 8-10, 2010*, pages 241–252.

[Littlestone and Warmuth, 1994] Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, 108:212–261.

[Liu and Tien, 2010] Liu, C. S. and Tien, F. C. (2010). Design of single featured ewma x control chart for process mean shift detection. In *Proceedings of the* 2^{nd} *International conference on Applied Operational research*, pages 301–314. Lecture Notes in Management Science.

[Liu et al., 2013] Liu, S., Yamada, M., Collier, N., and Sugiyama, M. (2013). Change point detection in time series data by relative density ratio estimation. *Journal of Neural Networks*, 43:72–83.

[Lucas, 1982] Lucas, J. M. (1982). Combined shewhart cusum quality control schemes. *Journal of Quality Technology*, 14(2):51–59.

[Lucas and Crosier, 1982] Lucas, J. M. and Crosier, R. B. (1982). Fast initial response for cusum quality control schemes: Give your cusum a head start. *Technometrics ,*, 24(24):199–205.

[Maloof and Michalski, 2000] Maloof, M. A. and Michalski, R. S. (2000). Selecting examples for partial memory learning. *Machine Learning*, 41:27–52.

[Maloof and Michalski, 2004] Maloof, M. A. and Michalski, R. S. (2004). Incremental learning with partial instance memory. *Artificial Intelligence*, 154:95–126.

[Mejri et al., 2013] Mejri, D., Khanchel, R., and Limam, M. (2013). An ensemble method for concept drift in nonstationary environment. *Journal of Statistical Computation and Simulation*, 83:1115–1128.

[Men et al., 2014] Men, Z., Yee, E., Lien, F. S., Yang, Z., and Liu, Y. (2014). Ensemble nonlin-

ear autoregressive exogenous artificial neural networks for short term wind speed and power forecasting. *International Scholarly Research Notices*, ID 972580.

[Michalski, 1969] Michalski, R. (1969). On the quasi minimal solution of the general covering problem. In *Proceedings of the Fifth International Symposium on Information Processing*, pages 125–128.

[Montgomery, 2005] Montgomery, D. (2005). *Introduction to Statistical Quality Control*. 5^{th} edition, New York: John Wiley and Sons.

[Nembhard and Kao, 2003] Nembhard, H. B. and Kao, M. S. (2003). Adaptive forecast based monitoring for dynamic systems. *Technometrics*, 45(3):208–219.

[Nikovski and Jain, 2009] Nikovski, D. and Jain, A. (2009). Fast adaptive algorithms for abrupt change detection. *Journal of machine learning*, 79:283–306.

[Nishida et al., 2005] Nishida, K., Yamauchi, K., and Omori, T. (2005). ACE: adaptive classifiers ensemble system for concept drifting environments. In *Multiple Classifier Systems, 6th International Workshop, MCS 2005, Seaside, CA, USA, June 13-15, 2005, Proceedings*, pages 176–185.

[Page, 1954] Page, E. S. (1954). Continuous inspection scheme. *Biometrika*, 41(1/2):100–115.

[Pasman, 2015] Pasman, H. (2015). *Risk Analysis and Control for Industrial Processes - Gas, Oil and Chemicals. A System Perspective for Assessing and Avoiding Low-Probability, High-Consequence Events*. Elsevier Science.

[Polikar, 2006] Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45.

[Rangel-Merino et al., 2005] Rangel-Merino, A., López-Bonilla, J. L., and y Miranda., R. L. (2005). Optimization method based on genetic algorithms. *Apeiron*, 12:393–408.

[Raza et al., 2013] Raza, H., Prasad, G., and Li, Y. (2013). Dataset shift detection in non-stationary environments using ewma charts. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 3151–3156.

[Raza et al., 2015] Raza, H., Prasad, G., and Li, Y. (2015). Ewma model based shift detection methods for detecting covariate shifts in non stationary environments. *Pattern Recognition*, 48(3):659–669.

[Reis and Mandl, 2003] Reis, B. Y. and Mandl, K. D. (2003). Time series modeling for syndromic surveillance. *BMC medical informatics and decision making*, 3:1–11.

