Handling Concept Drift in the Context of Expensive Labels

Patrick Lindstrom
Technological University Dublin, patrick.lindstrom@tudublin.ie

Handling Concept Drift in the Context of Expensive Labels

by

Patrick Lindström

Supervisors: Dr. Sarah Jane Delany
Dr. Brian Mac Namee
Prof. Pádraig Cunningham

School of Computing

Dublin Institute of Technology

A thesis submitted for the degree of

Doctor of Philosophy

September, 2013

This thesis is lovingly dedicated to my parents, Tommy and

Paula Lindström who have always been there for me.

Abstract

Machine learning has been successfully applied to a wide range of prediction problems, yet its application to data streams can be complicated by concept drift. Existing approaches to handling concept drift are overwhelmingly reliant on the assumption that it is possible to obtain the true label of an instance shortly after classification at a negligible cost. The aim of this thesis is to examine, and attempt to address, some of the problems related to handling concept drift when the cost of obtaining labels is high.

This thesis presents Decision Value Sampling (DVS), a novel concept drift handling approach which periodically chooses a small number of the most useful instances to label. The newly labelled instances are then used to re-train the classifier, an SVM with a linear kernel, to handle any change in concept that might occur. In this way, only the instances that are required to keep the classifier up-to-date are labelled. The evaluation of the system indicates that a classifier can be kept up-to-date with changes in concept while only requiring 15% of the data stream to be labelled. In a data stream with a high throughput this represents a significant reduction in the number of labels required.

The second novel concept drift handling approach proposed in this thesis is Confidence Distribution Batch Detection (CDBD). CDBD uses a heuristic based on the distribution of an SVM's confidence in its predictions to decide when to rebuild the classifier. The evaluation shows that CDBD can be used to reliably detect when a change in concept has taken place and that concept drift can be handled if the classifier is rebuilt when CDBD signals a change in concept. The evaluation also shows that CDBD obtains a considerable labels saving as it only requires labelled data when a change in concept has been detected.

The two concept drift handling approaches deal with concept drift in a different manner, DVS continuously adapts the classifier, whereas CDBD only adapts the classifier when a sizeable change in concept is suspected. They reflect a divide also found in the literature, between continuous rebuild approaches (like DVS) and triggered rebuild approaches (like CDBD). The final major contribution in this thesis is a comparison between continuous and triggered rebuild approaches, as this is an underexplored area. An empirical comparison between representative techniques from both types of approaches shows that triggered rebuild works slightly better on large datasets where the changes in concepts occur infrequently, but in general a continuous rebuild approach works the best.

Declaration

Acknowledgements

Doing a PhD has been a long, but deeply rewarding experience. I have no doubt that it would have been impossible if not for the tremendous support shown by the people around me. I would like to take this opportunity to thank them for their much-needed help along the way.

I would firstly like to thank my supervisory team, Dr. Sarah Jane Delany and Dr. Brian Mac Namee for their advice, guidance and encouragement from the first day, to the day I handed in my thesis. Any PhD student would be very lucky to have one supervisor of such a high calibre, I was truly blessed to have two. There was never a question to small, an email too late or problem too large for a thoughtful reply. I would also like to express my gratitude to Prof. Pádraig Cunningham for the advice and insight he provided, particularly around the crucial time of my transfer exam.

I will look back at the time spent in the Dublin Institute of Technology with fond memories, due in large part to the friendships I have made there. I would like thank everyone in the Applied Intelligence Research Centre, past and present. Particularly my friends, and fellow travellers along the weird and wonderful path to a PhD, Dr. Rong (Amy) Hu, Dr. Kenneth Kennedy, Colm Sloan, Niels Schütte and Yan Li. Their knowledge, enthusiasm and good humour brightened up even the dreariest of days.

I would also like to acknowledge the wonderful staff in the School of Computing in the Dublin Institute of Technology. They have played a vital role throughout my academic life, both as outstanding lecturers while I was doing my undergraduate degree, and as fountains of knowledge and sage advice while I was doing my PhD.

I would also like to express my appreciation to Niall Cunningham, Ian Primrose and Laura Ferry who generously offered to proofread early drafts of the thesis. Their valuable feedback and many interesting insights improved the thesis immensely.

I would of course also like to thank my family and friends who have believed in me and encouraged me to follow my heart. My siblings Angela and Michael, parents Tommy and Paula, and extended family have shown great patience and kindness during the last few years, which allowed me to pursue a PhD. Finally, I'd like to thank Jessie for her unwavering encouragement, understanding and love throughout the process, I could not have done it without her.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Artificial Intelligence (AI) (McCarthy *et al.*, 1955) has moved from the realm of academics and science fiction writers to everyday life in a relatively short time. Obvious examples of AI are now part of our daily lives, for example, search engines, video games, recommender systems and manufacturing robots. However, AI is also present in more unexpected places such as voice and handwriting recognition, face detection in digital cameras and spam filtering. One of the major factors contributing to the growth of AI is the rapid expansion of our collective digital fingerprint. Data is being collected, stored and analysed at an unprecedented rate, in large part due to the pervasiveness of the Internet. This has made Machine Learning (ML) (Mitchell, 1997), a subfield of AI, particularly important.

Machine learning can be used to leverage past data to make predictions about the future. For example banks use data about previous customers' ability to pay back loans to predict whether or not a new customer will be able to pay back a proposed loan. This can be achieved by *training* a *classifier* on customer data. Classifiers attempt to determine the mapping between the characteristics of the data and the *actual outcome* during the training process. The actual outcome is the "correct" prediction and is essential in the training process. Once trained, the classifier is able to predict the outcome of data where the outcome is unknown.

Machine learning excels at finding patterns in large and complex data, making it very well suited to a variety of tasks which require predictions about new data based on past data. Examples to which machine learning has been successfully applied include: sentiment analysis (also known as opinion mining) (Pang & Lee, 2004), face recognition (Guo *et al.*, 2000), spam filtering (Delany *et al.*, 2005), handwriting recognition (Xu *et al.*, 1992), breast cancer diagnosis (Manning & Walsh, 2013), fraud detection (Fawcett & Provost, 1996) and recommender systems (Mooney & Roy, 2000).

A key assumption about the data is that it does not change significantly over time, i.e. that the data used to train a classifier is representative of the data that the classifier will encounter in the future. In machine learning terms this is known as a *stable concept.* In many real-world prediction problems the concept is not static but rather changes over time. The degradation of classifier performance due to the non-stationary nature of the concept is known as *concept drift.* For example, take spam filtering, a machine learning problem where a classifier is trained on a collection of historical emails, and then attempts to predict if incoming emails are relevant (ham) or non-relevant (spam) to a particular user. Initially the classifier might be able to correctly predict which emails are spam, but over time the classifier becomes less accurate due to concept drift. Concept drift in spam filtering can be attributed to various factors, including changes to the content of the emails, changes in what the user considers spam and an active effort by spam creators to obfuscate the spam. Other examples of changing concepts can be seen in a variety of real-world applications: weather predictions are affected by seasonal weather variations, customer buying preferences can be influenced by fashion trends or seasonal inclinations, and financial predictions can be shaped by macroeconomics.

The classifier needs to be re-trained with new, up-to-date data when a change in concept has occurred in order to maintain classifier performance. One way to differentiate how the approaches deal with handling concept drift is based on when they decide to re-train the classifier (Kuncheva, 2009). The first approach regularly updates the classifier, assuming that this will allow the classifier to handle concept drift whenever it occurs (such as (Baena-García *et al.*, 2006; Gama *et al.*, 2004; Klinkenberg & Joachims, 2000; Klinkenberg & Renz, 1998; Kubat, 1989; Nishida & Yamauchi, 2007; Widmer & Kubat, 1996; Zhu *et al.*, 2007)). This will be referred to as a *continuous rebuild* approach, it does not explicitly attempt to detect a change in concept. The second approach explicitly attempts to detect when a change in concept has occurred, and only then adapts the classifier (such as (Fan *et al.*, 2004a; Kifer *et al.*, 2004; Lanquillon, 1999; Sebastião & Gama, 2007; Zliobaite, 2010)). This type of approach will be referred to as a *triggered* *rebuild* approach.

A shared trait in both types of approach is the requirement for the actual outcomes to be known after the prediction is made. A continuous rebuild approach needs new data with the actual outcomes to re-train the classifier. Triggered rebuild approaches usually base the decision to rebuild on metrics which require the actual outcome in their calculation.

In many domains the requirement that the actual outcome is known shortly after classification is not a restriction. For example in short term stock market predictions where the correct outcome is known the day after the prediction is made. However, it can be a significant constraint in domains such as *information filtering*.

In information filtering the classification task is to present a user with documents which the system believes are relevant to a user, while filtering out non-relevant documents. A practical example of this might be a news filtering application which receives a continuous stream of news articles which it attempts to categorise as relevant or not-relevant to a particular user. The stream might experience concept drift either because the content of the documents have changed significantly, or the users' opinion of what is relevant has changed.

In both cases new documents with their associated actual outcomes are needed to keep the classifier up to date. In a text classification problem like information filtering this means that someone needs to read each document and assign each one a true outcome, a process known as *labelling*. There is expense and effort involved in creating this new labelled data, due to the effort involved in reading and categorising texts. This problem is magnified in a domain where a large amount of data needs to be processed. The high labelling cost provides a strong motivation to develop concept drift handling approaches that only require a subset of the data in the stream to be labelled.

This area of research has received a relatively small amount of attention until recently. Most concept drift handling approaches assume that the actual outcome is readily available. However, there has been an increased emphasis on this area lately, and it has been proposed that reliance on knowing the actual outcome is one of the problems preventing machine learning techniques being deployed more extensively in industry (Zliobaite *et al.*, 2012). This is the problem this thesis aims to solve.

1.1 Scope and Contributions of this Thesis

This thesis aims to explore the area of concept drift handling in a scenario where it is infeasible for the full data steam to be labelled. The work in this thesis will be grounded in the text classification sub-field of information filtering as this field has a high volume of data, experiences concept drift and has a high labelling cost.

Addressing changes in concept can be broken down into two subtasks: *concept drift detection* and *concept drift adaptation*. Drift detection deals with detecting when a significant change in concept has taken place. Concept drift adaptation is concerned with how the classifier is updated to take account of a change in concept. Reducing the need for labelled data can be achieved by improvements in concept drift detection, concept drift adaptation, or both.

Improvements to concept drift adaptation tend to be achieved through adjustments to the way that a continuous rebuild approach re-builds the classifier. This thesis presents Decision Value Sampling (DVS), a continuous rebuild approach which periodically chooses a small number of the most useful instances to label. The newly labelled instances are then used to retrain the classifier, an SVM with a linear kernel, to handle any change in concept that might occur (Lindstrom *et al.*, 2010a). In this way only the instances that are required to keep the classifier up to date are labelled, which greatly reduces the labelling effort required. Evaluation of the system indicates that a classifier can be kept up-to-date with changes in concept at a labelling cost of only 15% of the data stream being labelled. In domains where large numbers of documents are classified this represents a significant reduction in labelling costs.

This type of approach uses a fixed amount of labelled data (for example 15%) regardless of whether the concept is changing or not. A more label efficient approach might be to use a concept drift detection approach which estimates if a change in concept has taken place or not, and only requests labelled data when a change in concept is suspected. Triggered rebuild approaches which can estimate if a change in concept has taken place without needing labelled data are of particular interest. This thesis proposes Confidence Distribution Batch Detection (CDBD) (Lindstrom *et al.*, 2013, 2011), which uses a heuristic based on the distribution of an SVM's confidence in its predictions to decide when to rebuild the classifier. The CDBD heuristic does not need labelled data, and evaluations show that CDBD can be used to reliably detect when a change in concept has taken place. The evaluation also shows that concept drift can be handled if the classifier is rebuilt when CDBD signals a change in concept.

The comparison between the continuous and triggered rebuild approaches is an underexplored area in the literature. An empirical comparison between representative techniques from both types of approaches was carried out and the results showed that triggered rebuild works slightly better in large datasets where the changes in concepts occur infrequently, but in general a continuous rebuild approach works best.

The main contributions of this thesis can be summarised as follows:

- A review of the literature, including literature dealing with handling concept drift with a limited amount of labelled data (Chapter 3).
- A novel continuous rebuild approach for handling concept drift that handles concept drift using just 15% of the data available (Chapter 5).
- A detection and rebuild approach for handling concept drift which only needs labelled data when rebuilding and has been shown to accurately detect changes in concept in text data streams (Chapter 6).
- An empirical evaluation of representative continuous and triggered rebuild concept drift handling approaches that shows that a continuous rebuild approach is better than a triggered rebuild approach in most circumstances (Chapter 7).

1.2 Summary and Structure of this Thesis

The remainder of this thesis is structured as follows. Chapter 2 provides a high level overview of machine learning and how it can be applied to text classification, as the approaches proposed in this thesis are evaluated on text data. Chapter 3 surveys state of the art research in concept drift and how concept drift can be handled and detected. A particular emphasis is placed on approaches which reduce the need for labelled data. Chapter 4 outlines the methodology used to evaluate the concept drift handling approaches proposed in this thesis. Chapter 5 presents the design, implementation and evaluation of DVS while Chapter 6 details the design, implementation and evaluation of CDBD. Chapter 7 compares continuous and trigged rebuild approaches to establish under what conditions one approach might be more suited than the other. Chapter 8 summarises the key contributions of this work and highlights opportunities for additional research.

1.3 Publications

The publications that form the basis for this thesis are listed below:

Lindstrom, P., Delany, S.J. & Mac Namee, B. (2010). Handling concept drift in a text data stream constrained by high labelling cost. In H.W. Guesgen & R.C. Murray, eds., *Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference*, AAAI Press.

Lindstrom, P., Mac Namee, B. & Delany, S.J. (2011). Drift detection using uncertainty distribution divergence. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, 604–608, IEEE Computer Society.

Lindstrom, P., Mac Namee, B. & Delany, S. (2013). Drift detection using uncertainty distribution divergence. *Evolving Systems*, **4**, 13–25.[1]

Lindstrom, P., Delany, S.J. & Mac Namee, B. (2008). Autopilot: simulating changing concepts in real data. In *Proceedings of the 19th. Irish Conference on Artificial Intelligence and Cognitive Science*, 21.

Lindstrom, P., Hu, R., Delany, S.J. & Mac Namee, B. (2010a). SVM based active learning with exploration. In *AISTATS 2010 Workshop on Active Learning and Experimental Design*.

Hu, R., Lindstrom, P., Delany, S.J. & Mac Namee, B. (2010). Exploring the frontier of uncertainty space. In *AISTATS 2010 Workshop on Active Learning and Experimental Design*.

[1]This is an extended version of the paper presented at ICDMW.

Chapter 2

Machine Learning and Text Classification

The pace and scope of data gathering is constantly growing as the value of data is being fully appreciated. Shops, banks, social networks, online retailers and governments all save data which they hope to analyse for patterns to aid better decision making. Machine learning techniques are popular ways of analysing data, and can be applied to a myriad of problems. Particular focus will be placed on a sub-field of machine learning known as text classification, which deals with the automatic categorisation of text data. Text classification problems tend to have large volumes of data, experience concept drift and have a high labelling cost, which is why this thesis uses text classification as the real-world domain on which to base the work.

This chapter introduces machine learning and shows how it can be applied to text data, to support the material addressing concept drift. The remainder of this chapter is organised as follows, Section 2.1 gives a high level outline of how machine learning can be applied to data. Section 2.2 presents an overview of how raw data is prepared for machine learning, followed by Section 2.3 where classification is explained and some common machine learning algorithms are described in more detail. Section 2.4 shows how classifiers can be evaluated. Section 2.5 demonstrates how machine learning can be applied to the abundance of data stored in text format, such as blogs, news articles and tweets. This is followed by Section 2.6, a brief conclusion.

2.1 The Knowledge Extraction Process

Using machine learning algorithms is only one step in the larger knowledge extraction process. The knowledge extraction process can be described using a formal knowledge extraction framework such as the CRoss-Industry Standard Process for Data Mining (CRISP-DM) (Shearer, 2000) or Knowledge Discovery in Databases (KDD) (Fayyad *et al.*, 1996).

It has been argued that one of the major weaknesses of CRISP-DM is that it is not designed for high-frequency, high-volume, real-time, multi-dimensional time series data (Catley *et al.*, 2009). However, this limitation will not be dwelled on as the CRISP-DM process is not fundamental to the work in this thesis, but is rather only used to structure the discussion in this chapter. The CRISP-DM process involves the following phases:

1. **Business understanding** Understanding the project objectives from a business perspective and translating these objectives to a machine learning problem.

2. **Data understanding** Collecting the initial data and ensuring the quality of the data e.g. handling missing and implausible values.

3. **Data preparation** Transforming the data into a format suitable for use with machine learning techniques.

4. **Modelling** Selecting the appropriate machine learning technique, and training the model from all, or a subset of the collected data.

5. **Evaluation** Evaluating the model based on an evaluation metric and ensuring that it fulfils the business goals.

6. **Deployment** Deploying the model inside the business decision making process.

The work in this thesis focuses primarily on phases three to five, but will reference the other phases when appropriate. The subsequent three sections will discuss phases three to five in more detail.

2.2 Data Preparation

The knowledge extraction process can be illustrated using a simplified example of a bank wishing to use customer data to guide their decision making processes. The bank has a large customer database at its disposal which it wants to use for the following machine learning objectives:

- Predicting if a new customer will be able to pay back a loan if one is granted.
- Predicting how long it will take a current customer to repay their loan.
- Identifying new groupings of customers.

The next step is the data understanding phase, in which one goal is to collect the initial data. It is assumed that the bank in this example already has a large customer database, which leads to the next step, ensuring the quality of the data. This generally involves the identification and correction of incorrect, missing and redundant data. After this the bank must select what data to use, as it does not collect all possible customer attributes. Some attributes might be left out due to legal restrictions and other attributes might be ignored as they are not relevant to the machine learning goal(s). The attributes which are suspected to affect the machine learning task are known as *features*. This might include features such as current salary, number of dependants and home ownership status.

In machine learning each object is known as an *instance* and each instance is comprised of a set of features. In the bank example each customer is an instance and each instance has features, such as current salary, which describe the instance. The final data preparation step is to translate the bank data into a collection of instances, known as a *dataset*. An instance can be denoted mathematically as $x \in R^d$ where x is a d dimensional vector, containing the d features of x. In a dataset X instance i can be referenced using x_i and feature j of instance i can be referenced as x_{ij}.

2.3 Modelling

The first task in the modelling step is the selection of an appropriate modelling technique for the machine learning goal(s). Machine learning techniques can be subdivided into three fields, *supervised learning*, *semi-supervised* learning and *unsupervised learning*. In supervised and semi-supervised learning, models are constructed which try to make predictions about unseen data based on historical data. In unsupervised learning, models are constructed which try to find hidden structure in the data.

This thesis focuses mostly on supervised learning, but will reference semi-supervised and unsupervised learning when appropriate. In supervised machine learning a set of instances, known as *training data*, is used to train a classifier using a classifier dependent training algorithm. The training data needs to be labelled, i.e. the instances need to be coupled with the variable the machine learning algorithm is trying to predict. The variable may be categorical or numerical. If the variable is categorical the prediction task is known as *classification*, and the outcome the classifier is trying to predict is known as the *class* or *label*. In the bank example a classification task might be to predict if a potential new customer belongs to a high-risk or low-risk class. On the other hand, if the outcome is numerical the prediction task is known as *regression*. In the bank example a regression task might be to predict how long it will take a given customer to pay back a loan. This thesis is primarily concerned with classification problems.

If an instance has a corresponding class it is known as a *labelled instance*, conversely an instance without a label is known as an *unlabelled instance*. A labelled instance is therefore denoted as (x, y), where y is the label of x, $y \in \{1, ...K\}$ and $K \in N$ for K classes. Classification problems with only two classes are known as *binary classification* problems and the instances belonging to the class of interest are called the positive instances while the other instances are known as the negative instances.

During the training process the classifier uses the training data to learn a function which maps between the features and the label. Once the classifier is trained it can be used to predict the label of unlabelled instances. The next section covers some supervised classification algorithms which will be referred to in other parts of this thesis.

2.3.1 Decision Trees

A Decision Tree (DT) is essentially a sequence of nested tests of feature values which results in a prediction (Quinlan, 1986). The process is analogous to trouble shooting where an expert asks a series of questions to determine where the problem lies. The structure can be illustrated using the example in Figure 2.1. In this example cars are classified based on their suitability as a family car. An instance is either considered “good” or “bad” based on four features, its safety record, the number of persons it fits, the size of its luggage boot and the number of doors it has. A new instance is classified by following the path though the *decision nodes* (the rectangles in Figure 2.1) until a *leaf node* (the circles in Figure 2.1) is reached. The decision tree predicts that the instance belongs to the class in the leaf node reached.

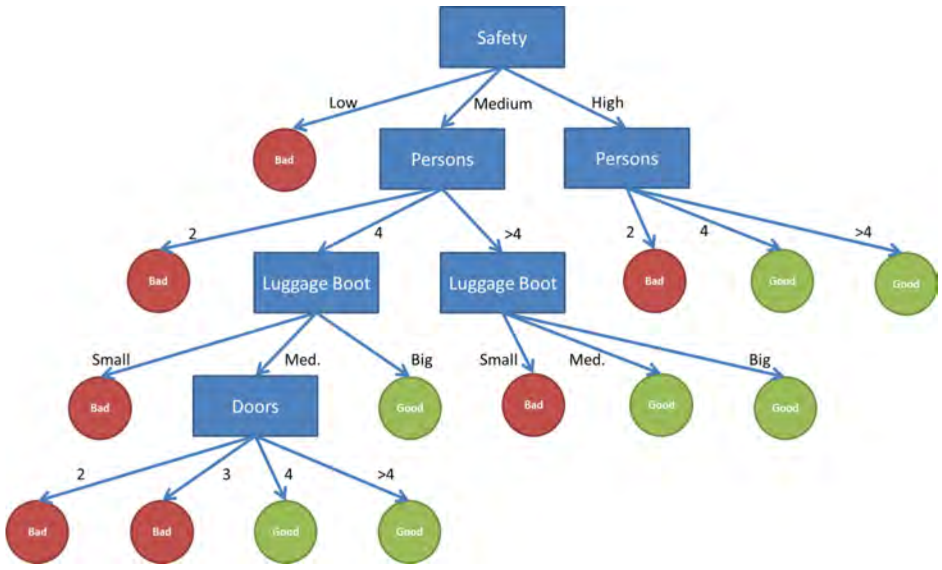


Figure 2.1: A simple example of a decision tree.

Decision trees are relatively slow to train (particularly if the data has a large number of features), but fast at classifying unlabelled instances. Another attractive property of decision trees is their interpretability. The ability to explain to a non-expert how a classifier arrived at a particular prediction can be vital in some industries, such as the financial and medical domain.

2.3.2 Similarity Based Classifiers

Similarity based classifiers classify unlabelled instances by measuring how similar they are to class prototypes. The process can be illustrated using one of the simplest similarity based classifiers, the *single prototype classifier* (Lanquillon, 1999) (also known as "Find Similar" in (Dumais *et al.*, 1998)). In a single prototype classifier each class is represented by a prototype instance, which is an average of all the training instances of that class (also known as a centroid).

More formally, let X be the set of training instances with d features

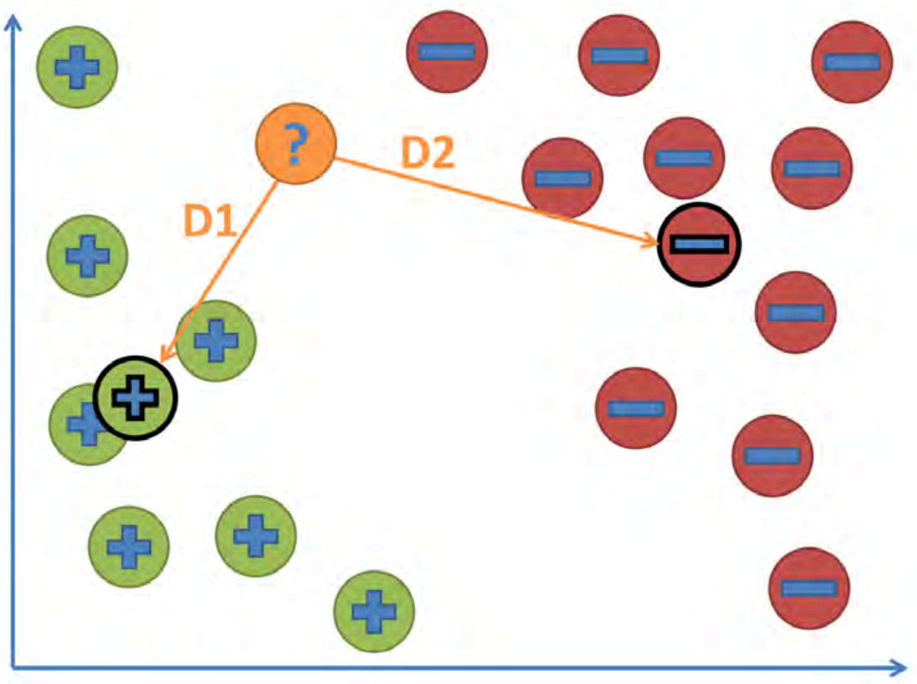


Figure 2.2: A simple example of a single prototype classifier.

and let X^y be the subset of X of class y and N be the size of X^y. Using the notation from Section 2.2 the prototype of class y can be written as: $p = \{p_1, p_2 ... p_d\}$ where $p_j = (1/N) \sum_{i=1}^{N} x_{ij}^{y}$. The prototype has the exact same structure as a training instance, but is seldom an actual instance in the training data, but rather a virtual instance. One prototype is created for each class and unlabelled instances are predicted to belong to the same class as the prototype they are the most similar to, based on a Euclidean distance measure.

The process is illustrated in the toy example in Figure 2.1 where positive and negative training examples are plotted in two dimensions. The positive instances are depicted as circles with a plus sign, negative training instances as circles with a minus sign and the unlabelled instance as a circle with a question mark. The class prototypes are marked using a black border. In this example $D1$ is smaller than $D2$ so the unlabelled instance is predicted to belong to the positive class.

The single prototype classifier is conceptually simple and has a very short training time, however it does not always achieve a classification accuracy comparable with more sophisticated classification techniques (Dumais *et al.*, 1998).

A more common similarity based classifier is the k Nearest Neighbour classifier, or k-NN (Cover & Hart, 1967). A k-NN classifier treats each instance in the training data as a prototype. Unlabelled instances are predicted to belong to the same class as their k closest training instances as measured by a distance function[1].

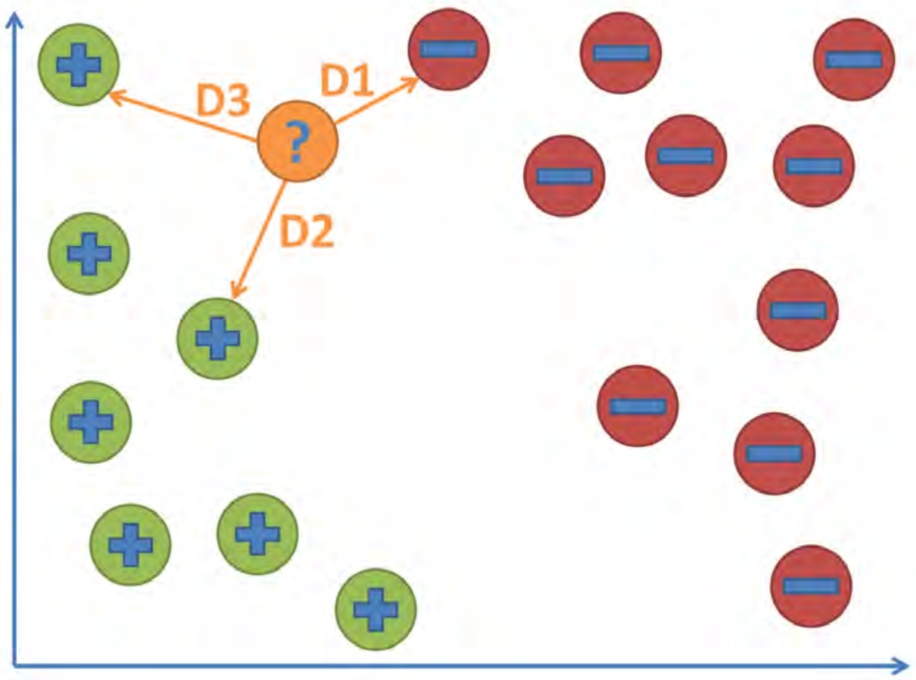


Figure 2.3: A simple example of a k-NN classifier.

Figure 2.3.2 shows how k-NN works using the same toy example as pre-

[1]For a comprehensive review of distance functions suitable for similarity based classifiers see (Cunningham, 2009)

viously. A 1-NN classifier would predict that the unlabelled instance belongs to the negative class as $D1$ is the shortest distance. However, a 3-NN classifier would find the three shortest distances ($D1$, $D2$ and $D3$) and predict that the unlabelled instance belongs to the positive class, as two out of three nearest neighbours are positive instances.

Similarity based classifiers are typically very fast to train and easy to update. However they can be slow to classify unlabelled instances, particular if the data is high dimensional. This limitation can be somewhat mitigated by using feature selection, which will be covered in Section 2.5.2.

2.3.3 Support Vector Machines

A *Support Vector Machine* (SVM) attempts to find the hyperplane which separates the instances from two classes while maintaining the largest possible margin between the hyperplane and the instances closest to the hyperplane, which are known as *support vectors* (Vapnik, 1999). Figure 2.4 shows an example plotted in two dimensions. The positive and negative instances are linearly separable but there are various different lines (or 1-dimensional hyperplanes) that can be used to separate the instances. The decision boundary should be as far away from the instances of both classes as possible. For this reason SVMs are also known as maximum margin classifiers. In Figure 2.4 $L1, L2, L3$ would separate the positive and negative instances but $L2$ would provide the largest margin, so $L2$ is chosen as the decision boundary, $\mathbf{w}$.

SVMs work well on linearly separable data, however some data is not linearly separable. To overcome this limitation a kernel function is used. The

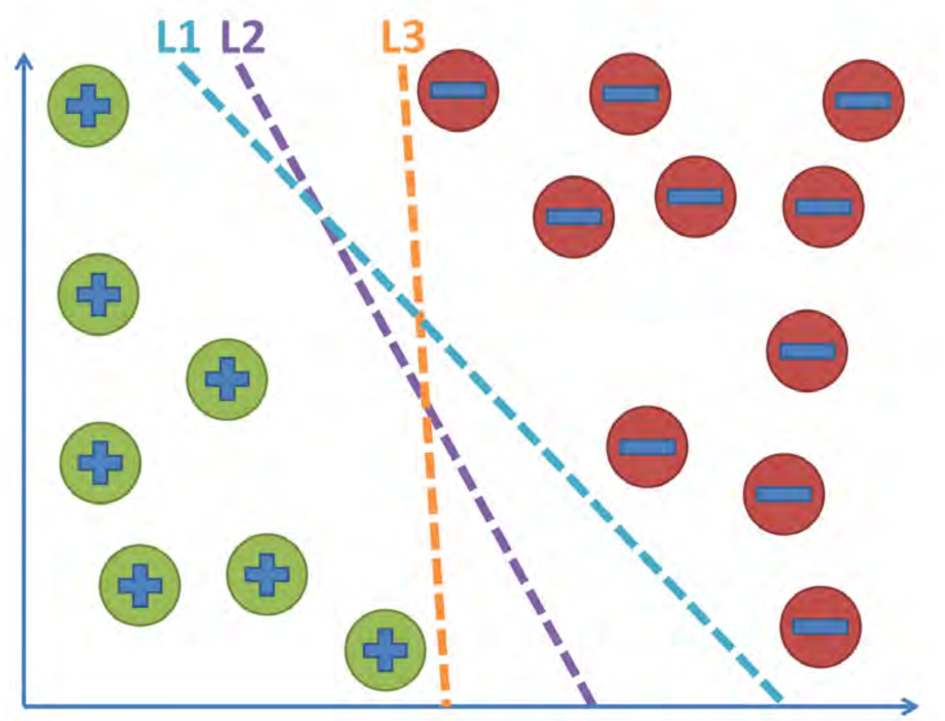


Figure 2.4: A simple example of an SVM classifier.

kernel of an SVM is a function which maps the data into a higher dimensional space, where data can be linearly separated. A few commonly used kernels are Linear, Polynomial and Radial Basis Function (see (Hsu *et al.*, 2003) for more information about choosing an SVM kernel).

Once an SVM is trained it can produce a *decision value* for an unlabelled instance x_i using $dv_i = \mathbf{w} \cdot \mathbf{x_i} + b$ and the predicted class is $y = sign(dv_i)$.

Several studies have shown that SVMs are particularly well suited to text data (Dumais *et al.*, 1998; Joachims, 1998; Yang & Liu, 1999). Joachims (1998) suggests this because of how well SVMs handle data which is high dimensional, sparse and contains few irrelevant features. Another reason SVMs handle text classification so well is because text classification problems tend to be linearly separable, due to their high dimensionality. This also makes the linear kernel a good choice for text classification problems (Dumais *et al.*, 1998; Yang & Liu, 1999). However, one of the drawbacks of SVMs is that, unlike similarity based classifiers the training time of an SVM can increase dramatically as the size of the training set increases.

2.3.4 Ensemble Classifiers

An *ensemble* is a collection of classifiers, where the prediction of each classifier, known as a *base classifier*, is combined to classify unlabelled instances (Rokach, 2010). This process is illustrated in Figure 2.5.

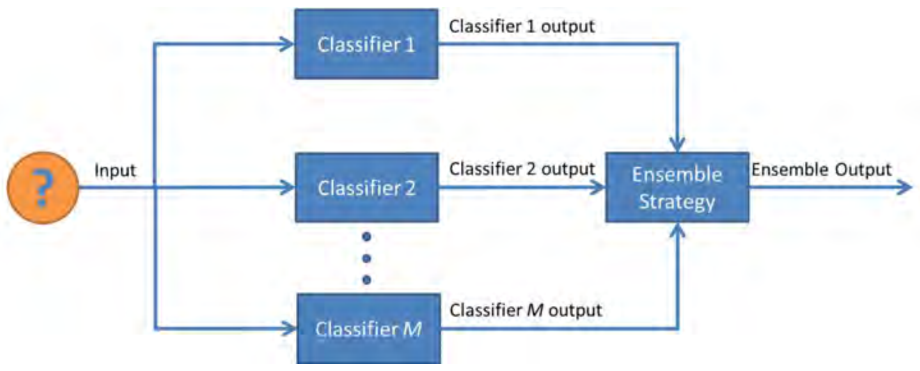


Figure 2.5: A simple example of an ensemble classifier.

For an ensemble of base classifiers to be more accurate than any of its individual members the classifiers must be accurate and diverse (Dietterich, 2000). A base classifier is considered accurate if its predictions are more accurate than random guessing and two classifiers are considered diverse if they make different errors on new instances. Opitz & Maclin (1999) state that the main emphasis of creating ensembles is creating diverse base classifiers, which centre around producing classifiers that disagree on their prediction. An ensemble can be categorised based on how diversity is introduced and how the final prediction of the ensemble is produced.

Ensemble diversity is usually created by using different training data in each base classifier. The two most common methods for selecting training data are bagging and boosting. Bagging (**B**ootstrap **agg**regat**ing**) (Breiman, 1996) uses random sampling with replacement to select a subset of the original training set, which may contain duplicate instances. Bagging is particularly suitable for base classifiers where a small change in the training data can cause a large change in the internal structure of the classifier (like decision trees) but does not show the same performance on stable classifiers (like SVMs) (Dietterich, 2000). The other common training data selection approach Boosting (Freund & Schapire, 1996) increases the probability that an instance will be sampled as training data if the previous classifiers in the ensemble misclassified that instance. Boosting can produce better results than bagging, but can also produce a worse performance than a single classifier if the data is noisy (Dietterich, 2000; Opitz & Maclin, 1999).

The second most important aspect of ensembles, after how diversity is introduced, is the ensemble strategy used to construct the final prediction. There are essentially two approaches to forming the final prediction: *fusion methods* and *selection methods* (Rokach, 2010). Fusion methods combine the output from the base classifiers using a criteria such as majority voting (as in (Breiman, 1996)) or performance weighted voting (as in (Opitz & Shavlik, 1995)) [1]. Selection methods produce the final prediction by selecting which classifier(s) predictions should be used. This is achieved by giving one base classifier authority over a particular area of the feature space and if a test

[1]For a more detailed review of ensemble fusion methods see (Kuncheva, 2002; Ruta & Gabrys, 2000)

instance lies within that area that particular classifier provides the prediction (as in (Ho *et al.*, 1994; Woods *et al.*, 1997)).

Ensembles are generally more accurate than any one of their base classifiers. However, this gain in accuracy usually incurs a computational cost, as it is computationally more expensive to create an ensemble than a single base classifier.

2.4 Evaluation

An important aspect of machine learning is evaluating the model(s) trained. Models should be evaluated in terms of an evaluation metric suitable for the problem being modelled. This thesis focuses on classification tasks where the goal is to predict which class an unlabelled instance belongs to (as opposed to regression). The most common evaluation metric to evaluate classifiers is *classification accuracy*. Classification accuracy is the fraction of instances for which the classifier predicted the correct label. The classification accuracy is normally calculated on a set of instances that the classifier was not trained on, referred to as a *test set*. More formally, let C be the set of instances from the test set where the predicted class matched the true class and let I be the set of instances from the test set where the predicted class did not match the predicted class. Classification accuracy is defined as:

$$accuracy = \frac{|C|}{(|C| + |I|)} \tag{2.1}$$

Another commonly used evaluation metric is the *misclassification rate*, which is also known as the *error rate*. The misclassification rate is simply the fraction of instances for which the classifier predicted the incorrect label, i.e.

$$misclassificationRate = \frac{|I|}{(|C| + |I|)} \tag{2.2}$$

The misclassification rate can also be calculated using the classification accuracy as $misclassificationRate = 1 - accuracy$.

Classification accuracy might seem like a reliable evaluation metric, however, inappropriate use of this metric can lead to a situation where an increase in the classification accuracy actually results in a decrease in the predictive power of the classifier, which is known as the *accuracy paradox* (Bruckhaus, 2007). A common example of the accuracy paradox is found in data containing a high *class imbalance*, i.e. instances of one class are significantly less prevalent than the other classes. When the data has a high class imbalance the accuracy alone might hide the fact that the classifier is unable to predict the minority class, as only predicting the majority class can yield a high classification accuracy. In this type of scenario the *average class accuracy* can give a better measure of classifier performance:

$$avgClassAccuracy = \frac{\sum_{j=1}^{|Y|} accuracy_j}{|Y|} \tag{2.3}$$

where Y is the set of possible classes, $accuracy_j$ is the classifier accuracy on instances of class y_j.

2.5 Text Classification

Since the availability of unstructured text documents has increased in recent times, the ability to automatically group and organise documents is arguably more important now than ever before. *Text classification*, which can be defined as the automated categorisation of a text documents into a predefined categories based on the content of the documents (Dumais *et al.*, 1998; Sebastiani, 2002), has proven itself to be a good way of achieving this goal (Cooley, 1999; Joachims, 1998; Silva & Ribeiro, 2003).

A typical text classification scenario can be described using an example from the text classification sub-field of *information filtering*. Lanquillon (1999) defines information filtering as *an information seeking process in which non-relevant documents from an incoming stream are rejected according to a specific long-term user interest in such a way that only the relevant documents are presented to the user*. Information filtering can therefore be considered a binary text classification problem where the classification task is to predict if a new document belongs to the "relevant" or "non-relevant" class. To allow a machine learning technique to be used, the documents must first be transformed into a representation suitable for text classification.

2.5.1 Text Representation

An easy, yet effective, way to represent a collection of documents is to use the *Vector Space Model* (VSM) (Salton *et al.*, 1975). In the vector space model each unique term is considered an instance feature and the collection of all features is known as the *vocabulary*. Text classification problems usually have a large vocabulary but each individual document only contains a small subset of the vocabulary. This leads to high dimensional, yet sparse (many features have a value of zero) datasets. Text data is transformed into terms (or tokens) through a process known as *tokenization*. The definition of a term depends on the implementation used, though a common way to tokenize text data is to split the data into tokens based on whitespace characters, such as a space or line break. This is known as the Bag-Of-Words (BOW) (or Set-Of-Words) representation.

Using the previous notation an instance can be represented as, $x_i = \langle x_{i1}, x_{i2} ... x_{i|V|} \rangle$ where each term, x_{ij}, is a weight which reflects the term's perceived importance in a document and V is the vocabulary used. Some common term weighting schemes are listed below:

Binary Weighting is the simplest weighting scheme, where the value of x_{ij} is one if the term j is present in document i and zero otherwise.

Term Frequency (TF) is a common term weighting scheme where the value of x_{ij} reflects the number of times term j is present in document i.

Term Frequency - Inverse Document Frequency (TF-IDF) is probably the most common weighting scheme used in text classification. The Document Frequency (DF) of a given term is the number of documents that term occurs in. The Inverse Document Frequency (IDF) is the log inverse of the document frequency, which results in a large IDF if the document frequency is small. The intuition behind inverting the document frequency is that terms which occur less frequently in the document collection are more likely to be discriminative and thus receive a larger weight. TF-IDF of x_{ij} therefore reflects the number of times term j is present in document i weighted by the number of documents term j is present in, in an overall document collection, typically the training set.

Cooley (1999) performed a comparative study of weighting schemes and found that TF and TF-IDF produced similar results and were typically superior to binary weighting.

2.5.2 Feature Space Reduction

Text data tends to be high dimensional since each unique word is a feature. In fact many text classification datasets have tens of thousands of features, resulting in a dataset with a higher number of features than there are training documents. The high dimensionality of text data can have a detrimental effect on the training and classification time of certain algorithms (Silva & Ribeiro, 2003). This motivates the field of *feature space reduction*, the removal of redundant and noisy terms. Feature space reduction is normally achieved using techniques such as stop word removal, stemming and feature selection.

Stop words, such as “a”, “able”, “about”, “across”, “after” etc. are important for human understanding of texts, but not automated text classification, as they are usually too common to be discriminative. Stop words are either selected from a list of common stop words for the target language[1] or selected based on their frequency across the documents in the text collection being used. Another way to find non-discriminative terms is by using *feature selection*. Feature selection techniques base the decision of which terms to include on statistics derived from the text collection. One of the simplest approaches, document frequency thresholding, removes terms with a document frequency below a given threshold. Other feature selection methods include information gain (Quinlan, 1986), term strength (Yang & Wilbur, 1996) and mutual information (Church & Hanks, 1990), but they are more computationally expensive. Terms with a common stem will usually have similar meanings. *Stemming* reduces the vocabulary by trimming words to their stem. For example the Porter stemmer (Porter, 1980) stems “connected”, “connecting” , “connection”, “connections” to “connect”.

Silva & Ribeiro (2003) found that stop word removal reduced the number of features and improved classifier performance significantly, while stemming reduced the number of terms used but did not appear to have a significant impact on classification performance. Yang & Pedersen (1997) found that advanced feature selection techniques like information gain and term strength did not seem to improve text classification significantly when compared to the much simpler document frequency thresholding approach. Feature selection can sometimes disimprove classifier performance. Three separate studies have found that SVM classifiers perform better on text classification tasks if feature selection is not performed, as text data contains few extraneous

[1]For one example of a list of English language stop words see: `http://www.textfixer.com/resources/common-english-words.txt`

features (Cooley, 1999; Joachims, 1998; Yang & Pedersen, 1997).

2.6 Conclusion

Machine learning has been very successful in a wide array of prediction tasks including text classification, a domain which is likely to experience concept drift and high labelling costs, the type of problems this thesis is trying to address.

Text classification is a sub-field of machine learning, this chapter has therefore explored some of the fundamentals of machine learning. The first section looked at how machine learning fits into the knowledge extraction process CRISP-DM. Most of the focus was on the data preparation to evaluation phases. The data preparation step deals with how data has to be collected and prepared before it can be used as an input to the machine learning algorithm. In the modelling step the data is fed into a machine learning algorithm, which can then produce predictions. There are various modelling algorithms with different strengths and weaknesses. The appropriateness of a machine learning algorithm can be evaluated in the fifth step of the CRISP-DM process, evaluation. In the evaluation step the model is evaluated using an evaluation metric, such as classification accuracy. If the class distribution of the data is significantly skewed then an evaluation metric like average class accuracy is normally better suited.

This chapter also looked at how the data preparation step of the CRISP-DM can be adjusted to allow text data to be used. Text data can be transformed into a format appropriate for machine learning by using the vector space model. The number of features in text data is normally very high. The accuracy and efficiency of trained models can usually be improved by reducing the number of features using feature selection techniques such as stop word removal, stemming and feature selection. SVM classifiers are particularly well suited for text classification both in terms of their ability to handle text data and their high prediction accuracy.

The next chapter addresses concept drift in greater detail, including the issues associated with concept drift and how it can be handled. It will also cover the state of the art in handling concept drift with a reduced amount of labelled data, as this is the main issue addressed in this thesis.

Chapter 3

Concept Drift

The previous chapter discussed how a set of labelled instances can be used to train a classifier, which can then be used to predict the class of a set of unlabelled data. In some prediction tasks the unlabelled data of interest can be collected into one dataset and then labelled by the classifier, however, in many cases the data arrives in a stream. In these cases an *online classifier* is required. Real life data streams often contain concept drift, which may present a problem for online classifiers. A concept can be formally defined as a set of instances where the function generating the instances, known as a *source*, is stationary (Gama *et al.*, 2004; Narasimhamurthy & Kuncheva, 2007; Zliobaite, 2009). Concept drift can therefore be defined as a change in the source generating the data. Each time a new instance is being processed an assumption needs to be made about its source, either the source is assumed to be the same as the last source, or else the source is assumed to be inferable by estimation or prediction (Zliobaite, 2009).

However, a change in the source does not necessarily mean that the classifier needs to be adapted. Concept drift will therefore be defined in this thesis as a change in source which causes a deterioration in the classifier performance. Under this definition, concept drift renders once accurate classifiers less than optimal, which motivates the field of concept drift handling. There are many approaches for handling concept drift, however the choice of an appropriate concept drift handling algorithm is contingent on some of the characteristics of the expected change in concept. This chapter will therefore begin with an examination of some of the issues related to concept drift before reviewing some representative approaches to handling concept drift.

The remainder of this chapter is organised as follows, Section 3.1 discusses the different ways in which the function generating the data can change to give rise to concept drift and Section 3.2 looks at how the change in concept can manifests itself. Section 3.3 reviews some common approaches to handling concept drift. Section 3.4 covers the main subject of this thesis, the issues and approaches involved with handling concept drift in a scenario where the true classes can be obtained, but it is expensive to do so. This is followed by the chapter conclusion in Section 3.5.

3.1 The Causes of Concept Drift

Gao *et al.* (2007) proposes that the main causes of change in concept are either an inherent change in the data stream, known as a *feature change*; a change in the decision boundary, known as *conditional change*; or a combination of both, known as *dual change*.

These causes of concept drift can be formally defined using probabilities. If P is the probability of a given event, x is an unlabelled instance and y is a class label, the causes of concept drift can be:

- **Feature change**: a change in the probability of the occurrence of a particular set of feature values, i.e. a change in $P(x)$. Feature change is also known as *virtual concept drift* (or population shift (Kelly *et al.*, 1999)). It has been argued that the fact that the drift might be virtual is not important from a practical point of view as the the model needs to be adapted regardless of whether the drift is virtual or not (Tsymbal, 2004; Zliobaite, 2009).

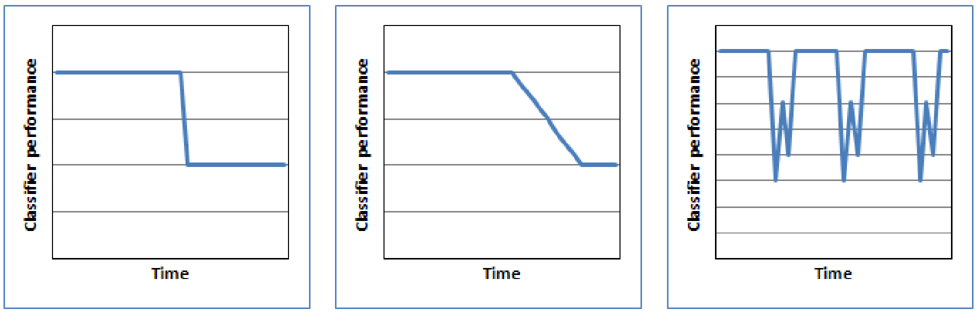
- **Conditional change**: a change in the conditional probability of a class given a particular set of feature values, i.e. a change in $P(y|x)$. If $P(y|x)$ changes but $P(x)$ does not, the drift cause is known as a *hidden context* (Widmer & Kubat, 1998). This type of change can occur when all the attributes needed to predict the class are not included as features.

- **Dual change**: a feature and conditional change, i.e. a change in both $P(x)$ and $P(y|x)$.

Regardless of the cause, concept drift manifests itself in a number of different ways. These have been categorised into different types based on the resultant effect on classifier performance and will be described in the next section.

3.2 Types of Concept Drift

Concept drift can be broadly categorised into three types based on the impact on classifier performance over time (Kuncheva, 2008; Tsymbal, 2004):



(a) Concept shift (b) Concept drift (c) Trends

Figure 3.1: The effect of different types of concept drift on classifier performance over time.

Sudden shift occurs when the concept changes abruptly. For example, in a news filtering application, the death of a prominent media figure can make articles about that person relevant to the user, where they were non-relevant before. This often manifests itself as a sudden drop in classifier performance, as illustrated in Figure 3.1a.

Gradual drift generally happens when the concept gradually changes from one concept to another. For example, articles about an election might gradually become less relevant to a user after the election. This normally results in a gradual degradation in classifier performance as shown in Figure 3.1b. Helmbold & Long (1994) define the extent of drift as the probability that two subsequent concepts disagree on a randomly drawn example and Stanley (2003) categorises drift into moderate and slow drift, depending on the rate of change.

Recurring trends/contexts are trends or patterns which repeat themselves at intervals and might look something like Figure 3.1c. Recurring trends are commonly found in seasonal data (Widmer & Kubat, 1998).

Classifier performance can also experience temporary fluctuations, such as noise or blips (Kuncheva, 2008), but these will not be consider fully fledged types of concept drift. Noise are changes that are deemed non-significant which can enter the data for various reasons such as, imprecision in data recording or mislabelled instances. A blip is a“rare event”, or an outlier. Examples include fraudulent transactions, network intrusion and rare medical conditions. Noise and blips usually cause small fluctuations in classifier performance, but not a long term trend.

Certain concept drift handling techniques work well on a few types of drift, but perform poorly on others. By understanding the types of concept drift that are expected to occur in a data stream an appropriate drift handling approach can be selected. However, it is worth noting that data streams can experience different types of concept drift simultaneously, such as gradual recurring drift, as the above categories are not mutually exclusive. This work is mainly focused on sudden shift and gradual drift, but will reference recurring trends, noise and blips when appropriate.

3.3 Handling Concept Drift

Chapter 1 notes that concept drift handling approaches can be categorised based on when the classifier is rebuilt. The model is either adapted continuously at regular intervals, a continuous rebuild, or is only adapted when it is suspected that a sufficiently large change in concept has occurred, a triggered rebuild. This is an important distinction which will be referred to often, however, it is not the only important characteristic of a concept drift handling approach. Zliobaite (2009) outlines two other important characteristics upon which concept drift approaches should be categorised on: how the classifier is adapted and what causes it to adapt.