[Reynolds, 2008] Reynolds, JR., M. R. S. Z. G. (2008). Combinations of multivariate shewhart and mewma control charts for monitoring the mean vector and covariance matrix. *Qulaity Technology*, 40(4):381–393.

[Rijsbergen, 1979] Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann

1979.

[Robert, 1959] Robert, S. W. (1959). Control chart test based on geometric moving average. *Technometrics*, 42(1):97–102.

[Sampaio et al., 2014] Sampaio, E. S., Ho, L. L., and de Medeiros, P. G. (2014). A combined np_x $\overline{X}$ control chart to monitor the process mean in two stage sampling. *Quality and Reliability Engineering International*, 30(7):1003–1013.

[Schlimmer and Granger, 2011] Schlimmer, J. and Granger, R. (2011). Fuzzy classification in dynamic environments. *Soft Computation*, 15(5):1009–1022.

[Schlimmer and Granger, 1986] Schlimmer, J. C. and Granger, R. H. (1986). Beyond incremental processing: Tracking concept drift. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 502–507. AAAI Press, Menlo Park, CA.

[Shewhart, 1931] Shewhart, W. A. (1931). *Economic Control of Quality of Manufactured Product*. New York: D. Van Nostrand Company.

[Steiner, 1999] Steiner, S. H. (1999). Ewma control charts with time-varying control limits and fast initial response. *Journal of Quality Technology ,*, 31(1):75–86.

[Street and Kim, 2001] Street, W. N. and Kim, Y. (2001). A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382, New York, NY, USA. ACM.

[Takeuchi and Yamanishi, 2006] Takeuchi, J.-i. and Yamanishi, K. (2006). A unifying framework for detecting outliers and change points from time series. *IEEE Trans. on Knowl. and Data Eng.*, 18(4):482–492.

[Tan et al., 2008] Tan, F., Fu, X., Zhang, Y., and Bourgeois, A. G. (2008). A genetic algorithm based method for feature subset selection. *Soft Computation*, 12:111–120.

[Tsymbal, 2004] Tsymbal, A. (2004). The problem of concept drift: definitions and related work. Technical report.

[Wang et al., 2003] Wang, H., Fan, W., Yu, P. S., and Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 226–235, New York, NY, USA. ACM.

[Widmer and Kubat., 1996] Widmer, G. and Kubat., M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23:69–101.

[Wilson, 1995] Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175.

[Wu and Yu, 2010] Wu, B. and Yu, J. B. (2010). A neural network ensemble model for online

monitoring of process mean and variance shifts in correlated processes. *Expert Systems with Applications*, 37(6):4058âĂŞ4065.

[Yandell, 1997] Yandell, B. S. (1997). *Practical Data Analysis for Designed Experiments*. Chapman and Hall/CRC Texts in Statistical Science.

[Yang and Liao, 2015] Yang, W., Y. G. and Liao, W. (2015). A hybrid learning based model for simultaneous monitoring of process mean and variance. *Quality Reliability Engineering*, 31:445–463.

[Zeng et al., 2007] Zeng, D., Gotham, I., Komatsu, K., Lynch, C., Thurmond, M., Madigan, D., Lober, B., Kvach, J., and Chen, H. (2007). *Intelligence and Security Informatics : Biosurveillance*. Second NSF Workshop, New Brunswick, NJ, USA, Proceedings, vol. 4506 of Lecture Notes in Computer Science. Springer.

[Zhang et al., 1991] Zhang, X., Elishakoff, I., and Zhang, R. (1991). *A Stochastic Linearization Technique Based on Minimum Mean Square Deviation of Potential Energies*. Springer Verlag.

[Zhou, 2012] Zhou, Z. H. (2012). *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition.

[Zhu, 2010] Zhu, X. (2010). Stream data mining repository.

[Zhu et al., 2013] Zhu, X., Zhang, P., Lin, X., and Shi, Y. (2013). Active learning from stream data using optimal weight classifier ensemble. *IEEE Transactions on systems, Man, and Cybernetics, Part B: Cybernetics,*, 40(6):1607–1621.