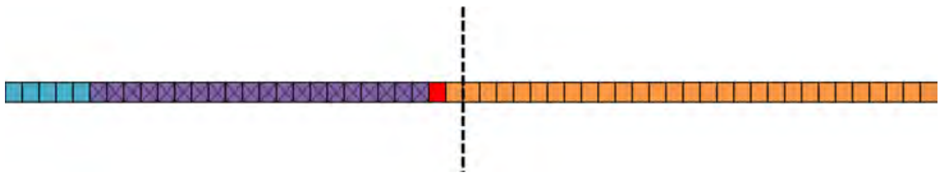
Tsymbal (2004) proposes that a model can be adapted in one of three ways: (1) by changing the data in the training set (instance selection), (2) by changing the ensemble rules or (3) by changing the parameters to the model. The adaptation is nearly always driven by some heuristic which influences how the model is adapted. This heuristic will be referred to as an *indicator* and an indicator over time as a *signal*. Klinkenberg & Renz (1998) introduce three sources of indicators: (1) performance measures (such as the classifier error rate), (2) properties of the classification model and (3) properties of the data. An indicator can be used by both continuous and triggered approaches. A continuous approach uses the indicator to decide how to adapt the model, and a triggered approach uses the indicator to decide when to rebuild.

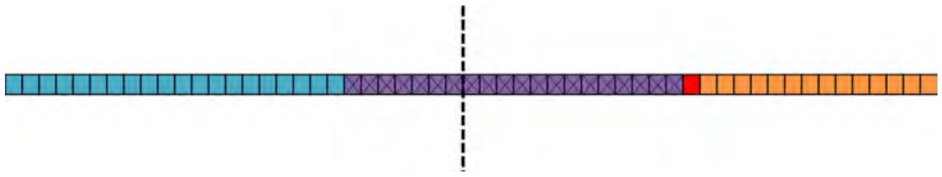
Concept drift handling approaches can either be *instance-based* or *batch-based*. Instance-based concept drift handling approaches process the instances as they arrive whereas batch-based concept drift handling approaches collect instances into batches before being processed. However, the way the instances are processed will not be considered a characteristic of the same importance as those discussed previously and will only be mentioned when appropriate. The remainder of this section will instead review concept drift handling approaches by analysing each approach using all three criteria outlined above (when, why and how adaptation takes place), grouped by how they adapt to changes in concept.

3.3.1 Instance Selection Based Concept Drift Handling Approaches

Instance selection approaches handle concept drift by selecting which instances to train on, based on their perceived relevance to the current concept. The most common instance selection technique is a variation of the continuous rebuild approach known as a sliding window. In sliding window approaches the training set on which the classifier is trained is known as the *training window*. The data in the training window is refreshed periodically with new data, based on the assumption that recent data is more likely to be from the same concept as the current concept.

With sliding windows new instances are added to the front of the training window as they arrive. Once the new training window has been formed the classifier is rebuilt, making sliding window approaches continuous rebuild approaches. The number of instances in the training window is known as the size of the window. A *fixed sized sliding window* maintains a fixed number of instances in the window. When new instances are added to the front of the window, an equal number of instances are removed from the end of the window. Figure 3.2 illustrates how a fixed size sliding window behaves on a data steam containing concept drift. The example shows a data stream split into batches, but the principal of a sliding window can be applied to an instance based data stream in a very similar manner. Each box is a


(a) A fixed size sliding window at time t.


(b) A fixed size sliding window at time $t + 15$.

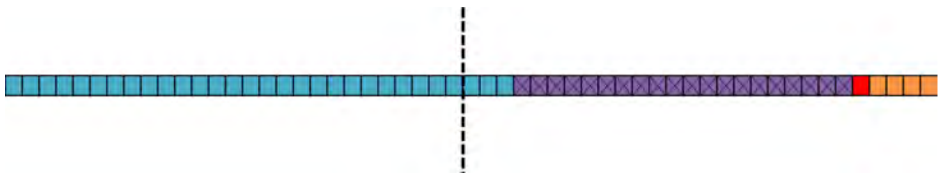
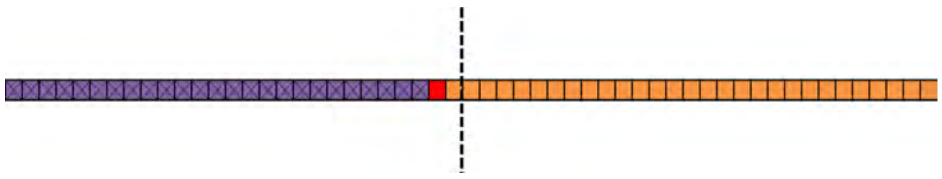

(c) A fixed size sliding window at time $t + 30$.

Figure 3.2: An illustration of how a fixed size sliding window handles concept drift.

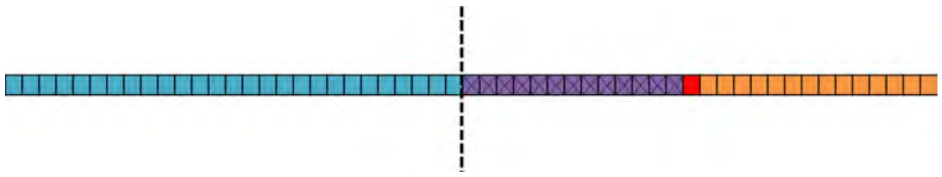
batch of data in a data stream, the red batch is the current batch, the batch of data that the classifier is currently processing, and the blue batches are "historic data", batches which the classifier has processed in the past. The purple batches are historic data which forms the training data of the classifier and the orange batches are unlabelled data yet to be processed. At each batch the classifier is trained on the data in the training window and classifies the unlabelled data in the current batch. After the current batch has been classified it is added to the training window and the oldest batch is removed from the training window. In Figure 3.2a the sliding window is one batch away from a change in concept, marked by the dashed vertical line. Figure 3.2b shows the same data stream after the training window has moved 15 batches to the right. The training window now contains data from both before and after the change in concept, which might result in a drop in classifier performance. Finally Figure 3.2c shows the process 15 batches further along. At this stage the data in the training window is composed solely of data from the new concept which should result in a rebound in classifier performance. The size of the sliding window dictates the properties of the fixed size sliding window. A small window reacts to a change in concept faster than a large window, but is more sensitive to noise, whereas a large window tends to perform better when the concept is stable.

A variable size sliding window allows the size of the window to change, rather than remain fixed. With a variable size window the window is usually allowed to grow when the concept is stable (Figure 3.3a) and shrink when there is a suspicion that the concept has changed (Figure 3.3b).

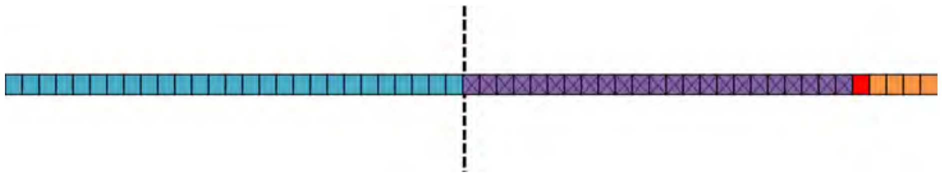
One of the earliest sliding window approaches was the Floating Rough Approximation (FLORA) family of concept drift handling approaches starting with FLORA (Kubat, 1989). FLORA uses a sliding window with a fixed window size. This was improved in FLORA2 (Widmer & Kubat, 1992) which uses a variable size window with the window size adjusted based on the classifier error rate. FLORA3 and FLORA4 (Widmer & Kubat, 1996) improved



(a) A variable size sliding window at time t.



(b) A variable size sliding window at time $t + 15$.



(c) A variable size sliding window at time $t + 30$.

Figure 3.3: An illustration of how a variable size sliding window handles concept drift.

the handling of recurring trends and noise respectively.

Klinkenberg & Renz (1998) introduced another notable sliding window approach which uses accuracy, precision and recall indicators to adjust the window size. The average value of each indicator is calculated over a number of previous batches. If any of the indicators for the current batch are above the average value for that indicator for the previous batches a change in concept is flagged. The difference between the current indicator value and the average value obtained from the previous batches determines by how much the training window is shrunk. If the difference is large, then a concept shift is suspected and the window is shrunk to the current batch. Otherwise a more gradual change in concept is presumed and the window size is shrunk by a user defined constant.

Gama *et al.* (2004) also created a sliding window approach using an error based signal. If the error rate is above a *warning threshold* a change in concept is suspected and a new classifier is trained in parallel with the current classifier. If the error rate goes above a second threshold, the *error threshold*, a change in concept is declared and current classifier is replaced by the one trained from where the error rate exceeded the warning threshold.

Kuncheva (2009) uses statistical tests on the error rate to adjust the window size. When the error rate is greater than the mean error rate plus three times the standard deviation the window collapses to the current batch, otherwise the window grows.

The above mentioned sliding window approaches all use the error rate of a classifier as an indicator which influences the window size. This is a very common way of selecting the window size (other examples using this idea includes (Baena-García *et al.*, 2006; Klinkenberg & Joachims, 2000; Nishida & Yamauchi, 2007)), and tends to handle concept drift in an intuitive and effective manner. However, they require that the classification error can be calculated.

Vorburger & Bernstein (2006) use an adaptation of Shannon's entropy to calculate a window size indicator. The concept drift handling is achieved using a sliding window, when the entropy goes below a threshold the training window collapses, otherwise it grows.

Instance based selection approaches attempt to handle concept drift by selecting which instances are used to train the classifier. The most common selection technique is a sliding window variant. Sliding window based techniques are intuitive, do usually not require too much parameter tuning and are based on an the reasonable assumption that instances which are chronologically close are likely to belong to the same concept. However, sliding window techniques are restricted by the classifier used. For example some simple classifiers are fast to update, which is important for processing data streams, but may suffer in terms of accuracy. Ensembles have been shown to be able to achieve high accuracy on some non-evolving data, and can be altered to handle concept drift. The next section will look at how ensembles can be used on data exhibiting concept drift.

3.3.2 Ensemble Based Concept Drift Handling Approaches

Section 2.3.4 introduced the idea of ensembles, a collection of classifiers whose individual predictions for an unlabelled instance are combined using fusion rules to form the overall prediction of the ensemble. The approaches discussed in this section attempt to handle concept drift using ensembles. This is generally achieved by adjusting the contribution of each base classifier towards the overall prediction depending on the classifiers perceived relevance to the current concept.

Concept Drift Committee (Stanley, 2003) and Dynamic Weighted Majority (Kolter & Maloof, 2003) are two of the earliest approaches which use ensembles to handle concept drift. Both approaches weight each base classifier's votes based on its classification accuracy on recent data. When a base classifier's classification record falls below a threshold it is removed from the ensemble and a new classifier, trained on recent data, is added. Other approaches to handling concept drift with ensembles include: weighting base classifier's vote by their accuracy on the k nearest neighbours of a test instance (Tsymbal *et al.*, 2008) and weighting by the cost of misclassification coupled with their probability of misclassification (Wang *et al.*, 2003).

However, not all ensemble approaches use some form or classifier accuracy to weight the base classifiers, Conceptual Clustering & Prediction is an ensemble based concept drift handling approach which clusters batches of data together into concepts and forms one classifier per concept (Katakis *et al.*, 2010). Each batch of data is classified by the classifier representing the concept which the previous batch belonged.

Ensemble based concept drift handling approaches have been shown to achieve strong classification accuracy and can be implemented in such a way that each base classifier becomes an expert in a particular part of feature space or concept. However, ensemble based concept drift handling approaches tend to be more computationally complex than a single classifier and require many parameters.

3.3.3 Model Parameter Based Concept Drift Handling Approaches

The third high-level approach to handling concept drift is through changing the model parameters. One way to do this is through instance weighting. In most machine learning algorithms each instance contributes equally to the hypothesis. Certain classifiers, like SVMs allow instances to be weighted to create a bias towards certain instances. In (Klinkenberg, 2004) a couple of weighting schemes were used in conjunction with an SVM, including weighting instances according to age and competence in regard to the current concept. However, the results seemed to indicate that instance weighting compared unfavourably to a sliding window approach.

Another way to handle concept drift using model parameters is to modify the classifier's internal structure. Black & Hickey (1999) continuously adapt a decision tree based on the discriminant ability of a time based feature. All instances in the current training window are given the timestamp attribute of "current", when a new batch of data arrives the timestamp attribute for all instances in the new batch is set to "new". The decision tree is then rebuilt with the old training data and the new batch of data. Branches where the timestamp attribute is highly discriminant are considered out of date. The invalid rules (and instances covered by those branches) are pruned from the tree and timestamp feature is removed on the remaining branches.

Model parameter based concept drift handling approaches can offer novel ways of handling concept drift, but their usefulness can be limited by the fact that they can only be applied to certain classifiers.

3.4 Handling Concept Drift in the Context of Expensive Labels

The previous section covered the important approaches to handling concept drift. However all of the approaches discussed so far operate on the assumption that the true labels for all of the instances in the data stream are available shortly after classification. For example sliding window approaches require a training window of labelled data. This means that they need a continuous stream of labelled data as each instance in the stream will be in the training window eventually and the training window needs to be made up of labelled instances. Similarly, many approaches use the error rate to adjust the model, for example by changing the window size or weighting base classifiers in an ensemble. Calculating the error rate requires labelled data, rendering these types of approaches very expensive in terms of labels used.

In some domains the label assumption is justified. For example in short-term stock predictions the true label is available shortly after the model's prediction is made. In other domains, such as information filtering, labelled data does not automatically emerge as part of the process (as they do for the stock example) and are expensive to generate. It is therefore important to consider how concept drift can be handled with a reduced amount of labelled data so as to avoid this expense. However, before looking at existing approaches there are a few issues specific to handling concept drift with expensive labels which require further examination.

3.4.1 Labelling Scenario

The labelling scenario refers to what fraction of the data in the stream is labelled. There are three special cases which should be highlighted: fully, partially and unlabelled data streams.

A fully labelled data stream, is one where all the instances are labelled.

This is the scenario most concept drift algorithms assume. Concept drift handling algorithms which require this assumption to be true include (Baena-García *et al.*, 2006; Bifet & Gavalda, 2007; Nishida & Yamauchi, 2007). In a domain such as spam filtering this is a reasonable assumption because the user will correct the learning algorithm by moving emails to and from the spam folder, or short-term stock predictions, where the change in share price provides the true label shortly after the prediction is made.

A partially labelled data stream, is one where a certain percentage of the instances in the data stream are labelled. For example in quality control a fixed number of instances are sampled periodically and the true label of those instances is supplied to the learning algorithm. Approaches based on this assumption include: (Klinkenberg, 2001; Lindstrom *et al.*, 2010a; Masud *et al.*, 2008; Zliobaite *et al.*, 2011).

An unlabelled data stream, is one where no instances in the data stream are labelled. This scenario is particularly common in areas where the true label of an instance is delayed, such as bankruptcy prediction. The problem with an unlabelled data stream is that there is no up to date training data which can be used to re-train the classifier.

This thesis will mainly focus on a particular type of partially labelled data stream. Namely an unlabelled data stream in which any instance label can be acquired, but the point is to request as few labels as possible. This type of labelling scenario can be found in many real life classification tasks.

For example in information filtering, where the system can select which documents a human expert should label. The key to this type of algorithm is deciding which instances to get labelled. Approaches operating under this assumption include (Lindstrom *et al.*, 2010a; Zhu *et al.*, 2007; Zliobaite *et al.*, 2011).

3.4.2 Detectability

The cause of the change in concept is an important consideration when selecting which concept drift handling approach to use, as some types of concept drift are not detectable without labelled data. Table 3.1 is based on the work in (Zliobaite, 2010) and illustrates the contingency table of possible changes (using the notation from (Gao *et al.*, 2007) described in Section 3.1).

Table 3.1: Concept change detectability contingency table.

		$P(y\|x)$	
		Changes	**Does not change**
$P(x)$	**Changes**	(1) Dual change	(2) Feature change
	Does not change	(3) Conditional change	(4) No concept drift

(1) Dual change (a change in the data and the conditional probabilities) can be detected without labelled data, as the conditional change is also accompanied by a change in the data, which can be detected.

(2) Feature change (a change in the data without a change in the conditional probabilities), also known as virtual concept drift, is detectable without labelled data. For example a detection algorithm which monitors the frequency of key words in a text data stream would be able to detect when the word distribution changes significantly, which might signify a change in

concept, without using labelled data

(3) Conditional change (a change in the conditional probabilities without a change in the data) is always important to handle, but not possible to detect without labelled data. For example, in information filtering a user might find articles about a certain celebrity interesting for a while and then lose interest in that celebrity. In this case there has been a change in the decision boundary (i.e. a change in $P(y|x)$) without a significant change in the data ($P(x)$). Therefore concept drift handling approaches which aim to handle conditional change must use at least some labelled data.

3.4.3 Existing Approaches to Handling Concept Drift in a Partially Labelled Data Stream

This thesis investigates how concept drift can be dealt with in a data stream where obtaining labelled data is expensive.

Chapter 1 has already introduced the two ways concept drift handling approaches tend to reduce the amount of labelled data required, through improvements to their drift detection or drift adaptation component.

The drift adaptation component of concept drift is usually improved using semi-supervised learning. The difference between supervised and semi-supervised learning is that a supervised learner uses labelled data to find the mapping between features and the class label, whereas semi-supervised learning uses both labelled and unlabelled data to train the classifier. This often allows a classifier trained on a small amount of labelled data, and a large amount of unlabelled data to achieve classification accuracies comparable with classifiers trained on all the labelled data (Lewis & Gale, 1994; Tong & Koller, 2002). Semi-supervised learners designed to handle concept drift are typically continuous rebuild approaches where the amount of labelled data required is reduced by using a semi-supervised learner instead of a supervised learner every time the classifier is retrained.

Concept drift detection can be improved by devising a drift detection indicator which does not use labelled data. This indicator can then be incorporated into a triggered rebuild framework meaning that the classifier is only retrained, and a labelling cost incurred, when a significant change in concept is suspected. The approaches in the subsequent literature review will be grouped based on the label reduction approach (semi-supervised or triggered) used.

3.4.3.1 Semi-supervised Learning

Klinkenberg (1999) uses a variable size sliding window, just like in (Klinkenberg & Renz, 1998), but extends the approach by calculating the window size heuristic only on the small number of instances in the data stream which are labelled. Exactly which instances are labelled and therefore used in the window size heuristic calculation is outside the control of the algorithm. The experimental evaluation indicates that this approach is a good starting point as concept drift can be handled without a fully labelled data stream. However, there is scope for improvement if the algorithm is allowed to decide which subset of the unlabelled instances should be labelled.

Active learning (AL) (Cohn *et al.*, 1994) is a powerful way of selecting which instances the classifier would benefit the most from having labelled. The label is only requested for those examples that are deemed to be most informative to the training process.

Zhu *et al.* (2007) combines an ensemble of decision trees with active learning. In this approach a small portion of the data within a batch is selected at random and labelled, after which a classifier is built from the labelled data. The new classifier is added to a fixed size ensemble and all classifiers in the ensemble are weighted according to their accuracy on instances in the batch that were randomly sampled and labelled in the previous step. Active learning is then used to sample the instance that cause maximal classifier variance within the ensemble for labelling. After the sampled instance has been labelled the process takes one of three actions, the classifier can be rebuilt, the labelling process can stop (if the number of instances sampled from the current batch exceeds a user-specified quota) or the ensembles can be re-weighted and a new instance sampled based on classifier variance. Concept drift is handled through the weighting of the base classifiers and by only maintaining a fixed number of classifiers with the best classification accuracy on the labelled data.

Zliobaite *et al.* (2011) uses a variable size sliding window approach to handle concept drift. The reduction of labelled data is achieved by using active learning to select a small number of instances which are labelled and added to the training window. The approach uses a sampling heuristic which ensures that instances close to the decision boundary are more likely to be labelled when a change in concept occurs. Instances further from the decision boundary are sampled when the concept is stable. There is also a component of randomness in the sampling process to ensure that some instances far from the decision boundary are sampled periodically. This is done to ensure that local concept drift occurring far from the decision boundary does not go unnoticed. A user-specified parameter known as the labelling budget works in conjunction with the sampling strategy to control how many instances are sampled over the whole data stream. The window resizing approach introduced in Gama *et al.* (2004) is used to adjust the windows size.

There are also examples of semi-supervised learners which do not use active learning. Masud *et al.* (2008) proposed Semi-Supervised Stream Clustering, a semi-supervised approach which is designed to work on a partially labelled data stream by utilising clustering. In each batch both labelled and unlabelled instances are clustered while ensuring that all labelled instances in a cluster belong to the same class. A k-NN classifier is created from each cluster and the classifier is added to the ensemble of classifiers created from the previous batches. Concept drift is handled by only keeping the classifiers which obtained the highest accuracy on the labelled data in the current batch in the ensemble. This approach is further refined in (Woolam *et al.*, 2009).

3.4.3.2 Triggered Rebuild

The other major approach to handling concept drift with a limited amount of data is to use a triggered rebuild with an indicator not calculated from labelled data. The advantage of a triggered rebuild with this type of indicator is that no labelled data is needed until a change in concept is suspected. This stands in contrast to the semi-supervised approaches in Section 3.4.3.1 where a fixed amount of labelled data is required in every batch regardless of whether the concept is changing or not.

The most common way of creating an indicator not dependant on labelled data is the *two-window paradigm* for event detection (Kifer *et al.*, 2004). The two-window paradigm is a high level approach which compares some summary information about the past behaviour in a *reference window* to the summary information in the *current window.* Approaches vary depending on if the windows have a fixed or sliding starting point and if the windows have a constant or growing size (for a more comprehensive exploration of the two-window framework see (Ada & Berthold, 2011)). The two-window paradigm can be used to create a signal which can be used in both triggered and continuous rebuild approaches. An example of a continuous rebuild approach which uses a two-window paradigm is the aforementioned entropy approach used in (Vorburger & Bernstein, 2006), where the difference in the entropy between the reference and current window is used to control the size of a variable size sliding window. However, the two-window paradigm is mostly used to create an indicator for use in triggered rebuild approaches, some of which will be discussed next.

One of the earliest examples of this type of approach was introduced by Kifer *et al.* (2004). The distribution of a feature inside a reference batch is compared to the distribution inside the current batch to determine if the data in both batches is likely to have been generated by the same underlying process. This is achieved using a statistical distance function based on Chernoff bounds. Sebastião & Gama (2007) take a similar approach but use Kullback-Leibler divergence to measure the difference. Both approaches require the identification of a feature which is sensitive to change in concept. In a dataset such as a text dataset where each document is represented by word frequencies, monitoring the distribution of one word is unlikely to yield satisfactory detection.

Zliobaite (2010) uses a two-window paradigm to compare the posterior class membership probabilities in a reference window to the probability of class memberships in the current test window, using statistical tests to flag significant differences, and hence the occurrence of concept drift. Although this approach have been shown to be capable of detecting concept drift without labelled data it has not been tested as an end-to-end system which detects a change, updates the model and evaluates the resultant classification accuracy.

Lanquillon (Lanquillon, 1999) does develop an end-to-end approach which uses the training data to estimate the classifier confidence range in which the predicted label is likely to be correct. A change in concept is flagged if in subsequent batches the fraction of confidences within that range exceeds the detection threshold of $\mu + (3 * \sigma)$ where μ and σ are the average and standard deviation of first ten fractions in the data stream. However this approach is only evaluated on very simplistic artificial data.

Fan *et al.* (2004a) propose an approach which uses an indicator based on decision tree leaf node statistics. If this indicator is above a threshold, a small number of instances are labelled and based on those labels the error rate is estimated. If the estimated error rate is higher than a tolerable maximum the classifier is reconstructed using the instances labelled in the previous step (for details on how the classifier is updated see Fan *et al.* (2004b)). Huang & Dong (2007) take the detection approach from Fan *et al.* (2004a) but use active learning to select which instances to sample when concept drift is detected.

3.5 Conclusion

This chapter has examined the problem of concept drift. Concept drift is the deterioration in classifier performance due to the non-stationary nature of the concept being modelled. It is caused by change in the data, a change in the decision boundary or both. Changes in concept, regardless of the cause, usually manifests themselves as a deterioration of a model's predictive accuracy. The decline can be either sudden or gradual and might also contain recurring patterns or random noise.

There is a growing body of research dealing with how concept drift can be handled but most of it tends to assume that all the instances in the data stream have their true label available shortly after classification.

Some approaches specifically designed not to require a fully labelled data stream have emerged. These can be divided into semi-supervised and triggered rebuild approaches[1].

[1](Fan *et al.*, 2004a) is categorised as a triggered rebuild approach, but it could be considered an hybrid approach as it also adapts the classifier with a reduced amount of data.

Semi-supervised approaches focus on creating a classifier from a partially labelled data stream, but tend to be computationally expensive and require extensive parameterisation. Triggered rebuild approaches focus on detecting changes in concept without needing labelled data, but tend to be designed for a specific classifier or data type.

One semi-supervised, and one triggered rebuild approach aimed at addressing these shortcomings will be presented in forthcoming chapters, however, the next chapter will look at how both of these approaches will be evaluated.

Chapter 4

Experiment Methodology

The overarching goal of this thesis is to examine how concept drift can be handled with as little labelled data as possible. This goal can be achieved using a continuous or triggered classifier rebuild strategy. A continuous rebuild approach, Decision Value Sampling (DVS) and a triggered rebuild approach, Confidence Distribution Batch Detection (CDBD) were briefly introduced in Chapter 1. How these approaches work and how they compare to other approaches will be covered in further detail in the next two chapters. The methodology used to evaluate both approaches is very similar as they are both designed to work on a partially labelled data stream using as few labels as possible.

This chapter will outline the aforementioned methodology and is organised as follows: Section 4.1 describes the datasets used, followed by Section 4.2 which covers the experimental set-up. Section 4.3 details the evaluation measures and, finally, a conclusion is given in Section 4.4.

4.1 Datasets

Concept drift is often present in real data. Chapter 1 gave the examples of weather predictions, customer buying preferences and changes in users' interests in information filtering. Other common examples of data which is likely to experience concept drift include financial (Abdullah & Ganapathy, 2000), biological (Tsymbal *et al.*, 2006) and social media (Wang, 2010) data. A common source for machine learning datasets on which to perform experimentation is the University of California, Irvine, Machine Learning Repository (Frank & Asuncion, 2010)[1], which contains a large collection of datasets of varying sizes and characteristics. However, most of the publicly available datasets are not datasets which exhibit concept drift. The lack of appropriate datasets is not a problem exclusive to the concept drift domain[2], but it seems to be particularly prominent in this area. This can be partly attributed to the fact that many domains where concept drift is typically found, such as the financial domain, are constrained by privacy and legal concerns (Kennedy *et al.*, 2011; Narasimhamurthy & Kuncheva, 2007).

Another issue with real data is that the properties of the changes in concept contained in the data are usually not known. It can be hard to determine when the concept changed, at what rate it changed, what caused the change and so on. Without this crucial information developing and evaluating an

[1]`http://archive.ics.uci.edu/ml`

[2]For a more through discussion about the lack of dataset sharing in the academic community as a whole see Fischer & Zigmond (2010).

algorithm for dealing with concept drift can be problematic.

A second option which overcomes many of the problems with real data is artificial data. Artificial data exhibiting concept drift can be controlled so that important properties of the change in concept are known, and is seldom as constrained by privacy and legal issues as real data is. While it is paramount that concept drift handling approaches are evaluated on real datasets exhibiting natural concept drift, it is also useful to evaluate approaches on artificial datasets with controlled concept drift.

Artificial datasets exhibiting concept drift tend to be one of two types: *synthetic data* or *drift induced data* (Lindstrom *et al.*, 2008). This thesis focuses mainly on drift induced data but will also briefly cover synthetic data.

4.1.1 Synthetic Data

Synthetic datasets are generated algorithmically in such a way as to ensure that concept drift occurs. One of the most widely used synthetic approaches to generating datasets is the STAGGER approach (Schlimmer & Granger, 1986), which will be used to illustrate how synthetic datasets tend to be generated.

The STAGGER dataset has three feature:, $size \in \{small, medium, large\}$, $color \in \{red, green, blue\}$ and $shape \in \{square, circular, triangular\}$. A dataset is generated by creating instances where each feature value is randomly selected from one of the allowed values, e.g. $\{small, blue, square\}$. The instances are then assigned their true label according to specific concept rules, for example instances with the features $size = small$ and $color = red$ are assigned to one class while all other instances are assigned to the other class (STAGGER datasets focus on binary classification problems). Changes in concept are introduced by changing the concept rules over time. The rules based labelling method used in the STAGGER approach makes it easy to generate datasets which exhibit sudden concept shift and recurring trends, as a change in rules produce a sudden change and rules can be reused. However, the labelling method makes it hard to produce a gradual change.

Another common way to create synthetic concept drift datasets is the moving hyperplane (Kolter & Maloof, 2003) approach. In this approach the instances are separated using a hyperplane and labelled based on what side of the hyperplane they fall. Sudden concept drift is introduced by abruptly moving the hyperplane, whereas gradual concept drift is introduced by rotating the hyperplane over time. The rate of drift can be controlled by how quickly the hyperplane is rotated. Recurring trends can also be simulated using by re-using hyperplane positions.

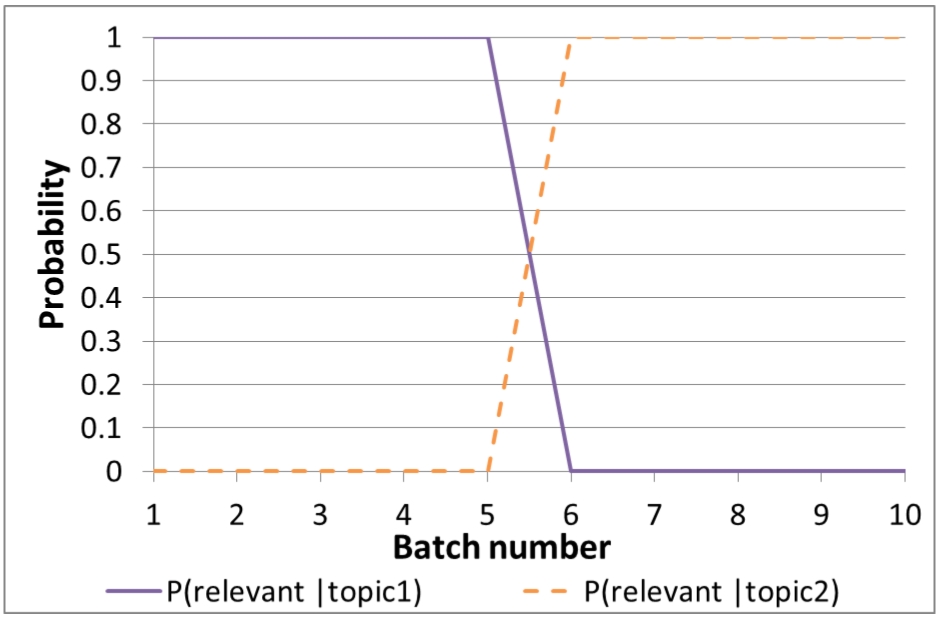
In both the STAGGER and moving hyperplane approaches noise and blips can be introduced by intentionally mislabelling instances. The mislabelling manifests itself as blips if it occurs infrequently, and noise if it is sustained. Both the STAGGER and moving hyperplane approaches also share the fact that they introduce conditional change rather than feature change.

Synthetic data allows a larger degree of control than real and drift induced data. However, the author believes that the process used to generate synthetic data makes it inherently sterile and prevents it from capturing the nuances of real concept drift problems. Therefore, the evaluations in this thesis all use drift induced, rather than synthetic data, as it is the author's opinion that this strikes a good balance between realism and control.

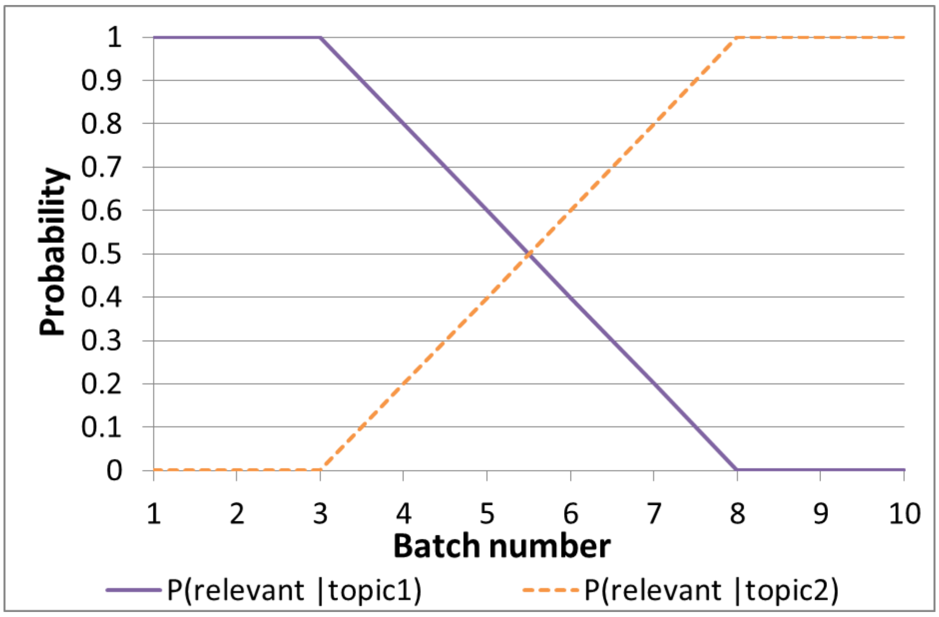
4.1.2 Drift Induced Data

Drift induced datasets are created by taking real datasets, which may not contain a significant amount of concept drift, and tweaking them to introduce controlled concept drift. A change in concept can be induced in real data by using instances from a real dataset and adjusting the function which provides instances with their true label. The process can be illustrated using an example from information filtering where the classification task is to predict if a text document is relevant to a particular user, given its content.

A classification problem can be created from a real text dataset by assigning a document's true label based on what topic it belongs to. For example in a collection of news articles each article might be associated with a topic, such as "Business" or "Sport". All of the articles of one topic, the *target topic*, are labelled as "relevant" to the user and the other documents are labelled as "non-relevant". Controlled concept drift is introduced by adjusting which topic is the target topic. Sudden concept shift can be introduced by swapping the topics some way into the stream, so the target topic becomes the non-target topic and vice versa, which is the approach used in (Ada & Berthold, 2011; Kuncheva, 2009).



(a) Concept shift.



(b) Concept drift.

Figure 4.1: Using topic probabilities over time to introduce changes in concept.

A more general way to introduce concept drift in a text data stream is to use a probabilistic labelling function which assigns document labels based on their topic. If the probability of a given topic being relevant is high, then the labelling function is likely to set its true label to "relevant". Conversely, if the probability of the topic being relevant is low then it is likely to be labelled as “non-relevant”. Drift is induced by adjusting the probabilities over time. The simplified example in Figure 4.1a show how the topic probability can be adjusted to create a sudden concept shift whereas Figure 4.1b shows how gradual concept drift can be simulated.

The treatment of the non-target topic is an important consideration as it dictates if conditional or feature change is induced. The next two sections will discuss this in further detail and show how conditional and feature change datasets can be created.

4.1.2.1 Creating a Conditional Change Dataset

Conditional change can easily occur in information filtering tasks. For example, in the scenario described in the previous chapter where articles about a presidential candidate might become less relevant after the election there was not a change in the data itself, but rather a change in the mapping between the document and its label, i.e. conditional change.

Conditional change is induced in text datasets by labelling all non-target topics as “non-relevant”, including topics which will become target topics at another stage. The process can be illustrated using the example in Figure 4.2 which depicts a stream of instances where the colour signifies what topic an instance belongs to on the first row, and what true label an instance is assigned on the second row. The labelling function used is in this example is $P(relevant|topic) = 1$ if the topic is the target topic and $P(relevant|topic) = 0$ if the topic is not the target topic. This allows a concept shift to be induced by changing which topic is the target topic.

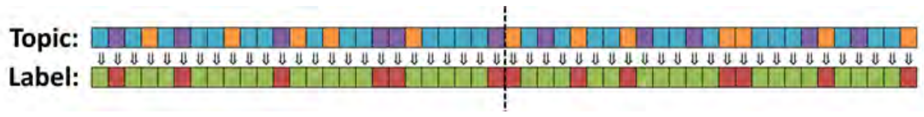


Figure 4.2: The labelling process for conditional change datasets.

In Figure 4.2 the purple instances belong to the topic that at first is considered the target topic, orange instances belong to the topic that becomes the target topic after concept drift has occurred and blue instances belong to a topic which will never become a target topic. Before the change in concept (the dashed vertical line), purple instances are assigned the label "relevant" (a red box in the second row), while orange and blue instances are assigned the label "non-relevant" (a green box in the second row). After the change in concept the target topics are reversed. This approach has been used extensively in previous research, for example in (Klinkenberg, 1999, 2001, 2004; Klinkenberg & Renz, 1998).

4.1.2.2 Creating a Feature Change Dataset

Feature change can also occur in information filtering, for example when articles of a new topic appear. In this scenario the classifier might be unable to accurately predict the class of documents from the second topic as there are no documents from the second topic in the training data. This mimics situations likely to occur in real data where the training data may only contain a partial concept.

In feature change text datasets, non-target topics which will become target topics at another stage never overlap with the current target topic. This is, again, best illustrated using an example. Figure 4.3 shows another data stream and how the topics map to true labels.

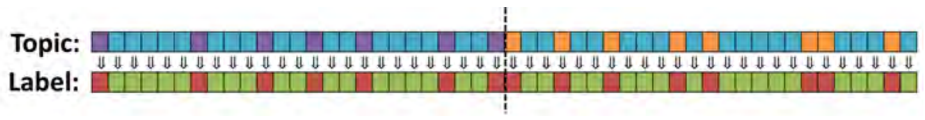


Figure 4.3: The labelling process for feature change datasets.

However, in this example there are no instances belonging to a target topic in the part of the stream where they are not the current target topic. Purple instances are only found before the change in concept takes place, and orange instances are only found after the change in concept took place. This approach is used in (Lanquillon, 1999).

4.1.3 Our Datasets

The experiments described in the later chapters of this thesis are anchored in the information filtering domain, as text data tends to come in large data streams with high labelling costs. Our evaluations are performed on both real-world datasets from the spam filtering domain and artificial datasets generated from large corpora of text documents.

4.1.3.1 Real Datasets

The real-world datasets used in the experiments in this thesis are two spam filtering datasets[1], $Spam_1$ and $Spam_2$ introduced in (Delany *et al.*, 2005). These were both collected from real users' email accounts over a period of time. Spam filtering can be considered a type of information filtering as the

[1]http://www.dit.ie/computing/staff/sjdelany/datasets

classification task is to classify an email as *ham* (relevant) or *spam* (non-relevant) to a given user. The spam datasets are characterised by very high class imbalance, and like many real datasets, a lack of knowledge about the concept drift cause and type.

4.1.3.2 Drift Induced Datasets

The experiments in this thesis will also use artificial datasets, to allow for control for factors like drift type and cause. Drift induced datasets are used, as they retain some of the interesting artefacts which fully artificial data might lack, such as underlying *natural drift* (in addition to that induced artificially), and therefore lead to more informative evaluations. The drift induced datasets are based on three text corpora: *Reuters-21578*[1], *20 newsgroups*[2] and news sources.

The Reuters corpus is a collection of 21,578 text articles which appeared on the Reuters newswire in 1987. The documents are tagged with one or more topics. The two most common topics in the Reuters collection are both business related, namely "corporate acquisitions" ("acq") and "earnings" ("earn").

The 20 newsgroups corpus contains nearly 20,000 newsgroup documents across 20 different newsgroups. Each text document belongs to a newsgroup, such as "comp.graphics" or "rec.motorcycles". Some newsgroups can be grouped together as they belong to the same parent group. For example "comp.os.ms-windows.misc", "comp.windows.x" and "comp.graphics" all

[1]http://www.daviddlewis.com/resources/testcollections/reuters21578

[2]http://people.csail.mit.edu/jrennie/20Newsgroups

belong to the same parent group “comp” (computers).

The news sources corpus is a collection of just over $70,000$ news articles collected from various sources over a period of two months in 2011. Each article belongs to one topic, such as “Sports”, “Business” or “Politics”.

Chapter 3 discusses some of the important characteristics of concept drift, such as drift cause and type. This thesis will consider four factors when generating a drift induced text dataset: (1) corpora (Reuters, 20 newsgroups or news sources), (2) drift cause (feature or conditional change), (3) drift type (sudden or gradual drift) and (4) class distribution (balanced or imbalanced). The first three variables have been discussed previously, while the issue of class distribution has only been mentioned briefly so far. Two distinct cases are considered, a balanced data stream where the class distribution is a fixed, uniform distribution, and an imbalanced data stream where the class distribution changes over time.

These four variables can be varied to generate 24 datasets, listed in Table 4.1. The remainder of this thesis will use the dataset names in the ID column as shorthand for the dataset generated using that specific combination of variables. Not all combinations are necessarily needed when evaluating concept drift approaches. For example, triggered rebuild approaches are usually only evaluated on concept shift datasets with feature change. All permutations have been included, but the discussions about the specific datasets used in each experiment will be delayed until the actual experiment descriptions.

The remainder of this section will instead focus on the process used to

Table 4.1: Summary of the properties of our drift induced datasets.

ID	Corpus	Drift Type	Class Distribution	Drift Cause
$20NG_{SBF}$	20 newsgroups	Sudden	Balanced	Feature
$20NG_{GBF}$	20 newsgroups	Gradual	Balanced	Feature
$20NG_{SIF}$	20 newsgroups	Sudden	Imbalanced	Feature
$20NG_{GIF}$	20 newsgroups	Gradual	Imbalanced	Feature
$20NG_{SBC}$	20 newsgroups	Sudden	Balanced	Conditional
$20NG_{GBC}$	20 newsgroups	Gradual	Balanced	Conditional
$20NG_{SIC}$	20 newsgroups	Sudden	Imbalanced	Conditional
$20NG_{GIC}$	20 newsgroups	Gradual	Imbalanced	Conditional
$Reuters_{SBF}$	Reuters	Sudden	Balanced	Feature
$Reuters_{GBF}$	Reuters	Gradual	Balanced	Feature
$Reuters_{SIF}$	Reuters	Sudden	Imbalanced	Feature
$Reuters_{GIF}$	Reuters	Gradual	Imbalanced	Feature
$Reuters_{SBC}$	Reuters	Sudden	Balanced	Conditional
$Reuters_{GBC}$	Reuters	Gradual	Balanced	Conditional
$Reuters_{SIC}$	Reuters	Sudden	Imbalanced	Conditional
$Reuters_{GIC}$	Reuters	Gradual	Imbalanced	Conditional
NS_{SBF}	News sources	Sudden	Balanced	Feature
NS_{GBF}	News sources	Gradual	Balanced	Feature
NS_{SIF}	News sources	Sudden	Imbalanced	Feature
NS_{GIF}	News sources	Gradual	Imbalanced	Feature
NS_{SBC}	News sources	Sudden	Balanced	Conditional
NS_{GBC}	News sources	Gradual	Balanced	Conditional
NS_{SIC}	News sources	Sudden	Imbalanced	Conditional
NS_{GIC}	News sources	Gradual	Imbalanced	Conditional

generate all of the drift induced datasets listed above. The process is based on the drift induction techniques discussed earlier in Section 4.1.2. Figure 4.4 shows a high level overview of the process. The process can be broken into two phases, *data pre-processing* and *drift induction*.

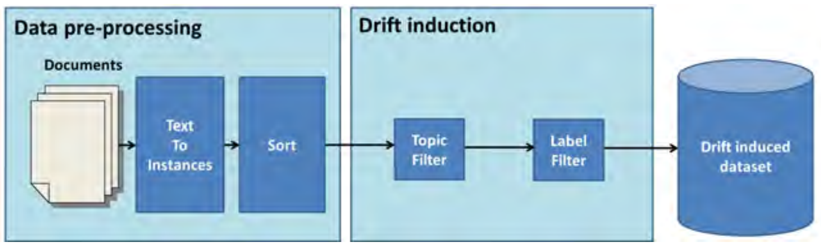


Figure 4.4: A high level overview of the process used to generate drift induced datasets.

The data pre-processing phase contains the following steps:

1. Convert the documents in the chosen corpus into a Bag-Of-Words (BOW) representation, weighting each term using term frequency.

2. Apply stop-word removal and Porter's stemming (Porter, 1980) to reduce the number of features.

3. Sort the instances chronologically, based on their timestamp.

The data pre-processing phase produces a collection of instances suitable for text classification, however, there is no guarantee that the data will exhibit concept drift. The next phase deals with both the induction of controlled concept drift and handling class distributions. In the drift induction phase instances are processed sequentially, one at a time through two filters. The filters can either discard an instance or allow it to pass through the filter. If an instance passes through the two filters it is added to a collection of instances which form the drift induced dataset.

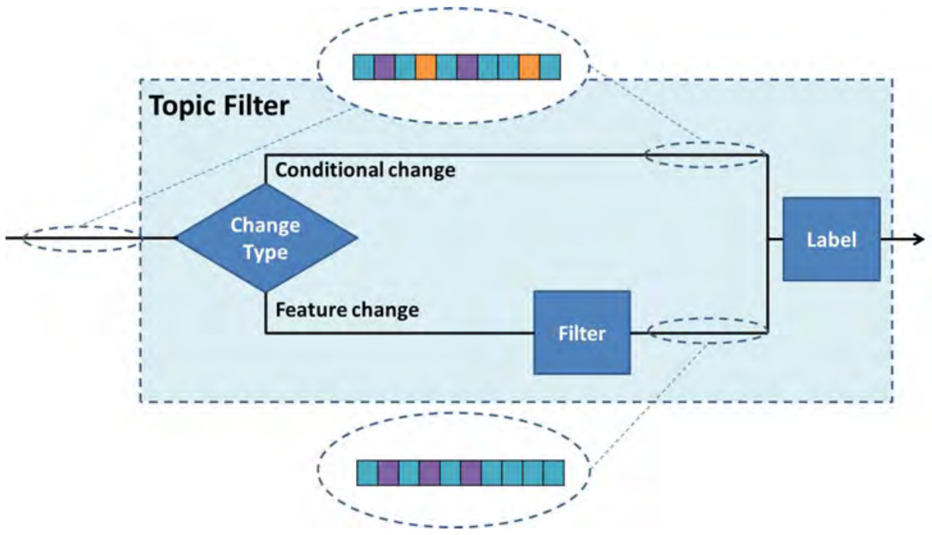


Figure 4.5: The topic filter.

The *topic filter* filters out instances based the topic the instance belongs to. Figure 4.5 shows how the topic filter works. The coloured blocks represent instances being passed through the topic filter. The purple, orange and blue blocks represent instances belonging to “Topic 1”, “Topic 2” and “Topic 3” respectively. An instance might be discarded based on its topic if a feature change dataset is being generated whereas the data stream will be left unmodified if a conditional dataset is being generated. The exact mechanism used to parameterise the topic filter and determine if an instance will be filtered out will be discussed shortly. The topic filter also provides the true label of the instance by labelling it probabilistically based on its topic. Concept drift is introduced by varying the topic filter parameters over time.

If the instance is not filtered out by the topic filter then the newly labelled instance is passed through the *distribution filter*. The distribution filter maintains a record of the label of the instances it allows to pass through and can dynamically adjust itself to ensure a given class distribution in the resultant data stream. This thesis focuses on two distinct distribution filters, the imbalanced filter and the balanced filter. Figure 4.6 shows the distribution filter in action. In this diagram the green and red blocks represent instances that have been assigned the true labels “relevant” and “non-relevant” respectively. If the process is set to generate an imbalanced data stream then the filter does not discard any instances. However, if a balanced data stream is being generated then the filter selectively discards instances to maintain a balanced class distribution in the resultant data stream. For example, if the process is generating a data stream with 50 “relevant” and “non-relevant”

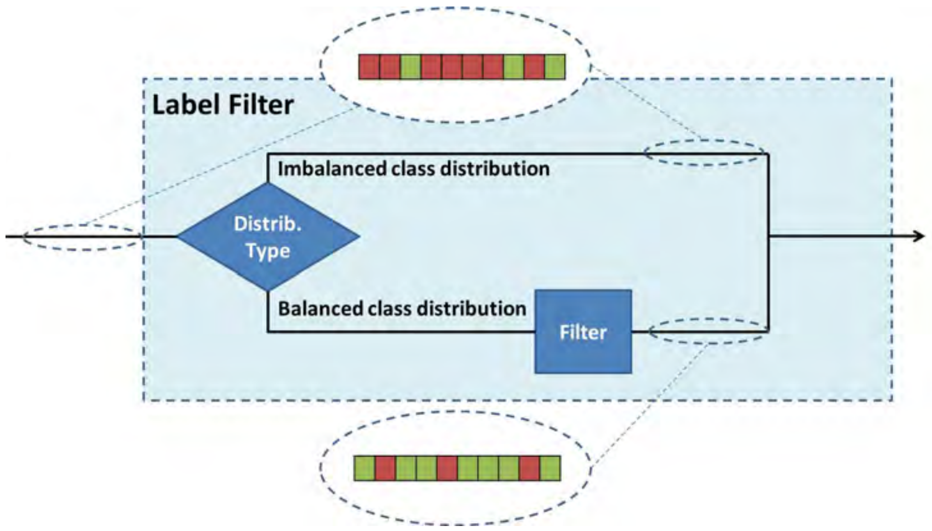


Figure 4.6: The class distribution filter.

instances per batch then the distribution filter will discard an instances if 50 instances of that class has already passed through the filter. Once 50 instances from each class has passed through the filter the filter is re-set and will allow instances through until the class limit is once again reached.

The algorithm for the full data generation processes is given in pseudo code in Algorithm 1.

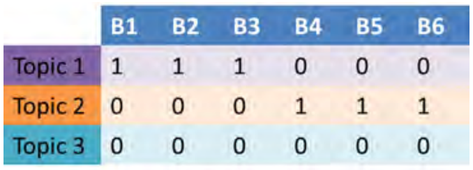
Most functions do not need further elaboration as their function can be inferred from their name, however, three functions need some further details:

- Head, accepts a collection of instances and returns the first instance from that collection.
- Tail, accepts a collection of instances and returns the same collection without the first instance.
- Floor, accepts a number and returns the integer value of that number, e.g $Floor(234/100) = 2$.

```
Function GenerateDataset(files, DistribFilter, topicFilters,
            batchSize)
    rawInstances ← ExtractInstances(files)
    fsInstances ← FeatureSelection(rawInstances)
    chronoOrdered ← SortChronologically(fsInstances)
    return ProcessStream(0, DistribFilter, topicFilters,
              batchSize, chronoOrdered, $\varnothing$)
end
Function ProcessStream(numPassedFilters, DistribFilter,
            topicFilters, batchSize, unfiltered, filtered)
    if unfiltered = $\varnothing$ then
        return filtered
    end
    candidate ← Head(unfiltered)
    currentBatchNo ← Floor(numPassedFilters / batchSize)
    TopicFilter ← $\text{topicFilters}_{\text{currentBatchNo}}$
    if TopicFilter(candidate) && DistribFilter(candidate) then
        return ProcessStream(numPassedFilters + 1,
                  DistribFilter, topicFilters, batchSize,
                  Tail(unfiltered), filtered $\cup$ candidate)
    end
    else
        return ProcessStream(numPassedFilters, DistribFilter,
                  topicFilters, batchSize, Tail(unfiltered), filtered)
    end
end
```

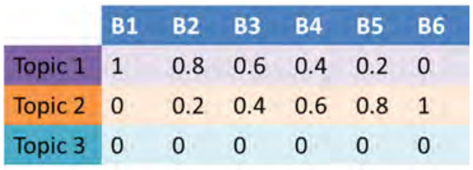
Algorithm 1: The concept drift dataset generation algorithm.

The key part to the algorithm is line 16 which parameterises the topic filter. This is a vital part of the algorithm as it is the varying of the topic filter which introduces controlled concept drift into a dataset. This line ensures that new parameters are applied to the topic filter every batchSize instances, where batchSize is a user-specified parameter specifying how many instances should be in each batch. The set of topic filter parameters are read from the *topic to probability matrix*.



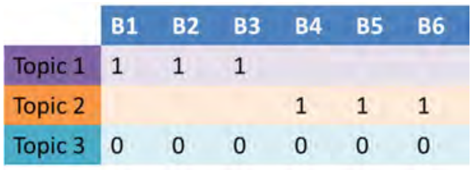
	B1	B2	B3	B4	B5	B6
Topic 1	1	1	1	0	0	0
Topic 2	0	0	0	1	1	1
Topic 3	0	0	0	0	0	0

(a) Conditional change with a sudden concept shift.



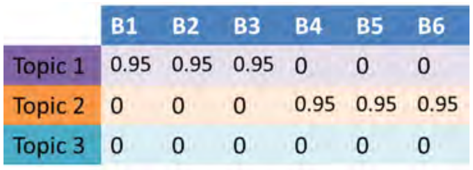
	B1	B2	B3	B4	B5	B6
Topic 1	1	0.8	0.6	0.4	0.2	0
Topic 2	0	0.2	0.4	0.6	0.8	1
Topic 3	0	0	0	0	0	0

(b) Conditional change with a gradual change in concept.



	B1	B2	B3	B4	B5	B6
Topic 1	1	1	1			
Topic 2				1	1	1
Topic 3	0	0	0	0	0	0

(c) Feature change with a sudden concept shift.



	B1	B2	B3	B4	B5	B6
Topic 1	0.95	0.95	0.95	0	0	0
Topic 2	0	0	0	0.95	0.95	0.95
Topic 3	0	0	0	0	0	0

(d) Conditional change with a sudden concept shift and 5% labelling noise.

Figure 4.7: Sample topic to probability matrices.

Figure 4.7 shows four sample topic probability matrices. Each column is one set of parameters to a topic filter, specifying the probability that an instance belonging to a given topic is given the true label of “relevant”. The topic filter parameters are updated every batch and one column in the topic to probability matrix gives the parameters for one batch. For example, Figure 4.7a specifies that instances belonging to “Topic 1” will receive the true label of “relevant” for the first three batches. This topic to probability matrix produces a conditional change dataset, with a concept shift at batch three, as “Topic 1” goes from being “relevant” to “non-relevant” while “Topic 2” goes from being “non-relevant” to “relevant”. A more gradual change in concept can be created by adjusting the probabilities over a number of batches, as in Figure 4.7b.

The matrix in Figure 4.7c has no probability specified for “Topic 2” for the first three batches. This means that instances belonging to that topic are completely filtered out while the topic filter parameters remain this way. Leaving entries in the topic to probability matrix empty allows for the creation of non-overlapping target topics which results in a feature change datasets.

Labelling noise can also be introduced by using probabilities which are less than one. For example, setting a probability of relevance to 0.95, as in Figure 4.7d, gives 5% labelling noise. This type of change is considered noise rather than gradual concept drift as it manifests itself as small fluctuations in classifier performance rather than a long term trend.

This approach to generating datasets is similar to the approaches used to generate conditional change datasets in (Klinkenberg, 1999, 2001, 2004; Klinkenberg & Renz, 1998) and feature change datasets in (Lanquillon, 1999). However, our approach differs in two important aspects. Firstly, the data in our framework is ordered chronologically, which maintains any natural concept drift present in the data. The author believes that datasets generated using this approach better capture the nuances of real concept drift problems. Secondly, it also allows the *natural* class distribution to be maintained (if so desired), whereas the approach used in (Klinkenberg, 1999, 2001, 2004;

Klinkenberg & Renz, 1998; Lanquillon, 1999) ensures that the class distribution remains fixed. The author expects that evaluating the approaches on data which reflects some of the issues found in real concept drift problems will lead to more informative evaluations. The full collection of drift induced datasets used in thesis can be downloaded at `http://arrow.dit.ie/sciendoc/147/`.

4.2 Experimental Set-up

The experiments described in this thesis aim to evaluate a collection of concept drift handling approaches on real and drift induced datasets. In each experiment the first 150 documents in the data stream of each class were selected as initial training data. The rest of the documents in the data stream were grouped into batches of 100 documents to be presented to the classifier for classification. The classifier makes a prediction for each instance in the batch and the classifier performance for the batch is recorded (using the evaluation measure discussed in Section 4.3). As this work is grounded in a text filtering domain a classifier which performs well on text data is needed. The classifier used in our approaches is an SVM as it has been shown to be especially suitable for text classification (Joachims, 1998; Yang & Liu, 1999).

The classifier is periodically rebuilt on more recent training data. The frequency of rebuilds depends the concept drift handling technique used. For continuous rebuild approaches the classifier is rebuilt every batch, whereas triggered rebuild approaches only cause the classifier to be rebuilt when a significant change in concept is suspected. Each time the classifier is rebuilt the number of new training instances that need to be labelled is recorded, as the amount of labelled data required is an important evaluation metric.

The framework used in this thesis is built around the Waikato Environment for Knowledge Analysis (WEKA) (Hall *et al.*, 2009) library. WEKA contains implementations of many of the most common machine learning algorithms and also contains implementations of many related tasks, such as data pre-processing. WEKA supports SVM using the LIBSVM (Chang & Lin, 2011) implementation through a wrapper class, WLSVM (EL-Manzalawy & Honavar, 2005).

4.3 Evaluation Measures

The classification accuracy over the whole data stream would normally be a good metric for determining how well an approach is handling concept drift. However the imbalanced datasets in Table 4.1 are likely to contain a high class imbalance, making the average class accuracy (Equation 2.3) a more appropriate evaluation metric. Average class accuracy tends toward the class accuracy if the class distribution is balanced but is not as susceptible to a high class imbalance.

$$avgClassAccuracy = \frac{\sum_{j=1}^{|Y|} accuracy_j}{|Y|} \tag{2.3 revisited}$$

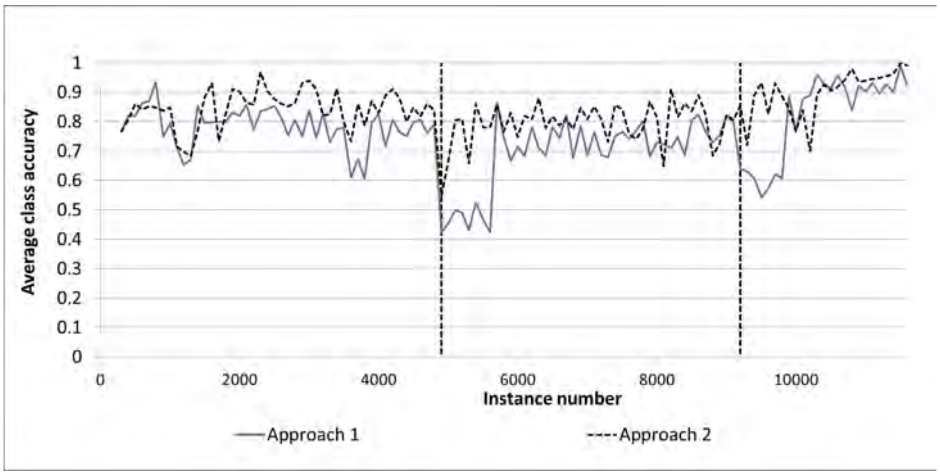
The second key evaluation measure for assessing our concept drift handling approaches is the number of labelled instances an approach uses. This is measured as the number of instances the approach requested a label for over the number of instances in the test stream. The largest contributor to this evaluation measure is the new training data required to rebuild a classifier but there are also approaches which uses labelled data to determine when the classifier should be rebuilt or to determine the size of a sliding window.

In addition to these quantitative measures graphs will also be provided to illustrate how the concept drift approaches work. One of the simplest ways of visualising the performance of a concept drift handling approach is to plot the classifier performance over time. The average class accuracy over the whole data stream is one number, which does not provide a high level of temporal information. Instead it needs to be on a batch level, to allow it to be plotted over time. This can be achieved by grouping the data in to fixed size batches and modifying Equation 2.3 into Equation 4.1.

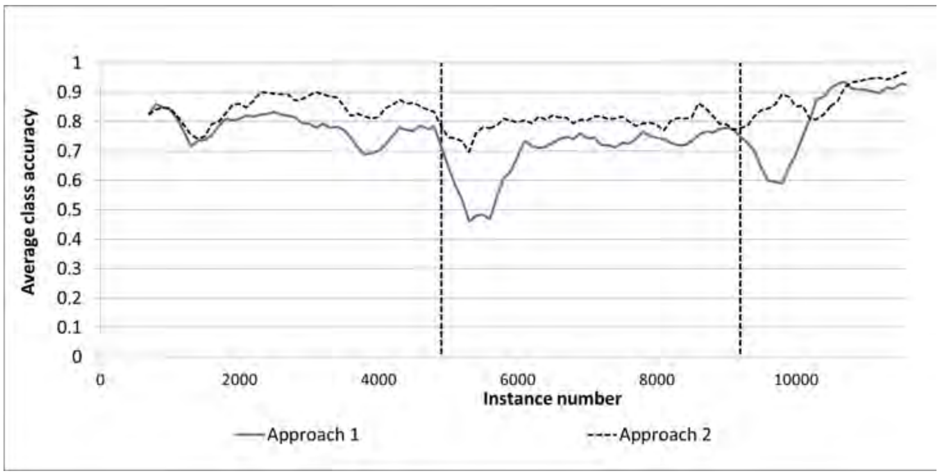
$$avgClassAccuracy_i = \frac{\sum_{j=1}^{|y_i|} accuracy_{ij}}{|y_i|} \tag{4.1}$$

where $accuracy_{ij}$ is the accuracy of class j on batch i. It is worth noting that in rare cases only examples of one class will be present in a batch and so $|y_i| = 1$.

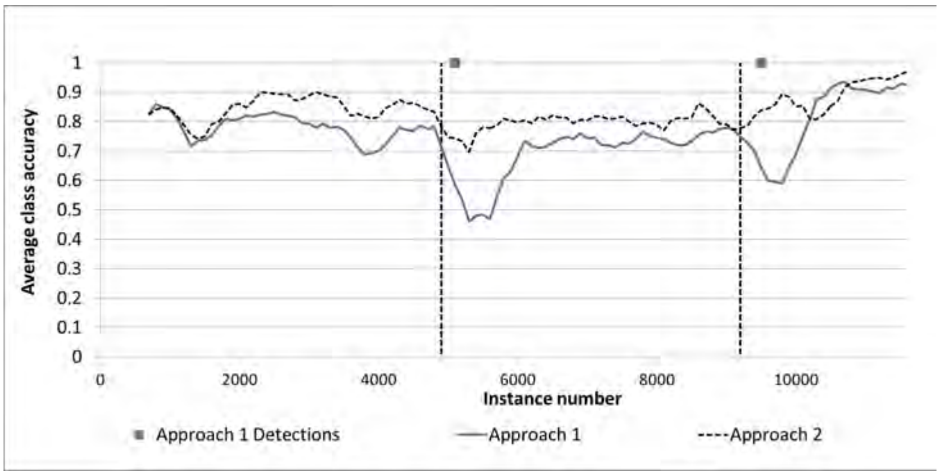
Figure 4.8a shows a typical average class accuracy over time graph for “Approach 1“ and “Approach 2”, two notional concept drift handling approaches. However, it can be hard to discern the long term trends in the data. An alternative approach to visualising the change over time is to plot the average of the last m batches, as in Figure 4.8b, where the average of



(a) Average class accuracy over time.



(b) Windowed average class accuracy over time.



(c) Windowed average class accuracy over time with detection points.

Figure 4.8: Different ways of displaying the average class accuracy over time.

the last five batches was used. This gives a smoothing effect which makes it easier to discern patterns and long term trends. For this reason average class accuracy over time graphs presented going forward will use a five batch smoothing unless otherwise specified.

The accuracy over time graphs can also be augmented with other useful information. This is mostly used for triggered approaches, as illustrated in Figure 4.8c, where a square is used to mark the points where a triggered rebuild approach detected a change in concept and rebuilt the classifier.

The accuracy over time graphs will be used on certain approaches to better illustrate their temporal behaviour, however, the two metrics which will be reported for each experiment are average class accuracy and fraction of labels used over the whole data stream. These two evaluation measures might make it hard to compare competing concept drift handling approaches. For example, it can be difficult to rank approaches if one approach obtains a high average class accuracy, but requires a large number of labels and a second approach obtains a lower average class accuracy yet uses less labelled data. This eventuality necessitates a performance metric which balances the relative importance of the two conflicting evaluation criteria, average class accuracy and the number of labels used. This will be the focus of the next section.

4.3.1 Multi-criteria Decision Analysis

Multi-criteria Decision Analysis (MCDA) allows the combination of various performance metrics (or criteria) into one number. MCDA has been successfully applied to classification problems (Peng *et al.*, 2011; Triantaphyllou & Baig, 2005), but has not, to the best of our knowledge been applied to the evaluation of cost sensitive concept drift handling. In MCDA criteria are a measure of the benefit or the cost of a solution. A benefit criterion is one where a large value is desirable and, inversely, a cost criterion is one where a large value is undesirable. An MCDA problem is framed as a matrix where the alternative solutions are rows and criteria are columns, as illustrated by the text classification problem in Table 4.2. In this problem multiple classifiers have classified the same text dataset and the classification accuracy and classification time (in ms) have been recorded. The ideal solution would have classification accuracy of one and a classification time of zero, however there is no such classifier in this example so the most suitable classifier is the one which strikes the best balance between classification accuracy and classification time.

Table 4.2: Classification example to illustrate a MCDA problem.

Classifier	**Accuracy**	**Classification Time**
SVM	0.81	193 ms
Single Prototype Classifier	0.78	924 ms
Decision Tree	0.69	9 ms
Ensemble	0.78	421 ms

The ranking of the solutions can be complicated by the fact that not all criteria are equally important to the task at hand. For example, it might be desirable to put less emphasis on the classification time if the classifier is run on a powerful server. MCDA approaches emphasise important criteria by giving them high weights, while less important criteria are given low weights.

The example in Table 4.2 only shows two criteria to keep the example simple, but MCDA can be applied to a problem with any number of criteria.

There are various MCDA approaches. This thesis uses the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) (Hwang & Yoon, 1981) approach. TOPSIS is well researched and intuitive and unlike conceptually similar techniques like the Pareto front, it provides a method of ranking approaches. TOPSIS is comprised of the following steps:

1. Normalise the MCDA Matrix by dividing each element in a criteria by the square root of the sum of squares of that criteria. This ensures that a criteria does not unduly influence the score due to its range.

2. Apply a weight to each criteria.

3. Find the notional positive and negative ideal solutions. The positive ideal solution is the largest value for each benefit criteria and lowest value for each cost criteria. The negative ideal solution is conversely the lowest value for each benefit criteria and largest value for each cost criteria.

4. Calculate the Euclidean distance between each solution and the positive and negative ideal solution.

5. The TOPSIS score of a solution reflects the relative closeness to the positive ideal solution and is calculated as:

$$\text{topsis}_i = \frac{dist(\text{solution}_i, neg)}{dist(\text{solution}_i, pos) + dist(\text{solution}_i, neg)} \tag{4.2}$$

where topsis_i is the TOPSIS score for solution i and $dist(\text{solution}_i, pos)$ and $dist(\text{solution}_i, neg)$ are the distances between solution i and the positive and negative ideal solution respectively.

6. The solutions can now be ranked by the TOPSIS score, with the solution with the largest TOPSIS score being considered the best solution.

The example in Table 4.2 can be visualised to further illustrate the TOPSIS process using a graph, like Figure 4.9.

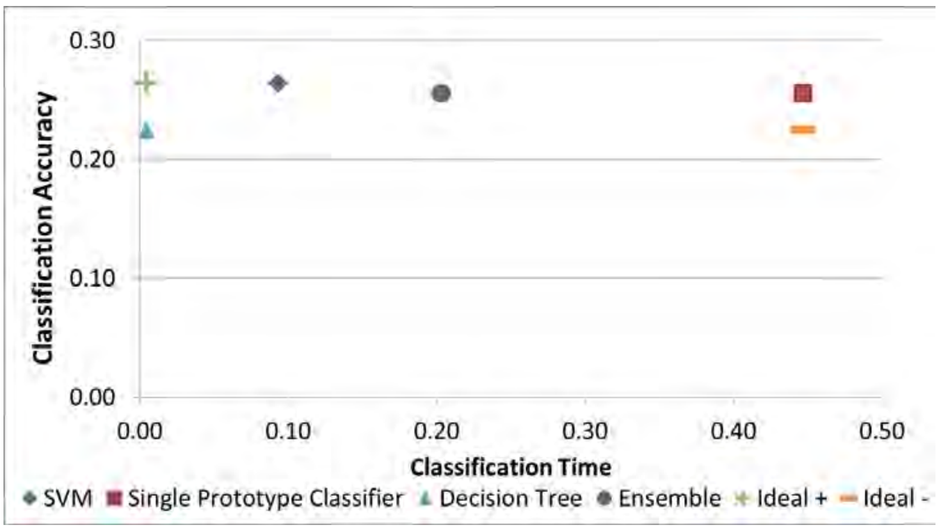


Figure 4.9: A visual representation of TOPSIS scores using equal weights.

This graph show the normalised version of the MCDA problem in Table 4.2 with the classification time and accuracy plotted on the x and y axis respectively. The positive and negative ideal solutions are also plotted. In this example both accuracy and classification time are given a weight of 0.50, but the weights could be any application appropriate values which sum up to one.

Applying the TOPSIS calculation to the data in Table 4.2 gives the TOPSIS scores and ranking listed in Table 4.3. The ranks show that when both

Table 4.3: TOPSIS scores and rank for a sample classification problem.

Classifier	**TOPSIS score**	**Rank**
SVM	0.80	2
Single Prototype Classifier	0.06	4
Decision Tree	0.92	1
Ensemble	0.55	3

criteria are considered using TOPSIS and the weights 0.50 for accuracy and 0.50 classification time, a decision tree is the best solution.

4.3.2 Statistical Evaluation Measures

TOPSIS can be used to combine average class accuracy and the number of labels used into one measure on which the different concept drift handling approaches can be ranked. The *sign test* can be used to determine whether the differences in TOPSIS scores between two approaches across multiple datasets is statistically significant or not (Demšar, 2006). The sign test examines the number of times one approach is better than another approach over multiple datasets, based on an evaluation metric such as classification accuracy or a TOPSIS score. For one approach to be considered statistically significantly better than the other it must win on most of the datasets. The exact number of wins required depends on the number of datasets used in the comparison and the desired confidence level. For example, if two approaches are compared over 15 datasets, then one approach has to win on 12 datasets for the difference to be statistically significant at the 95% confidence level.

The *Friedman test* (Friedman, 1940) is a more powerful, non-parametric, statistical test which is suitable for comparing multiple approaches over multiple datasets (Demšar, 2006). The Friedman test requires all of the approaches to be ranked from best to worst based on a metric (such as their TOPSIS score) on each dataset, after which it compares the average rank for all approaches for a statistically significant difference in average rank.

If a significant difference in results is found, however, the Friedman test does not indicate which approach is significantly different, for that a post-hoc test is needed. In this thesis the Nemenyi test (Nemenyi, 1963) is used as it is a well studied technique which is suitable to use in conjunction with the Friedman test (Demšar, 2006). The Nemenyi test uses the ranks to perform a pairwise comparison of the approaches to establish which pairs show a statistically significant difference.

4.4 Conclusion

This chapter has covered the methodology which will be used in the experiments described in the remainder of this thesis. It has also looked at how drift induced datasets exhibiting feature change and conditional change can be generated by adjusting the labelling process. This chapter has also covered the classification framework used and discussed the main evaluation metrics: average class accuracy and the fraction of labels used. The evaluation also covered how the data can be plotted to allow for a better understanding of the changes over time and how the two evaluation metrics can be combined into one measure using the MCDA technique TOPSIS. The last section dealt with the statistical tests that will be applied to TOPSIS scores to determine which concept handling approaches result in a statistically significant improvement over other approaches. The next chapter will apply the experiment methodology outlined in this chapter to one of our concept drift handling approaches, Decision Value Sampling (DVS) to establish if DVS can be used to handle concept drift while reducing the need for labelled data.

Chapter 5

Decision Value Sampling

Chapter 1 introduced two high level approaches to handling concept drift, continuous rebuild approaches and triggered rebuild approaches. However, most approaches in both categories make the strong assumption that the data will receive its true label shortly after classification, which can make them infeasible in some domains. A typical example of this is a sliding window approach which handles concept drift by continuously incorporating new labelled data into the training data, while discarding old training data. The size of the training window (the number of instances in the training set) determines how the sliding window approach handles concept drift and is normally the distinguishing difference between sliding window approaches (which are discussed in further detail in Section 3.3.1). Sliding window approaches share the need for a fully labelled data stream, as all instances in the data stream will be part of the training window at some stage and all instances in the training window need to be labelled. Further more, most sliding window approaches update the window size based on a labelled data dependant heuristic.

This limitation can be somewhat mitigated by altering the sliding window approach. Klinkenberg (1999) showed that the window size heuristic used in (Klinkenberg & Renz, 1998) can be applied to a data stream where a fixed number of random instances in the data stream were labelled. However, it is not clear if the authors only used the labelled random instances in the training window, or if all instances in the data stream were labelled and used in the training window. Either way, it provides a good starting point for further research. The approach operates on the assumption that the instances were labelled at random, which discards any information that might be obtained from the unlabelled instances. A natural next step is therefore semi-supervised learning, which leverages unlabelled data to improve the classifier. This chapter elaborates on our semi-supervised learning approach, Decision Value sampling (DVS), introduced in (Lindstrom *et al.*, 2010a).

DVS is a sliding window approach which reduces the labelled data requirement by using a *sampling strategy* to identify the most important instances in each batch and only requesting the label for those instances. These newly labelled instances are then added to the existing training window from which an equivalent number of the oldest instances are removed. The classifier is then rebuilt from the instances in the training window and the next batch of instances are presented for classification. This process repeats indefinitely.

The literature review in Chapter 3, details a small number of semi-supervised learning approaches aimed specifically at handling concept drift. Both (Zhu *et al.*, 2007) and (Masud *et al.*, 2008) use semi-supervised learning to improve the learning process. However, their approaches use an ensemble of classifiers, which make them computationally costly, a problem that can be further exasperated if applied to high dimensional data. They also require a large number of parameters, which can make them complicated to use.

The remainder of this chapter is organised as follows, Section 5.1 gives a high level overview of DVS and explains key components such as the sampling strategy and drift handling mechanism. Section 5.2 looks at the data and methodology used to evaluate DVS. Section 5.3 examines the results of the evaluation and Section 5.4 rounds off the chapter with a brief conclusion.

5.1 Overview

DVS aims to handle concept drift without requiring all the instances in the data stream being labelled. DVS is a continuous rebuild approach which means that the classifier is continuously adapted, in the case of DVS the classifier training data is augmented with a small number of labelled instances each batch. Figure 5.1 shows a high-level overview of how DVS works. After the initial classifier has been trained the instances arrive as a stream of unlabelled instances, are grouped into fixed size batches, and are then presented to the classifier for classification. DVS handles concept drift by continuously incorporating new training data, while discarding old training data, i.e. a sliding window approach. However, DVS only requires a

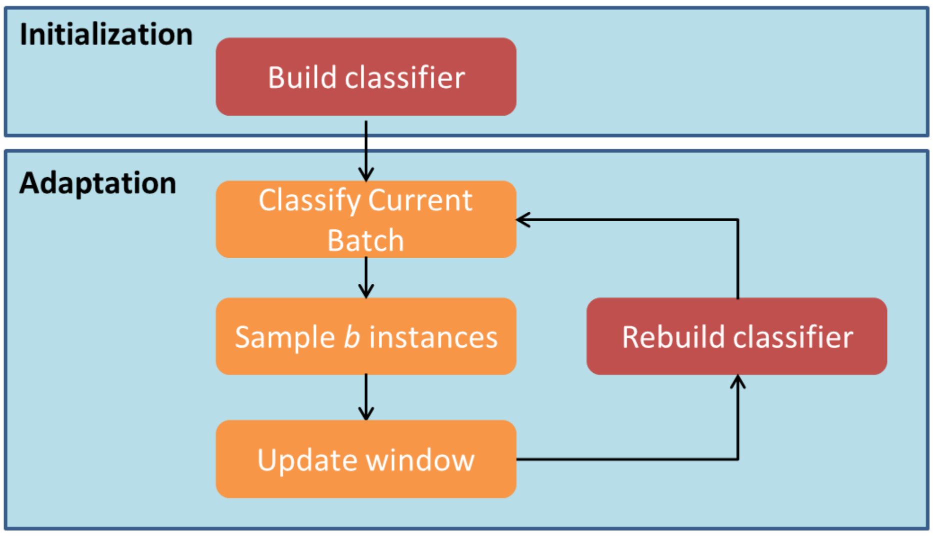


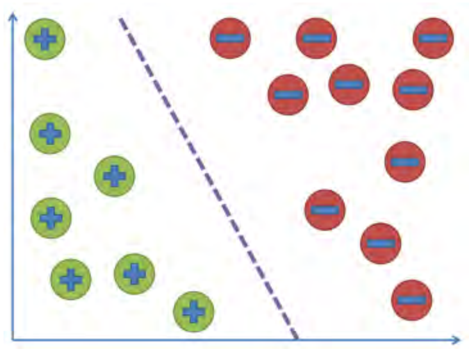
Figure 5.1: A overview of DVS.

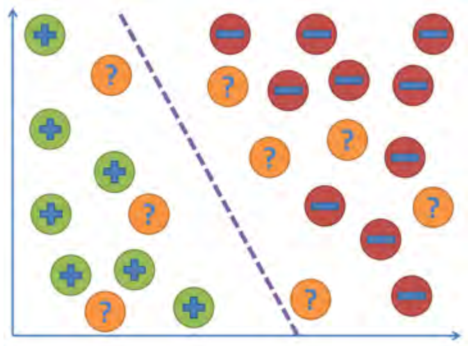
subset of the unlabelled instances in the batch to be labelled, unlike the sliding window approaches discussed in Section 3.3.1 which require all instances in the batch to be labelled. A sampling strategy is used to select a subset of the instances in the batch which are then labelled and added to the existing training window. Some of the oldest instances are removed from the training window and the classifier is rebuilt and ready to classify the next batch of data.

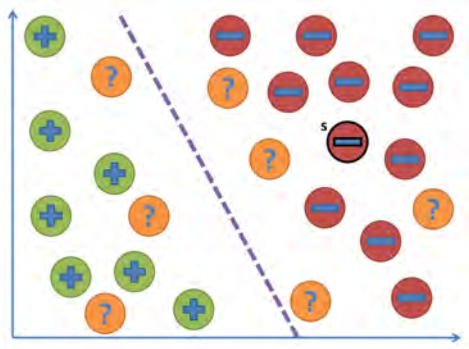
DVS is comprised of two major components which require further elaboration, the sampling strategy and the window update mechanism.

5.1.1 The Sampling Strategy

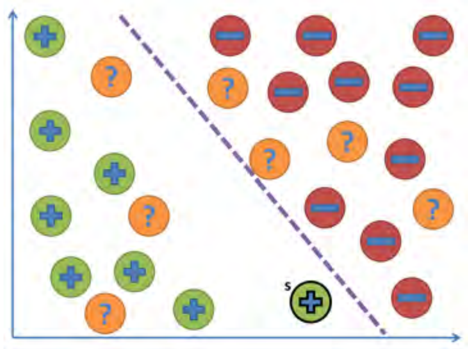
The sampling strategy is applied after a batch of unlabelled instances has been classified. The aim of the sampling strategy is to identify the unlabelled instances in the current batch which the classifier would benefit the most from having labelled and request the labels for those instances.


(a) Sampling scenario.


(b) Sampling scenario with unlabelled data.



(c) One possible result when using random sampling.



(d) The result when using DVS.

Figure 5.2: Comparing random sampling to DVS.

The sampling process can be illustrated using the 2D toy example in Figure 5.2. Figure 5.2a shows the step before the sampling process where positive and negative training examples are plotted in two dimensions. The positive instances are depicted as green circles with a plus sign, negative training instances as red circles with a minus sign and the dashed line is the decision boundary created by the classifier. The batch of unlabelled instances arrive, which are shown in Figure 5.2b as orange circles with a question mark. The unlabelled instances are classified by the classifier, after which the sampling process begins.

The number of instances to sample n, is controlled by the *labelling budget*

b, a parameter specifying what fraction of the data stream can be labelled. In DVS $n = Floor(b*\text{batchSize})$ (where the Floor function behaves the same way as defined for Algorithm 1), e.g if the batch size is 100 and the budget is 0.05 then the number of instances sampled in each batch is 5.

A surprisingly effective selection strategy, often used as a benchmark, is random sampling. Random sampling is a nondeterministic sampling approach where every instance is equally likely to be sampled. One possible case of random sampling from the scenario in Figure 5.2a can be seen in Figure 5.2c, where the instance s was sampled and labelled (as a negative instance). The addition of instance s to the training data does not change the decision boundary. For a batch based active learning approach, random sampling involves randomly sampling n unlabelled instances in each batch.

DVS uses a form of *uncertainty sampling*, a selection strategy which aims to sample the instances which the classifier is the most likely to misclassify (Lewis & Gale, 1994). It is expected that for a classifier with a discriminative decision boundary that the instances closest to the decision boundary are the most likely to be misclassified. Uncertainty sampling can be visualised as in Figure 5.2d where the instance closest to the decision boundary, instance s was sampled and labelled. The addition of s to the training data and subsequent re-training of the classifier moved the decision boundary from its original position in Figure 5.2a. The example demonstrates how uncertainty sampling can refine the decision boundary, whereas random sampling may select instances far from the decision boundary. A batched based uncertainty sampling approach samples the n instances closest to the decision boundary in each batch.

Uncertainty sampling can achieve high accuracy on a relatively small number of training instances as the instances selected for labelling are the instances the classifier is most likely to misclassify. However, the approach does need the decision boundary to be accurate and can be prone to sampling outliers (Roy & Mccallum, 2001).

Random sampling on the other hand may sample instances which are less important for finding a good decision boundary. This might therefore seem like a poor selection strategy, but it can be advantageous in certain circumstances. New concepts can only be handled if the selection strategy allows instances from the new concept to be sampled. If local concept drift occurs far from the decision boundary it might never be noticed by uncertainty sampling based approaches as they mostly sample from around the decision boundary. Random sampling on the other hand is just as likely to sample from areas experiencing localised changes in concept as any other area of feature space, and thus allowing the local concept drift to be handled.

5.1.2 Updating the Window

The next step in the adaptation phase is the updating of the training window. The addition of new training data allows a classifier to learn new concepts. A sampling strategy can be used to only label a small number of carefully chosen instances, rather than the whole data stream. However, just adding a few sampled instances to the training data each batch is unlikely to allow the classifier to handle concept drift in an efficient manner, as it can take a long time for a sufficient amount of data from the new concept to accumulate to overcome the affect of the training data from the outdated concept. Another problem with this approach is the increasing computational cost required to retrain the classifier as more instances are added to the training window. Some form of forgetting policy which allows old instances to be discarded is needed to overcome these problems. Section 3.3.1 discussed how a sliding window approach can be used to handle concept drift. The number of instances in the training window, or the window size is a key parameter which is either fixed or variable.

DVS uses a fixed size sliding window to handle concept drift. After classifying the instances in each batch the classifier is rebuilt using a training window augmented with the instances sampled by the selection strategy and then labelled. Concept drift handling approaches run on real data are very likely to encounter an imbalanced class distribution. Simply replacing the oldest n training instances with n new ones would result in training windows becoming heavily skewed towards the majority class over time. Instead, the instances removed from the training window at each iteration are the oldest instances of the same class as the n new instances. As a result the class distribution of the training window is kept constant, making DVS a fixed size, and fixed class distribution, sliding window approach.

5.2 Evaluation

The goal of this evaluation is to determine if DVS can be used to handle concept drift, and to understand the relationship between the labelling budget b and average class accuracy. The evaluation metrics used will therefore be the average class accuracy over the whole data stream, and b, the fraction of instances labelled. DVS aims to minimise the need for labelled instances, so b should be as small as possible while maintaining high classification accuracy. It would seem logical to assume that as b approaches one (all the instances in the batch are labelled and added to the training window, i.e. a fully labelled data stream), the accuracy increases.

The value of b might also affect the average class accuracy in other ways. A classifier will only be able to classify instances of an unfamiliar concept when there are a sufficient number of labelled instances of that concept in the training window. If b is large then a large number of instances from the new concept will be sampled, labelled and added to the training window. Consequently, as b approaches zero the number of instances of the new concept added to the training data is small and the time it takes for the classifier to "recover" to its previous accuracy might increase. So a secondary area of interest is evaluating the influence of b on how quickly the algorithm recovers after a change in concept. Intuition would suggest that a large value for b would allow the approach to recover quicker, as a larger proportion of the data in the training window would be from the new concept. Another area of interest is to establish how DVS compares to random sampling.

These goals can be accomplished by varying the labelling budget and sampling strategy as outlined in Table 5.1.

Table 5.1: DVS evaluation parameters.

b	**Sampling Strategy**
0.00	Random sampling
0.05	Random sampling
0.10	Random sampling
0.15	Random sampling
0.25	Random sampling
0.50	Random sampling
0.75	Random sampling
1.00	Random sampling
0.00	DVS
0.05	DVS
0.10	DVS
0.15	DVS
0.25	DVS
0.50	DVS
0.75	DVS
1.00	DVS

The combination of a particular sampling strategy and labelling budget will be abbreviated by concatenating the sampling strategy and budget. For example, DVS-0.05 means DVS using a 0.05 labelling budget while RS-0.75 refers to random sampling using a 0.75 budget.

5.2.1 Datasets

DVS will be evaluated on both real data, in the form of the two spam datasets detailed in Section 4.1.3.1, and drift induced datasets, generated using the methodology from Section 4.1.3.2. DVS should be able to handle: (1) both conditional and feature change as it is continuously updated with new labelled data, (2) both balanced and imbalanced datasets due to the update mechanism used, which ensures that the training window maintains a balanced class distribution, (3) both gradual and sudden concept drift. DVS will therefore be evaluated on dataset which exhibit a combination of the properties listed above.

The specifics of the datasets are given in Tables 5.2 to 5.4. The name column shows what concepts were created for the dataset, while the other four columns list the target topic, size and class distribution for each concept. The datasets were given their true labels as outlined in Section 4.1.2, i.e concept drift is induced by changing the target topic over time. The topics trade and religion are composite topics comprised of related topics. For example, the composite topic religion is comprised of the newsgroups "alt.atheism", "talk.religion.misc" and "soc.religion.christian". The composite topics were created as the single topics were not large enough. The next section will present the methodology used to evaluate DVS on the above listed datasets.

5.2.2 Methodology

In each experiment the first 150 documents of each class in the data stream were selected as initial training data. The rest of the documents in the data stream were grouped into batches of 100 documents to be presented to the classifier for classification and the average class accuracy achieved for each batch was recorded. An SVM is used as the approach classifier as it is well suited for text classification. Existing work in active learning has suggested that a good selection strategy for an SVM is to choose instances close to the separating hyperplane (Tong & Koller, 2002; Xu *et al.*, 2003). These

Table 5.2: The topic and class distributions of the 20 newsgroups datasets.

Dataset	**Name**	**Target topic**	**Size**	**No. Rel.**	**No. Non-rel**
20NG$_{SBF}$	Training	comp.*	300	150	150
	C1	comp.*	2000	1000	1000
	C2	religion	2100	1050	1050
	C3	sci.*	1800	900	900
20NG$_{GBF}$	Training	comp.*	300	150	150
	C1	comp.*	1800	900	900
	C1/C2	N/A	400	200	200
	C2	religion	1700	850	850
	C2/C3	N/A	400	200	200
	C3	sci.*	1800	900	900
20NG$_{SIF}$	Training	comp.*	300	150	150
	C1	comp.*	4600	1157	3443
	C2	religion	4300	1259	3041
	C3	sci.*	2500	1468	1032
20NG$_{GIF}$	Training	comp.*	300	150	150
	C1	comp.*	4400	1115	3285
	C1/C2	N/A	400	58	342
	C2	religion	3900	1151	2749
	C2/C3	N/A	400	86	314
	C3	sci.*	2500	1508	992
20NG$_{SBC}$	Training	comp.*	300	150	150
	C1	comp.*	2400	1200	1200
	C2	religion	2400	1200	1200
	C3	sci.*	2500	1250	1250
20NG$_{GBC}$	Training	comp.*	300	150	150
	C1	comp.*	2000	1000	1000
	C1/C2	N/A	400	200	200
	C2	religion	2000	1000	1000
	C2/C3	N/A	400	200	200
	C3	sci.*	2300	1150	1150
20NG$_{SIC}$	Training	comp.*	300	150	150
	C1	comp.*	6600	997	5603
	C2	religion	6300	1246	5054
	C3	sci.*	6200	1812	4388
20NG$_{GIC}$	Training	comp.*	300	150	150
	C1	comp.*	6400	957	5443
	C1/C2	N/A	400	102	298
	C2	religion	5900	1134	4766
	C2/C3	N/A	400	107	293
	C3	sci.*	6000	1768	4232

Table 5.3: The topic and class distributions of the Reuters datasets.

Dataset	**Name**	**Target topic**	**Size**	**No. Rel.**	**No. Non-rel**
Reuters$_{SBF}$	Training	earn	300	150	150
	C1	earn	1300	650	650
	C2	trade	1300	650	650
	C3	acq	1100	550	550
Reuters$_{GBF}$	Training	earn	300	150	150
	C1	earn	1100	550	550
	C1/C2	N/A	400	200	200
	C2	trade	900	450	450
	C2/C3	N/A	400	200	200
	C3	acq	1000	500	500
Reuters$_{SIF}$	Training	earn	300	150	150
	C1	earn	5200	1476	3724
	C2	trade	5200	402	4798
	C3	acq	4200	641	3559
Reuters$_{GIF}$	Training	earn	300	150	150
	C1	earn	5000	1418	3582
	C1/C2	N/A	400	57	343
	C2	trade	4800	366	4434
	C2/C3	N/A	400	43	357
	C3	acq	4100	627	3473
Reuters$_{SBC}$	Training	earn	300	150	150
	C1	earn	1500	750	750
	C2	trade	1500	750	750
	C3	acq	600	300	300
Reuters$_{GBC}$	Training	earn	300	150	150
	C1	earn	1400	700	700
	C1/C2	N/A	400	200	200
	C2	trade	1000	500	500
	C2/C3	N/A	400	200	200
	C3	acq	600	300	300
Reuters$_{SIC}$	Training	earn	300	150	150
	C1	earn	7200	1666	5534
	C2	trade	6200	328	5872
	C3	acq	5900	690	5210
Reuters$_{GIC}$	Training	earn	300	150	150
	C1	earn	7000	1640	5360
	C1/C2	N/A	400	41	359
	C2	trade	5800	307	5493
	C2/C3	N/A	400	30	370
	C3	acq	5700	674	5026

Table 5.4: The topic and class distributions of the news sources and spam datasets.

Dataset	Name	Target topic	Size	No. Rel.	No. Non-rel
NS$_{SBF}$	Training	Business	300	150	150
	C1	Business	7900	3950	3950
	C2	Sport	8000	4000	4000
	C3	Entertainment	5700	2850	2850
NS$_{SIF}$	Training	Business	300	150	150
	C1	Business	14900	3712	11188
	C2	Sport	15000	4867	10133
	C3	Entertainment	15000	2350	12650
Spam$_1$	Training	N/A	300	150	150
	Testing	N/A	9900	1036	8864
Spam$_2$	Training	N/A	300	150	150
	Testing	N/A	8200	688	7512

experiments use the SVM implementation LIBSVM, which means that the instances with low decision values are the ones close to the hyperplane, which is where decision value sampling gets it name. However, the general approach of using an active learning sampling strategy and a sliding window to handle concept drift can be applied to any classifier which can provide confidence scores.

After classifying the instances in the batch the selection strategy is used to sample n instances which are then labelled. The sampled instances are added to the training window and the window is updated as discussed in Section 5.1.2. It is desirable to keep the number of algorithm parameters to a minimum, so the size and class distribution of the window is set to the same size and class distribution as the initial training window (150 instances of each class). An interesting direction for future research would be to use a variable size sliding window, or a less rigid class distribution.

As random sampling is nondeterministic the random sampling experiments will be run 10 times for each value of b, and the average of the class accuracies will be used.

5.3 Results

Table 5.5 shows the relationship between the parameter b and the average class accuracy over the whole stream when random sampling was used as the sampling strategy. Each row gives the average class accuracy over the whole data stream for that value of b for a given dataset.

Table 5.5: The average class accuracy obtained by random sampling for various labelling budgets.

	0.00	**0.05**	**0.10**	**0.15**	**0.25**	**0.50**	**0.75**	**1.00**
20NG$_{SBF}$	0.68	0.74	0.75	0.76	0.78	0.80	0.81	0.81
20NG$_{GBF}$	0.67	0.72	0.72	0.73	0.76	0.78	0.79	0.80
20NG$_{SIF}$	0.61	0.73	0.78	0.79	0.81	0.83	0.83	0.84
20NG$_{GIF}$	0.60	0.73	0.77	0.79	0.81	0.82	0.83	0.83
20NG$_{SBC}$	0.49	0.58	0.66	0.69	0.73	0.77	0.78	0.78
20NG$_{GBC}$	0.48	0.58	0.65	0.69	0.73	0.77	0.78	0.78
20NG$_{SIC}$	0.54	0.66	0.72	0.75	0.78	0.81	0.81	0.81
20NG$_{GIC}$	0.54	0.66	0.71	0.75	0.78	0.81	0.81	0.81
Reuters$_{SBF}$	0.63	0.71	0.74	0.77	0.80	0.83	0.83	0.84
Reuters$_{GBF}$	0.63	0.70	0.73	0.75	0.79	0.82	0.83	0.83
Reuters$_{SIF}$	0.63	0.71	0.75	0.77	0.81	0.84	0.85	0.86
Reuters$_{GIF}$	0.63	0.70	0.74	0.76	0.79	0.83	0.84	0.85
Reuters$_{SBC}$	0.62	0.67	0.70	0.72	0.77	0.82	0.83	0.84
Reuters$_{GBC}$	0.62	0.67	0.70	0.73	0.77	0.82	0.83	0.84
Reuters$_{SIC}$	0.58	0.66	0.70	0.72	0.77	0.81	0.83	0.84
Reuters$_{GIC}$	0.58	0.67	0.71	0.73	0.77	0.82	0.84	0.85
NS$_{SBF}$	0.60	0.83	0.84	0.85	0.86	0.86	0.86	0.85
NS$_{SIF}$	0.60	0.80	0.84	0.84	0.85	0.85	0.85	0.84
Spam$_1$	0.81	0.96	0.96	0.96	0.96	0.97	0.97	0.97
Spam$_2$	0.72	0.87	0.89	0.90	0.91	0.93	0.94	0.94

The results show that not updating the classifier at all ($b = 0$) leads to a low average class accuracy, whereas rebuilding the classifier with even a small number of new labelled instances, such as $b = 0.05$, improves the accuracy of the classifier on most datasets. The results also suggest that the approach is "front loaded", as the biggest gains come in the beginning, while increasing the budget above 0.15 gives diminishing returns.

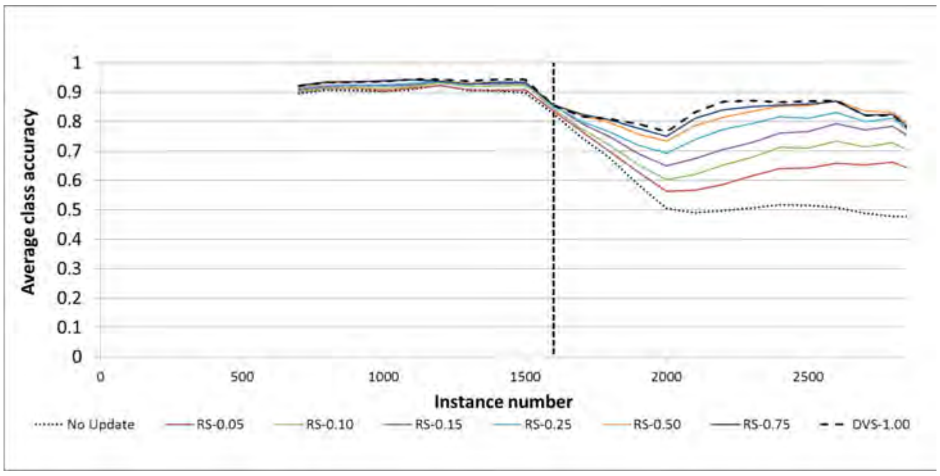


Figure 5.3: The influence of b on the average class accuracy over time using random sampling on the Reuters$_{SBF}$ dataset.

A further insight to the importance of b can be gained by plotting the average class accuracy over time. For brevity a graph for each dataset will not be included, instead a representative graph will be used.

Figure 5.3 show the average class accuracy over time of DVS using the random sampling selection strategy with different labelling budgets. The experiment was run on the first two concepts (C1 and C2) on the Reuters$_{SBF}$ dataset, using the result smoothing approach discussed in Section 4.3. The dashed vertical lines marks the concept change point.

It is clear from Figure 5.3 that when no concept drift handling is performed the classifier performance deteriorates once the sudden change in concept occurs, and the non-updating classifier never seems to recover.

Conversely, when all examples in each batch are labelled and used to update the classifier ($b = 1$, which effectively gives a standard sliding window approach) the performance of the classifier dips after the change in concept, but recovers shortly afterwards. The classification accuracy remains high until towards the end of the dataset, where a small decline occurs, likely due to natural concept drift in the data.

This trend is consistent across all the datasets, as the labelling budget is raised the average class accuracy over the data stream increases, just like in Figure 5.3. The difference in classifier accuracy over a range of labelling budgets can be largely attributed to the recovery time of the approach, which is heavily related to the value of b. The larger the value of b, the faster the classifier accuracy picks back up after a change in concept.

Table 5.6 shows the average class accuracy obtained on the same datasets when DVS was used.

The DVS results follow a very similar pattern to the results in Table 5.5. Even adding a small number of labelled instances to the training data each batch improves the classification accuracy on all datasets. Another similarity with the random sampling results is that most of the improvement occurs for smaller values of b.

The observations from Figure 5.3 apply equally to Figure 5.4, which shows the average class accuracy over time when DVS was used. A comparison between the random sampling graph and DVS graph seems to suggest that DVS starts to adapt slightly slower than random sampling for low values of b, but then outperforms random sampling. This probably occurs because it takes a few batches of training data from the new concept before DVS finds a

Table 5.6: The average class accuracy obtained by DVS for various labelling budgets.

	0.00	**0.05**	**0.10**	**0.15**	**0.25**	**0.50**	**0.75**	**1.00**
20NG$_{SBF}$	0.68	0.72	0.77	0.76	0.79	0.82	0.82	0.81
20NG$_{GBF}$	0.67	0.69	0.73	0.76	0.78	0.79	0.80	0.80
20NG$_{SIF}$	0.61	0.77	0.81	0.82	0.83	0.84	0.85	0.84
20NG$_{GIF}$	0.60	0.77	0.80	0.82	0.82	0.84	0.84	0.83
20NG$_{SBC}$	0.49	0.58	0.67	0.71	0.74	0.77	0.78	0.78
20NG$_{GBC}$	0.48	0.56	0.65	0.71	0.74	0.78	0.78	0.78
20NG$_{SIC}$	0.54	0.66	0.73	0.77	0.79	0.82	0.82	0.81
20NG$_{GIC}$	0.54	0.66	0.73	0.76	0.79	0.82	0.82	0.81
Reuters$_{SBF}$	0.63	0.71	0.73	0.76	0.80	0.83	0.84	0.84
Reuters$_{GBF}$	0.63	0.68	0.71	0.76	0.79	0.84	0.84	0.83
Reuters$_{SIF}$	0.63	0.72	0.78	0.80	0.84	0.86	0.86	0.86
Reuters$_{GIF}$	0.63	0.71	0.77	0.80	0.83	0.85	0.86	0.84
Reuters$_{SBC}$	0.62	0.66	0.68	0.72	0.78	0.82	0.83	0.84
Reuters$_{GBC}$	0.62	0.67	0.69	0.71	0.78	0.84	0.84	0.84
Reuters$_{SIC}$	0.58	0.67	0.73	0.76	0.79	0.83	0.84	0.84
Reuters$_{GIC}$	0.58	0.67	0.74	0.77	0.80	0.83	0.85	0.85
NS$_{SBF}$	0.60	0.83	0.86	0.86	0.87	0.87	0.87	0.85
NS$_{SIF}$	0.60	0.84	0.87	0.87	0.87	0.87	0.86	0.84
Spam$_1$	0.81	0.97	0.97	0.97	0.97	0.97	0.98	0.97
Spam$_2$	0.72	0.93	0.93	0.94	0.94	0.93	0.93	0.94

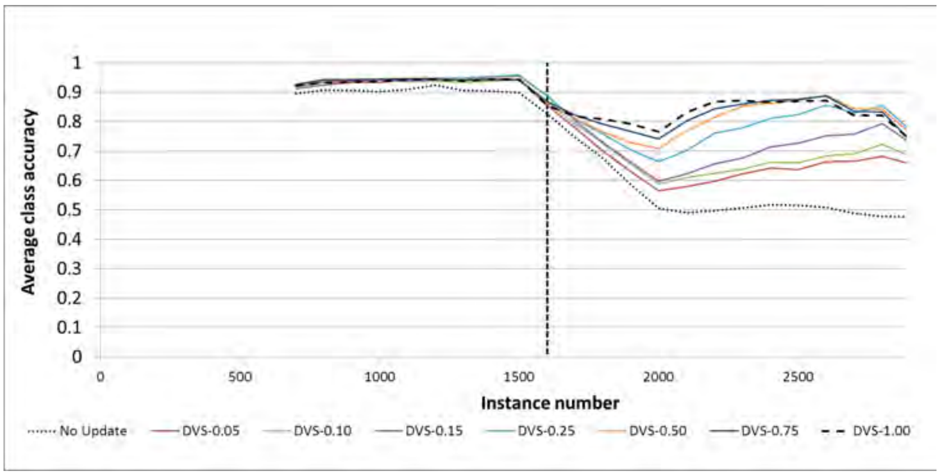


Figure 5.4: The influence of b on the average class accuracy over time using DVS on the Reuters$_{SBF}$ dataset.

stable decision boundary to sample around. Once a good decision boundary is found DVS seems to outperform random sampling for most values of b.

Comparing the two results tables shows a small but consistent difference between random sampling and DVS, with DVS giving a slightly higher average class accuracy for most values of b. Tabulating the wins, losses and draws, as in Table 5.7 allows for an easier comparisons between the two sampling techniques.

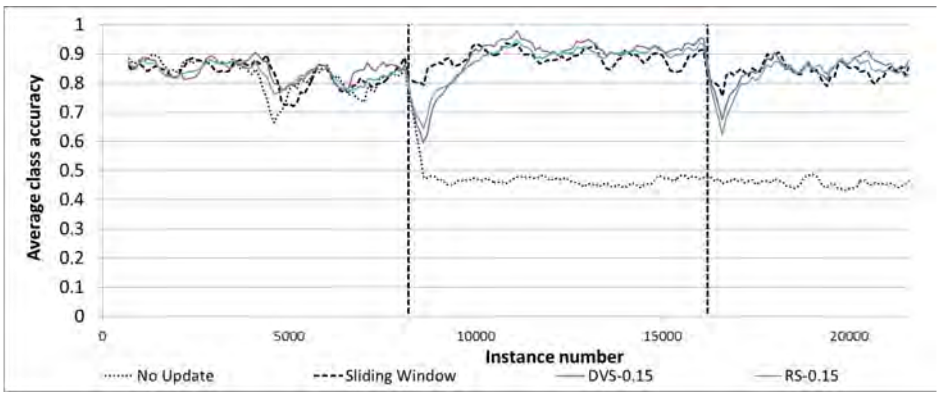
Table 5.7: The wins, losses and draws obtained by each sampling technique.

	0.05	**0.10**	**0.15**	**0.25**	**0.50**	**0.75**	**1.00**
20NG$_{SBF}$	RS	DVS	DVS	DVS	DVS	DVS	DRAW
20NG$_{GBF}$	RS	DVS	DVS	DVS	DVS	DVS	DRAW
20NG$_{SIF}$	DVS	DVS	DVS	DVS	DVS	DVS	DVS
20NG$_{GIF}$	DVS	DVS	DVS	DVS	DVS	DVS	DVS
20NG$_{SBC}$	RS	DVS	DVS	DVS	DVS	DVS	DRAW
20NG$_{GBC}$	RS	DVS	DVS	DVS	DVS	DVS	RS
20NG$_{SIC}$	RS	DVS	DVS	DVS	DVS	DVS	RS
20NG$_{GIC}$	RS	DVS	DVS	DVS	DVS	DVS	RS
Reuters$_{SBF}$	DVS	RS	RS	RS	DVS	DVS	DRAW
Reuters$_{GBF}$	RS	RS	DVS	DVS	DVS	DVS	DVS
Reuters$_{SIF}$	DVS	DVS	DVS	DVS	DVS	DVS	RS
Reuters$_{GIF}$	DVS	DVS	DVS	DVS	DVS	DVS	RS
Reuters$_{SBC}$	RS	RS	DVS	DVS	DVS	DVS	DRAW
Reuters$_{GBC}$	DVS	RS	RS	DVS	DVS	DVS	RS
Reuters$_{SIC}$	DVS	DVS	DVS	DVS	DVS	DVS	RS
Reuters$_{GIC}$	RS	DVS	DVS	DVS	DVS	DVS	RS
NS$_{SBF}$	RS	DVS	DVS	DVS	DVS	DVS	DRAW
NS$_{SIF}$	DVS	DVS	DVS	DVS	DVS	DVS	RS
Spam$_1$	DVS	DVS	DVS	DVS	DVS	DVS	RS
Spam$_2$	DVS	DVS	DVS	DVS	RS	RS	DVS
No. wins DVS	10	**16**	**18**	**19**	**19**	**19**	4
No. wins RS	10	4	2	1	1	1	10
No. draws	0	0	0	0	0	0	6

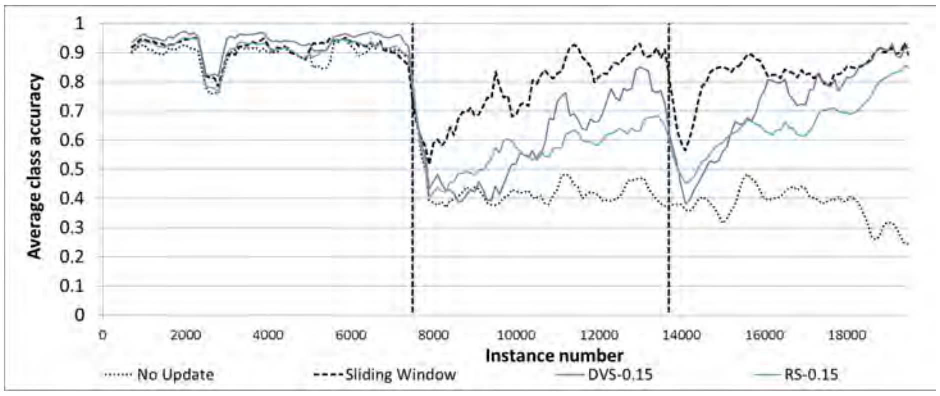
A sampling technique is considered to win on a particular dataset if it obtains a higher average class accuracy than the other sampling technique using the same labelling budget. For example, Table 5.7 shows that on the 20NG$_{SBF}$ dataset RS-0.05 was better than DVS-0.05, DVS-1.00 and RS-1.00 was a draw and DVS won the other tested labelling budgets. The last three rows give the total number of wins, losses and draws for each approach. According to the sign test, one approach is statistically significantly better than the other if it wins 15 out of 20 times (at the 95 % confidence level). DVS is therefore statistically significantly better than random sampling when using the a labelling budget of 0.10, 0.15, 0.25, 0.50 or 0.75.

The analysis of the difference between random sampling and DVS can be further helped by picking a particular value for b and plotting the average class accuracy over time. There are too many datasets and labelling budgets to present all possible combinations as accuracy over time graphs. Instead a selection of representative graphs will be relied upon to illustrate some differences between DVS and random sampling. The results for $b = 0.15$ give a good balance between the labelling effort and the performance achieved and are fairly representative of the graphs for other datasets. Appendix C.1 contains average class accuracy over time graphs on more datasets which have been excluded from this section for brevity. Figure 5.5 shows the average class accuracy over time with $b = 0.15$ on three datasets.

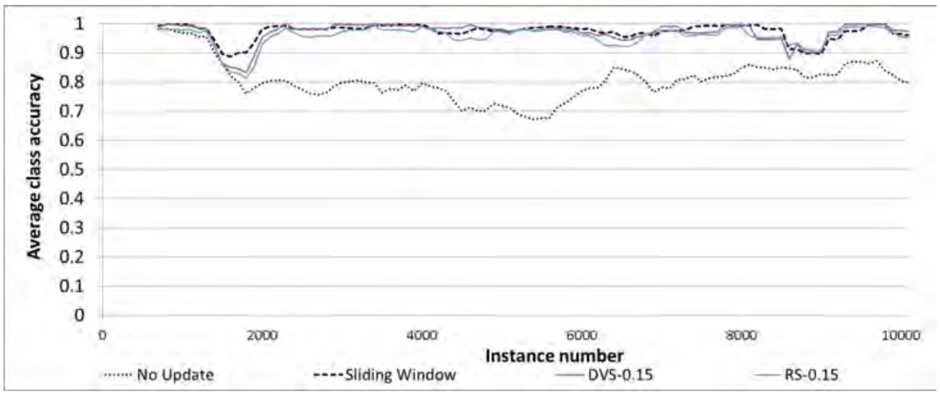
On each graph no update refers to a scenario in which no concept drift handling is used ($b = 0$), while sliding window refers to a fixed size sliding window approach that uses true labels for all examples in each batch (i.e. $b = 1.00$). DVS-0.15 and RS-0.15 refer to the scenarios where decision value and random sampling selection strategies with a 0.15 labelling budget, respectively, are used. The concept shifts are denoted using a dashed vertical line (except in Figure 5.5c where there concept shift and drift points are unknown).



(a) Average class accuracy over time on the NS$_{SBF}$ dataset



(b) Average class accuracy over time on the Reuters$_{SIC}$ dataset



(c) Average class accuracy over time on the Spam$_1$ dataset

Figure 5.5: Showing the affect of different sampling techniques and labelling budgets on the average class accuracy over time.

In general both the RS-0.15 and DVS-0.15 approaches to concept drift handling perform well. Figure 5.5a is illustrative of how both DVS and random sampling handle concept drift. Before the first change in concept both DVS-0.15 and RS-0.15 maintain an average class accuracy very comparable to, and sometimes exceeding, the sliding window approach. Particularly DVS-0.15 occasionally outperforms the sliding window approach (which uses all the labelled data) when the concept is stable, as can be seen both before the first concept shift and before the second concept shift in Figure 5.5a. This is an interesting result and probably arises because there is some noise in the data stream and DVS avoids the noisy instances to produce a better classifier. The average class accuracy for all three updating approaches declines when a change in concept occurs. The sliding window approach seems to recover the fastest, which is consistent with Figures 5.3 and 5.4, which showed that the larger the labelling budget the faster the average class accuracy recovers. DVS-0.15 and RS-0.15 take a bit longer to recover, but they both catch back up with the sliding window approach over time.

Figure 5.5b shows the same experiment on the Reuters$_{SIC}$ dataset. The pattern before the first change in concept is very similar to the pattern in the previous graph. It seems like RS-0.15 recovers faster after the change in concept than DVS-0.15. This is probably related to the previously mentioned point of how it might take DVS-0.15 a few batches of labelled data to find a good decision boundary to sample around. This graph is also interesting as the difference between the three approaches is more pronounced. It seems like a larger labelling budget is required to achieve an average class accuracy comparable to the sliding window as the classification task is harder. This reinforces the notion that the appropriate labelling budget is highly domain dependant.

Figure 5.5c shows how RS-0.15 and DVS-0.15 managed to handle concept drift on the Spam$_1$ dataset. This graph was included to ensure that these approaches can manage concept drift on a non-artificial dataset. The average class accuracy over time shows that the change in concept on the Spam$_1$ dataset is gradual rather than sudden. It should also be noted that there is little difference between the three approaches, with both DVS-0.15 and RS-0.15 achieving an average class accuracy which is very comparable to the one obtained by the sliding window approach. This is probably due to the fact that the two concepts are easily distinguishable. In fact, the results in Tables 5.5 and 5.6 show that both approaches can obtain a very high average class accuracy with a labelling budget of only 0.05. The number, magnitude and recurrence of changes in concept can provide an alternative, or perhaps complimentary explanation for the difference between the spam datasets and the drift induced datasets. However, this is hard to confirm as the properties of the drift in the spam datasets, such as the drift type and drift cause and are unknown.

5.4 Conclusion

The goal of this work is to reduce the need for labelled instances when handling changing concepts using a continuous rebuild approach. This was achieved by combining active learning and a sliding window. The need for labelled instances is reduced by using active learning to selectively sample the most useful instances for labelling each time the classifier is to be retrained. Experiments were performed on multiple text datasets from commonly used text corpora in which drift was induced, and on two real-world spam filtering datasets. On all of these datasets it was possible to maintain classification accuracies comparable with those achieved using full labelling of the data stream by labelling only 15% of the incoming instances - a significant reduction in the labelling effort required. However, it should be noted that there is an inverse relationship between the amount of labelled data used and the recovery time of DVS, i.e. as the amount of labelled data used decreases, the recovery time increases. The results also showed that decision value selection strategy boosted performance more than the random sampling selection strategy emphasising the usefulness of targeted selection of instances for labelling. The next chapter will cover our triggered rebuild approach, Confidence Distribution Batch Detection (CDBD).

Chapter 6

Confidence Distribution Batch Detection

The results from Chapter 5 suggest that the first high level approach to deal with concept drift, semi-supervised learning, manages to handle concept drift while reducing the need for labelled data. Our experiments show that the semi-supervised learning approach Decision Value Sampling (DVS) handled concept drift effectively while only requiring 15% of the data stream to be labelled. However, there is a significant drawback to DVS, and the other semi-supervised approaches to handling concept drift examined, in that they require labelled data regardless of whether the concept is changing or not. For example, imagine a data stream which experiences infrequent sudden concept shifts, yet is stable in between these shifts. In this type of scenario the semi-supervised approaches incur an unnecessary labelling cost as they require a fixed amount of labelled data (such as 15% for DVS), even when the concept is not changing. In this scenario the second high level approach to concept drift handling, triggered rebuilds, might be more appropriate.

Triggered rebuild approaches monitor a variable believed to be correlated with a change in concept, an *indicator*, and rebuild the classifier when the indicator value changes significantly. A common way to form an indicator is using the two-window paradigm (discussed in more detail in Section 3.4.3.2) where some summary information from a reference window is compared to the summary information in the current window. For example, Kuncheva (2009) and Nishida & Yamauchi (2007) compare the error rate in a reference window to the error rate in the current window using statistical tests to determine if a significant change in concept has taken place. In both cases the indicator is used to adjust the size of a sliding window, but it could also be used in a triggered rebuild framework to allow a classifier to be rebuilt on more recent training data when a change in concept is suspected. This type of approach might seem ideal, as new labelled training data is only needed when a change in concept is detected. However, labelled data is required to calculate the error rate indicator, so this type of approach does not reduce the need for labelled data.

This is where Confidence Distribution Batch Detection (CDBD) fits in. CDBD is a triggered rebuild approach introduced in (Lindstrom *et al.*, 2011) and further elaborated on in (Lindstrom *et al.*, 2013). CDBD aims to handle concept drift by using an indicator which can be calculated without using labelled data. The literature review in Section 3.4.3.2 cited a few approaches with a similar aim. Kifer *et al.* (2004) and Sebastião & Gama (2007) monitor the divergence between the distribution of a feature in a reference window and the current window and use this as a concept drift indicator. This divergence based indicator does not require labelled data to calculate, but it is limited in what kind of data it can be applied to. In domains like text classification, where there are a very large numbers of features, the distribution of any one feature value is not likely to be informative enough to warn of changes in concept.

Fan *et al.* (2004a) create an indicator from decision tree leaf node statistics, however, the approach only works with decision trees, which are not always the most suitable classifiers. Zliobaite (2010) and Lanquillon (Lanquillon, 1999) are the two approaches closest to CDBD. Both use classifier output to detect changes in concept, which removes the need to identify one feature for monitoring and makes them suitable for use on a large range of classifiers.

The remainder of this chapter is organised as follows, Section 6.1 gives a high level overview of CDBD and explains key components of the algorithm. Section 6.2 looks at how the CDBD indicator was evaluated and the result of the evaluation. Section 6.3 evaluates how CDBD works when the classifier is rebuilt based on the CDBD indicator and is followed by Section 6.4, a brief conclusion.

6.1 Overview

At a high level CDBD monitors an indicator for the occurrence of concept drift and when a change is suspected the classifier is rebuilt using recent data. The indicator is based on comparing the distribution of classifier confidences in two different parts of the data. This is based on the expectation that when feature change occurs, the distribution of classifier outputs will change significantly. CDBD is comprised of three phases: (1) Initialization, (2) Detection and (3) Adaptation. These are shown in Figure 6.1.

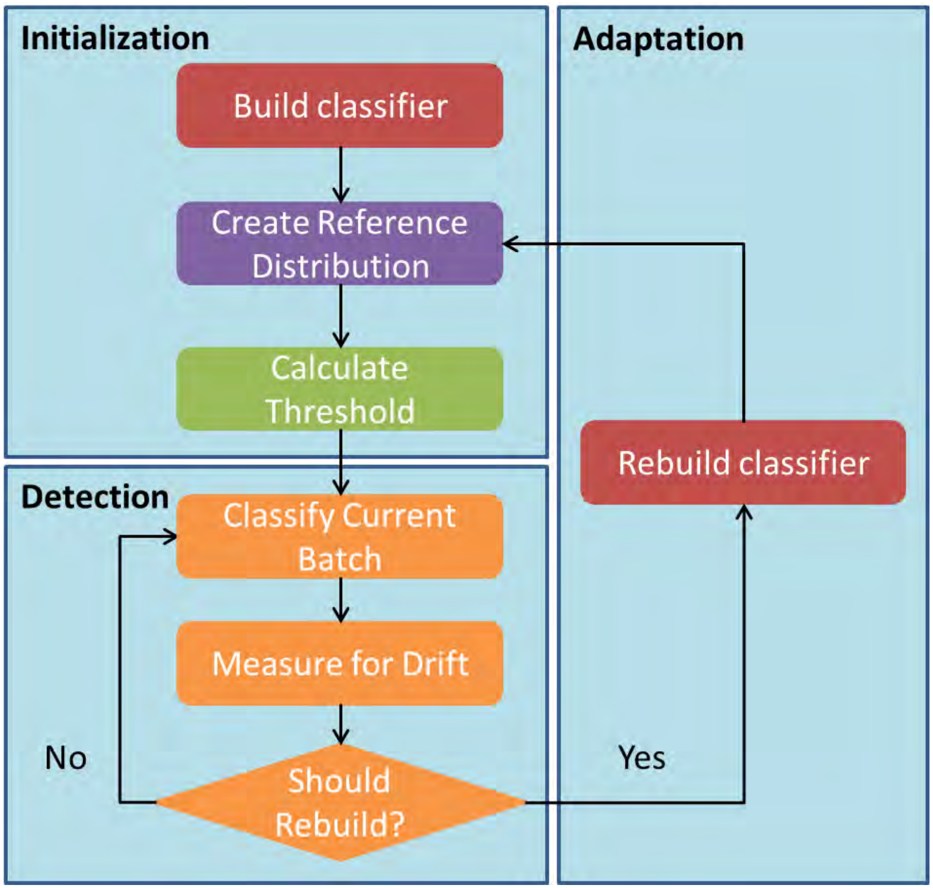


Figure 6.1: An overview of the CDBD approach.

The next three sections will expand on these phases and show how they combine to form the CDBD approach.

6.1.1 The Initialization Phase

The first step in CDBD is to train the initial classifier. After the initial classifier has been trained the data arrives as a stream of unlabelled instances, which are grouped into fixed size batches, and presented to the classifier for classification. CDBD uses the two-window paradigm to create the concept drift indicator, so a reference window is created from the first batch of instances immediately after the classifier is trained. The confidences in the reference window are then discretized into a histogram. The bins are selected by first identifying a range in which most of the confidences in the reference batch lie, then dividing the range into uniformly sized bins. Initial experiments seemed to indicate that between 7 and 13 bins produced a signal which was well correlated to changes in concept.

The next step is to set the threshold value used by CDBD to determine if a change of concept has taken place. The threshold is set using an approach similar to that used in (Lanquillon, 1999). The indicator value for the first v batches immediately after the reference batch is calculated by measuring the divergence between the distribution of the classifier outputs in the reference batch and the distribution of classifier outputs in each of those v batches. The exact method used to calculate the divergences will be covered in the next section. The threshold used by CDBD is $\mu + (\alpha * \sigma)$ where μ and σ are the mean and standard deviation of the v indicator values respectively while α is a user-specified threshold parameter. More information about how α was chosen empirically will be given in the Section 6.2.3.

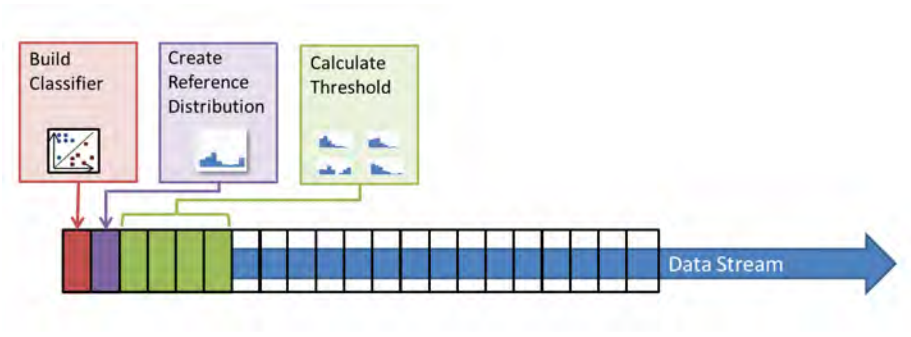


Figure 6.2: The Initialization Phase of CDBD.

Figure 6.2 shows a summary of the initialization phase of CDBD (with v set to four). Once the initialization phase is complete CDBD is ready to detect sudden concept shifts.

6.1.2 The Detection Phase

After the initialization phase the detection phase begins. During this phase the new unlabelled instances arrive in batches to be processed by CDBD. This phase consists of three steps:

- Classify all the instances in the current batch.
- Measure for drift by calculating the drift indicator.
- Decide if the value of the drift indicator warrants a rebuilding of the classifier on more recent data.

These steps will now be discussed in more detail. The first step is to classify all the instances in the current batch. A by-product from the classification is a confidence score for each instance in the batch.

The second step is to calculate the indicator value for the current batch. This is done by measuring the divergence between the distribution of the confidences for the current batch and the reference window. The indicator value for any batch can be calculated by discretizing the confidence values for that batch using the same bins as the reference window, normalizing both the reference histogram and the current histogram and then calculating the divergence using a divergence measure. The choice of measure used to calculate the divergence between distributions can greatly affect the indicator and subsequently the concept drift detection ability of the algorithm. Sebastião & Gama (2007) provide a good comparison of such measures and CDBD uses Kullback-Leibler divergence which was found to be particularly effective. The Kullback-Leibler divergence between two distributions represented as histograms can be defined as:

$$KL(h^1, h^2) = \sum_{i=1}^{k} h_i^1 log \frac{h_i^1}{h_i^2} \tag{6.1}$$

where h^1 and h^2 are both histograms with the same k bins, and h_i^1 refers to bin i of histogram h^1.

The final step is to decide if the classifier needs to be adapted. This is done using a rule, which will be referred to as a *trigger*. The trigger used in CDBD is a variation of the *Western Electric rules* (Montgomery, 2004). It fires when the indicator values for x out of the last y batches have been above the threshold. For example, a 3/5 trigger fires when three out of the last five indicator values are above the threshold, and so on. CDBD parameterised with a particular trigger will be referred to using CDBD-x-y, e.g. CDBD-3-5 means CDBD using a 3/5 trigger.

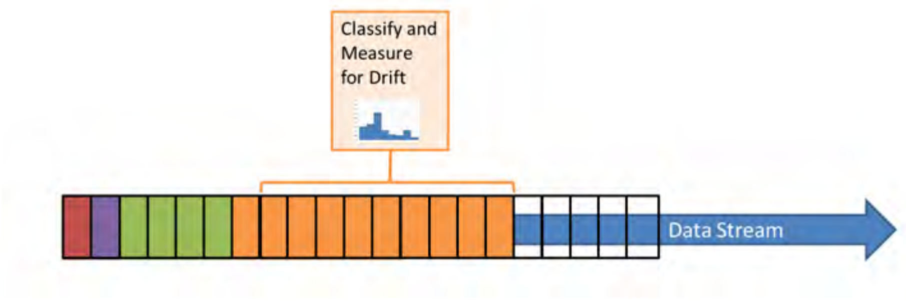


Figure 6.3: The Detection Phase of CDBD.

The process so far is illustrated in Figure 6.3. This process repeats with each new batch until the trigger fires. When the trigger fires the adaptation phase begins.

6.1.3 The Adaptation Phase

When the occurrence of concept drift is flagged by the trigger the classifier needs to be adapted. The simplest approach to adapting the classifier is to retrain it using the instances in the current batch as the new training data. However, class imbalance can be very common in real life data streams, so it is presumptuous to assume that the current batch will give a sufficiently balanced dataset from which to retrain the classifier. Instead, a balanced training set with d instances from each class, is constructed from as many batches as is required. The batch where the detection takes place becomes the beginning of the new training window and new test batches are added to the training window as they arrive until the number of instances of each class is equal to, or greater than d. At this point the d most recent instances of both classes are used as training data while the rest are discarded [1].

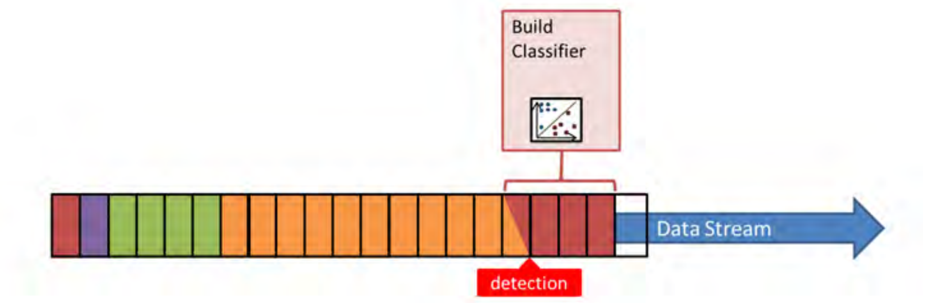


Figure 6.4: The Adaptation Phase of CDBD.

The process so far is illustrated in Figure 6.4. After the new classifier has been trained CDBD moves from the adaptation phase to the initialization phase where the reference window is reconstructed and the trigger threshold is recalculated and after that CDBD re-enters the detection phase. This process repeats indefinitely.

6.2 Signal Experiment

The overall goal of CDBD is to detect and handle concept drift while using as little labelled data as possible. However, the evaluation of CDBD can be broken down into two distinct sub-experiments.

The first experiment aims to establish the viability of using the confidence distribution divergence as a drift detection indicator. The second experiment couples the detection with an adaptation mechanism to establish if the two combined can handle concept drift using a limited number of labels.

[1]This is not the most efficient way to update the training window as the true class labels must be sought for every test instance in the training window. Further label savings could be achieved by improving the adaptation process and would make an interesting direction for future work. It is however outside the scope of this thesis.

This section deals with the first experiment, which will be referred to as the signal experiment. The signal experiment aims to examine if the CDBD signal can be used to detect changes in concept, but will also be used to empirically ascertain the best values for the CDBD parameters, such as the threshold and trigger values. This will be achieved by plotting the indicator over time to observe if the indicator value changes at the point where a concept shift was introduced and applying detection metrics to establish what threshold and trigger combination gives the best detection results.

6.2.1 Datasets

The signal experiment will not be evaluated on the spam datasets, as it is impossible to know where a change in concept took place in those datasets, which makes the evaluation of the signal impractical.

The indicator cannot be evaluated on conditional change datasets as it is not possible to detect conditional change without using labelled data and the goal of the CDBD indicator is to detect concept drift without using labelled data. Detecting a change in a data stream with an imbalanced and changing class distribution is a challenge significantly harder than detecting a change in a data stream with a balanced class distribution. The balanced datasets are therefore not used in the evaluation as they are superfluous, if CDBD works on a imbalanced data stream it is very likely to work on a balanced one.

The gradual concept drift datasets will also be discarded for this experiment as they do not have a fixed change point, which complicates the evaluation metrics. Instead the precedent set by other detection approach evaluations (like the ones in (Kuncheva, 2009) and (Lanquillon, 1999)) will be followed by evaluating CDBD on sudden concept shift datasets.

CDBD will be evaluated on imbalanced, concept shift induced, feature change datasets generated from the Reuters, 20 newsgroups and news sources corpora, i.e. Reuters$_{SIF}$, 20NG$_{SIF}$ and NS$_{SIF}$.

The specifics of the datasets are given in Table 6.1. The name column shows what concepts were created for each dataset, while the other four columns list the target topic, size and class distribution for each concept. The feature change datasets were given their true labels as outlined in Section 4.1.2.2. All documents belonging to the target topic in each concept are labelled as relevant for that concept. All other documents are labelled as non-relevant. As the target topic changes between concepts, documents that belong to a target topic will not appear in any other concept that has a different target topic to prevent introducing concept drift based on a conditional change. So, for example, for the NS$_{SIF}$ dataset no documents from the *Business* topic will appear in concept C2.

6.2.2 Methodology

For each experiment the initial classifier was trained on the training concept for that dataset, as described in Table 6.1. The documents in concepts C1 and C2 were grouped into batches of 100 documents to be presented to the classifier for classification. CDBD can be used with any classifier that produces a score that can be interpreted as an estimate of confidence that

Table 6.1: Details of the datasets used in the signal experiment evaluation.

Dataset	**Name**	**Target topic**	**Size**	**No. Rel.**	**No. Non-rel**
20NG$_{SIF}$	Training	comp.*	300	150	150
	C1	comp.*	4600	1157	3443
	C2	religion	4300	1259	3041
	C3	sci.*	2500	1468	1032
Reuters$_{SIF}$	Training	earn	300	150	150
	C1	earn	5200	1476	3724
	C2	trade	5200	402	4798
	C3	acq	4200	641	3559
NS$_{SIF}$	Training	Business	300	150	150
	C1	Business	14900	3712	11188
	C2	Sport	15000	4867	10133
	C3	Entertainment	15000	2350	12650

a prediction made by the classifier is correct. For example, in the case of a single prototype classifier the distance between a test instance and the closest class prototype might be considered indicative of how confident the classifier is that the test instance is of the same class as that class prototype. In these experiment an SVM with a linear kernel was used as the classifier and the decision values from the SVM were used as classifier confidence outputs.

The discretization of the classifier confidences requires that a set of bins is identified. Initial experiments seemed to indicate that the decision values produced by the SVM on these datasets lay in the -2 to 2 range. The preliminary experiments also showed that between 7 and 13 bins produced a signal which was well correlated to changes in concept so the bins $\{-2.0, -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, 2.0\}$ were used in all subsequent experiments, with any value outside the range put in the first and last bin respectively.

This primary goal of this set of experiments is to establish if there is a relationship between the indicator signal and changes in concept, so in this set of experiments the batches which caused a trigger to fire were noted, but the classifier was not adapted. A visual inspection of the signal plotted over time should answer the question if there is a relationship between the signal and changes in concept. The signal will also be evaluated more formally using a two-tailed t-test to determine whether or not the signal before the change in concept is significantly different from the one after the change in concept.

The secondary goal, assuming the signal works, is to establish which combination of threshold and trigger values give the best detection result. To this end the mean and standard deviation was calculated on the indicator value for the 6 batches after the reference batch.

The trigger can be evaluated in a quantitative way as detections should not take place in concept C1, and should take place in C2. More formally, the metrics used were True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) rates. A detection in C1 is an FP, while a detection in C2 is a TP. Conversely, a non-detection in C1 is a TN, while a non-detection in C2 is an FN. The relative importance placed on the system detecting a change when it should not have (an FP) and the system not detecting a change when it should have (an FN) is highly application dependant. For example, in a trial the presumption of innocence stipulates that an FP (convicting an innocent person) is highly undesirable.Conversely, cancer prediction systems should produce very few FNs, as not detecting cancer when it is present is very serious.

These figures can be further refined into accuracy ($\frac{TP+TN}{TP+FP+TN+FN}$), precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$) numbers. The final metric used is the Run Length (RL), which is defined as the number of batches between the batch where the concept shift occurred and where the detection algorithm flags a change in concept. In this set of experiments the 1/1, 2/3 and 3/5 triggers were evaluated on each dataset to find which trigger (or triggers), consistently detects a change in concept in C2 while not erroneously detecting a change in concept in C1.

6.2.3 Results

Figures 6.5, 6.6, and 6.7 show the indicator value over time on the 20NG$_{SIF}$, Reuters$_{SIF}$, and NS$_{SIF}$ datasets. The figures also show how varying the user-specified α parameter creates different detection thresholds. An α value of, one, two and three are shown as "1 Std. Dev.", "2 Std. Dev."and "3 Std. Dev" respectively (for more information about how the threshold is calculated see Section 6.1.1). The concept drift point is marked by the dashed vertical line. These graphs show that although the signal is not perfect, in general the indicator values before the change in concept are substantially different from the values after the change in concept. This was confirmed using unpaired two-tailed t-tests which showed a statistically significant difference (at the 95% confidence level) in indicator values before and after the concept drift on all three datasets.

On the NS$_{SIF}$ dataset it is evident from Figure 6.7 that the signal starts increasing before the change in concept. This could be due to naturally

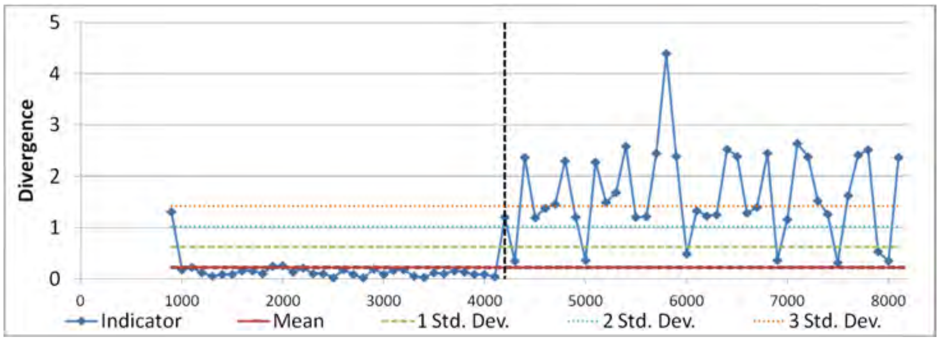


Figure 6.5: Signal over time on the 20NG$_{SIF}$ dataset.

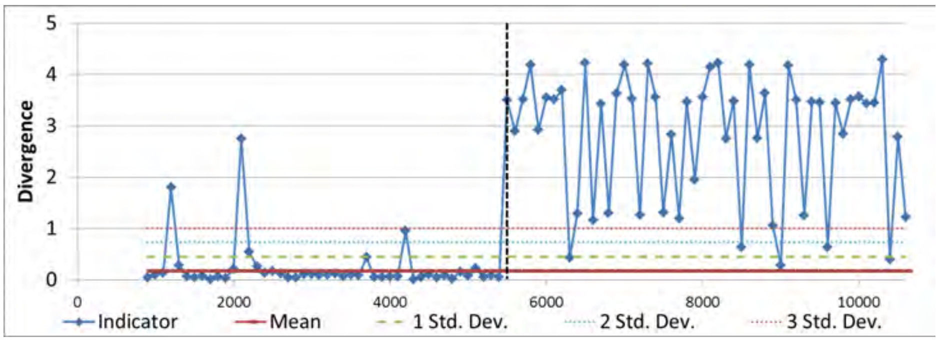


Figure 6.6: Signal over time on the Reuters$_{SIF}$ dataset.

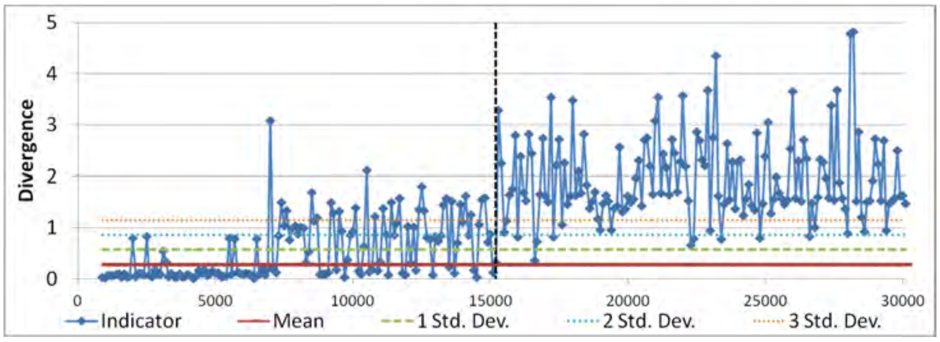


Figure 6.7: Signal over time on the NS$_{SIF}$ dataset.

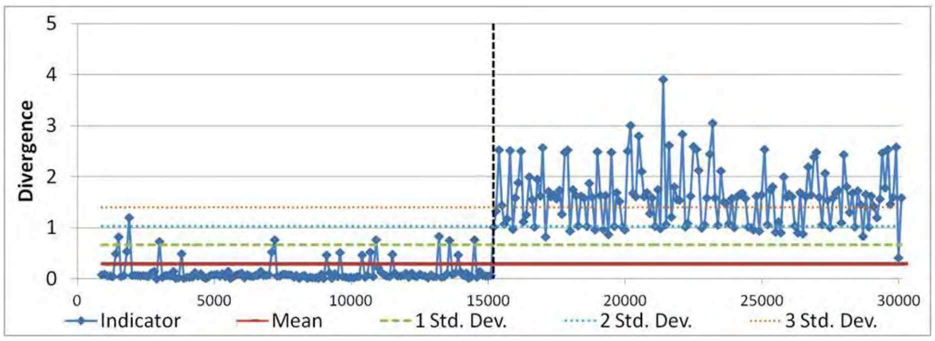


Figure 6.8: Signal over time on the NS$_{SIF}$ dataset with randomised instance order.

occurring concept drift in the data, as might be expected in real data (particularly in the NS$_{SIF}$ dataset as it is considerably larger than the other datasets used) making a gradual change in concept more likely to occur.

One way to test the hypothesis that there is naturally occurring concept drift due to the chronological ordering of the data is to randomise the order of the instances within each concept. Figure 6.8 shows the same experiment but with the instance order within C1 randomised and the instance order within C2 randomised.

The increase in the signal before the change in concept seen in Figure 6.7 is not present in Figure 6.8, which supports our argument that the signal increase seen in Figure 6.7 is due to natural concept drift. It also strengthens the argument made in Chapter 4 that ordering the data chronologically can preserve interesting characteristics of the data which might otherwise be lost and which might make the classification problem harder, but more realistic.

Once it has been established that the indicator signal seems to be related to changes in concept then Figures 6.5, 6.6 and 6.7 can be used to help choose a detection threshold. Based on these graphs it seems like most signal values are below the mean plus one standard deviation before the change in concept, and above the mean plus one standard deviation after the change in concept. A higher detection threshold of the mean plus two or three times the standard deviation seems unable to separate the before and after signal so a α of one was used giving a detection threshold of the *mean plus one standard deviation*, which was used for all the subsequent experiments.

Another interesting finding is that even though the signal is reasonably well behaved, it does often break the mean plus one standard deviation threshold before the concept changes. This shows the importance of using the Western Electric rules which require multiple indicator values to be above the threshold within a short time frame, rather than a simpler trigger which would cause a rebuild every time the indicator value goes above the threshold.

It is hard, however, to establish what specific trigger rule is the most suitable based on Figures 6.5 to 6.7. For this a more quantitative evaluation is needed. Table 6.2 shows the detection results for the 1/1, 2/3 and 3/5 triggers on the three datasets. The second and third column show how many detections took place in each concept for that trigger. These detection results can be further refined into FPs, TPs, FN and TNs (using the definitions from Section 6.2.2), which in turn can be distilled into accuracy, precision and recall scores.

Table 6.2: Summary table of detection results.

(a) Detections on the 20NG$_{SIF}$ dataset.

	#Detections									
Trigger	**C1**	**C2**	**#FP**	**#TP**	**#FN**	**#TN**	**Acc**	**Prec**	**Rec**	**RL**
1/1	1	33	1	33	7	32	0.89	0.97	0.83	0
2/3	0	36	0	36	4	33	0.95	1.00	0.90	1
3/5	0	37	0	37	3	33	0.96	1.00	0.93	2

(b) Detections on the Reuters$_{SIF}$ dataset.

	#Detections									
Trigger	**C1**	**C2**	**#FP**	**#TP**	**#FN**	**#TN**	**Acc**	**Prec**	**Rec**	**RL**
1/1	4	49	4	49	3	42	0.93	0.92	0.94	0
2/3	2	51	2	51	1	44	0.97	0.96	0.98	1
3/5	0	50	0	50	2	46	0.98	1.00	0.96	2

(c) Detections on the NS$_{SIF}$ dataset.

	#Detections									
Trigger	**C1**	**C2**	**#FP**	**#TP**	**#FN**	**#TN**	**Acc**	**Prec**	**Rec**	**RL**
1/1	59	148	59	148	3	84	0.79	0.71	0.98	1
2/3	57	148	57	148	3	86	0.80	0.72	0.98	2
3/5	63	150	63	150	1	80	0.78	0.70	0.99	0

The precision and recall values give an understanding of the detection characteristics of a particular trigger. A perfect trigger would have a precision and recall of one. A recall of one means that the trigger fired every time in C2, when the concept is different from the concept the classifier was trained on, and a precision of one means that it only fired in C2. The results in the detection tables show that no trigger is a perfect detector, but all three did well on the 20NG$_{SIF}$ and Reuters$_{SIF}$ datasets, but not as well on the NS$_{SIF}$ dataset due to natural concept drift in the data stream. The best trigger is dependant on the relative importance of detecting every change (recall) and not falsely detecting (precision) a change in concept. The run length column (RL) suggest that the 1/1 trigger is able to detect a change in concept the fastest, but the precision column shows that it is susceptible to false positives. The 2/3 and 3/5 triggers are more cautious approaches which make them slower to react to changes in concept. In a domain where the cost of labelling is high and a false positive trigger would result in unnecessary labelling, a 2/3 or 3/5 trigger would be better suited than a 1/1 trigger as they have a higher precision score. However, there does not seem to be a substantial difference between the 2/3 and 3/5 triggers.

The signal experiment shows that the CDBD signal can be used to detect concept drift in a document stream. The signal experiment is, however, artificial as when the full CDBD approach is applied the classifier will be rebuilt and the detection algorithm re-initialised when detection occurs. The aim of the next experiment is to evaluate whether the CDBD detection mechanism coupled with a rebuild policy could handle concept drift.

6.3 Detection and Rebuild Experiment

The result from the signal experiment suggest that a confidence distribution divergence based indicator can be used to detect concept drift. The aim of this experiment, which will be referred to as the detection and rebuild experiment, is to investigate if the CDBD signal coupled with a trigger and rebuild policy can maintain classification accuracy in a data stream containing sudden changes in concept while using very few labels. CDBD will also be compared against other triggered rebuild benchmarks (which will be detailed shortly in Section 6.3.2), to establish if CDBD is on par with comparable approaches which use a fully labelled data stream.

6.3.1 Datasets

The detection and rebuild experiment was evaluated on feature change datasets, just like the signal experiment, as conditional change can not be detected without labelled data. The gradual drift datasets were also discarded as CDBD is designed to detect sudden concept shifts.

Both balanced and imbalanced datasets were used so that the affect of a variable class distribution on triggered detection algorithms could be evaluated. The Spam datasets were also included to see if CDBD could handle concept drift on a dataset with an unknown drift cause and type.

This gives a total of eight datasets created using the concepts listed in Table 6.3. All documents with the target topic in each concept are labelled as relevant for that concept. For each corpora a balanced and an imbalanced dataset was generated. The imbalanced datasets retain the class distribution one gets when the documents are ordered chronologically. The documents in the balanced datasets are also ordered chronologically, but they are filtered to remove documents until the class distribution in each batch is 50-50 (using the topic filter as described in Section 4.1.3.2).

Dataset	**Name**	**Target topic**	**Size**	**No. Rel.**	**No. Non-rel**
$20NG_{SBF}$	Training	comp.*	300	150	150
	C1	comp.*	2000	1000	1000
	C2	religion	2100	1050	1050
	C3	sci.*	1800	900	900
$20NG_{SIF}$	Training	comp.*	300	150	150
	C1	comp.*	4600	1157	3443
	C2	religion	4300	1259	3041
	C3	sci.*	2500	1468	1032
$Reuters_{SBF}$	Training	earn	300	150	150
	C1	earn	1300	650	650
	C2	trade	1300	650	650
	C3	acq	1100	550	550
$Reuters_{SIF}$	Training	earn	300	150	150
	C1	earn	5200	1476	3724
	C2	trade	5200	402	4798
	C3	acq	4200	641	3559
NS_{SBF}	Training	Business	300	150	150
	C1	Business	7900	3950	3950
	C2	Sport	8000	4000	4000
	C3	Entertainment	5700	2850	2850
NS_{SIF}	Training	Business	300	150	150
	C1	Business	14900	3712	11188
	C2	Sport	15000	4867	10133
	C3	Entertainment	15000	2350	12650
$Spam_1$	Training	N/A	300	150	150
	Testing	N/A	9900	1036	8864
$Spam_2$	Training	N/A	300	150	150
	Testing	N/A	8200	688	7512

Table 6.3: Details of the datasets used in the detection and rebuild experiment.

6.3.2 Methodology

The methodology for the detection and rebuild experiment is very similar to the methodology used for the signal experiment. The only difference is that in the rebuild experiment the classifier is updated, as outlined in Section 6.2.2, with the distribution parameter d set to 150, when the trigger fires. The signal experiment was inconclusive on the question of what trigger rule is the most appropriate. All three triggers (1/1, 2/3 and 3/5) were therefore used in the experiment. In addition to CDBD-1-1, CDBD-2-3 and CDBD-3-5 the following benchmarks were also evaluated:

- **No Update**: No concept drift handling.
- **Sliding Window**: A fixed distribution, fixed-size sliding window approach.
- **Window Resize Algorithm for Batch Data** (WRABD) (Kuncheva, 2009): The triggered rebuild WRABD approach which detects drift when the classification error in the current batch is above the mean plus three standard deviations. The mean and standard deviation is calculated using 10 batches of data, as in (Kuncheva, 2009).
- **Perfect Detection**: A notional approach which is set to trigger a classifier rebuild at each of the known concept drift points (this is used purely as an indicator of the performance limits of the CDBD approach).

Both the perfect detector and WRABD use the same rebuild approach as CDBD so that they can be applied to data with varying and imbalanced class distributions. CDBD was evaluated and compared to the above approaches in terms of average class accuracy and fraction of labels used.

6.3.3 Results

Table 6.4 summarises the performance of all approaches. The table is vertically divided into concept drift handling approaches. For each approach the first column shows the average class accuracy obtained by the classifier, while the second column shows the fraction of instances labelled, excluding the initial training data.

The first important thing to note from these results is that considering the difference in performance between the no update and sliding window it is evident that all datasets exhibit substantial concept drift which can be handled.

It is also clear that the sliding window approach achieves the best performance on all datasets, although at the expense of 100% label usage. It is interesting to note that sliding window obtains higher average class accuracy than WRABD. It is likely that reason for this is that the sliding window approach recovers faster, in terms of average class accuracy, after a change in concept occurs. The sliding window also has another advantage, in that it handles any gradual concept drift which might be present in the data, whereas WRABD only rebuilds when a significant change is suspected.

The overall result seems to be that out of the CDBD family of approaches

Table 6.4: The average class accuracy and fraction of labels used for each of the tested approaches.

	No Update		**CDBD-1-1**		**CDBD-2-3**		**CDBD-3-5**		**WRABD**		**Perfect Detection**		**Sliding Window**	
20NG$_{SBF}$	0.68	0	0.73	0.10	0.75	0.10	0.76	0.10	0.77	1	0.77	0.10	0.81	1
20NG$_{SIF}$	0.61	0	0.79	0.28	0.78	0.16	0.76	0.09	0.76	1	0.78	0.10	0.84	1
Reuters$_{SBF}$	0.63	0	0.79	0.16	0.78	0.16	0.75	0.16	0.80	1	0.80	0.16	0.84	1
Reuters$_{SIF}$	0.63	0	0.78	0.33	0.79	0.23	0.78	0.19	0.72	1	0.77	0.19	0.86	1
NS$_{SBF}$	0.60	0	0.84	0.13	0.81	0.06	0.83	0.04	0.83	1	0.84	0.03	0.85	1
NS$_{SIF}$	0.60	0	0.81	0.17	0.84	0.16	0.83	0.09	0.81	1	0.85	0.04	0.85	1
Spam$_1$	0.81	0	0.96	0.45	0.93	0.24	0.90	0.21	0.93	1	-	-	0.97	1
Spam$_2$	0.72	0	0.86	0.44	0.88	0.42	0.84	0.45	0.82	1	-	-	0.94	1

both CDBD-2-3 and CDBD-3-5 obtain a good balance between high average class accuracy and low label usage. Most importantly, these CDBD approaches achieve comparable average class accuracies to the triggered rebuild WRABD approach, while using only a fraction of the labelled data.

The temporal behaviour of CDBD is best illustrated by plotting the average class accuracy over time. For brevity CDBD-2-3 will be presented as representative of CDBD as it is not as sensitive to signal fluctuations as CDBD-1-1 nor as conservative as CDBD-3-5. For the graphs showing CDBD-1-1 and CDBD-3-5 please see Appendix D.1.

Figures 6.9 to 6.16 show a more detailed exploration of CDBD-2-3. These figures plot the average class accuracy over time, using a five point moving average for smoothing, for the sliding window, no update, CDBD-2-3 and WRABD approaches. The concept drift points are marked by dashed vertical lines and the concept drift detection points for the CDBD-2-3 trigger are marked with squares at the top and the detection points for WRABD are marked with a triangle.

Figure 6.9 is illustrative of the full suite of graphs, and clearly illustrates the concept drift process. This graph shows classifier performance on the 20NG_{SBF} dataset as the relevant topic is changed from "comp.*" (which was also relevant during training) to "religion", and finally to "sci.*". The sliding window approach manages to maintain a reasonably constant classification accuracy throughout the concept changes, albeit with a slight dip right after the two changes in concept occur. The CDBD-2-3 profile is an almost perfect template for what is expected from a triggered rebuild strategy. After

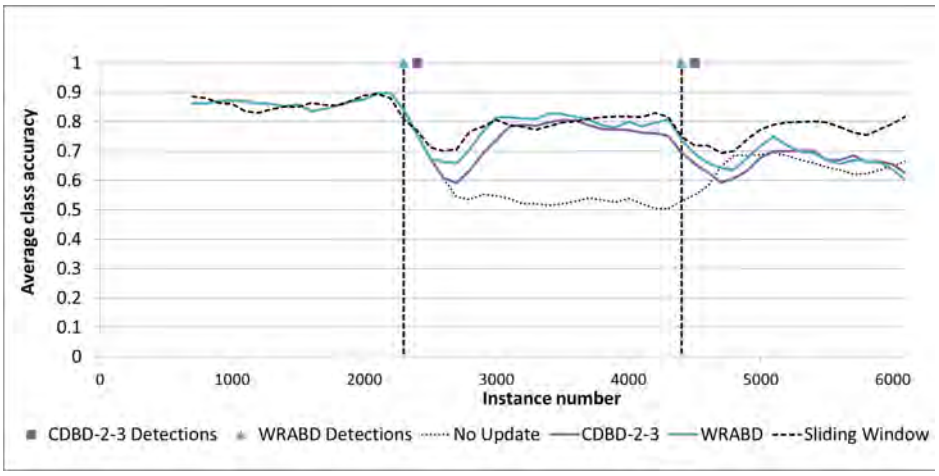


Figure 6.9: CDBD-2-3 average class accuracy over time on the 20NG$_{SBF}$ dataset.

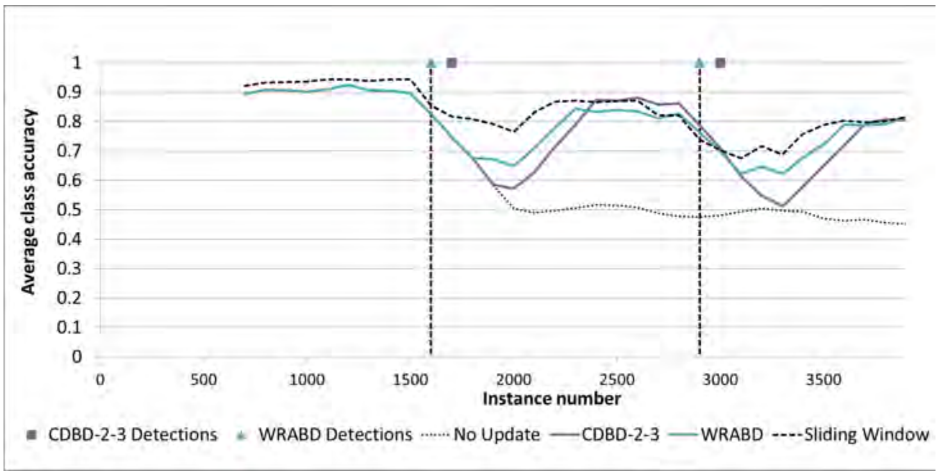


Figure 6.10: CDBD-2-3 average class accuracy over time on the Reuters$_{SBF}$ dataset.

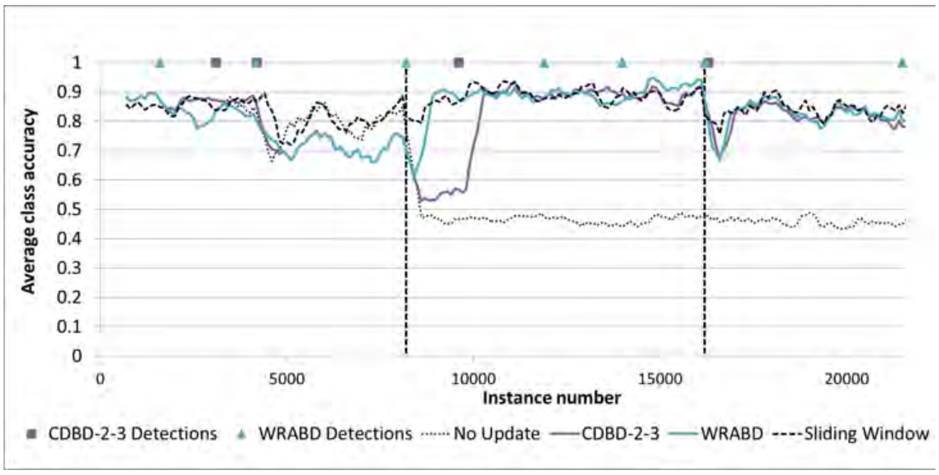


Figure 6.11: CDBD-2-3 average class accuracy over time on the NS$_{SBF}$ dataset.

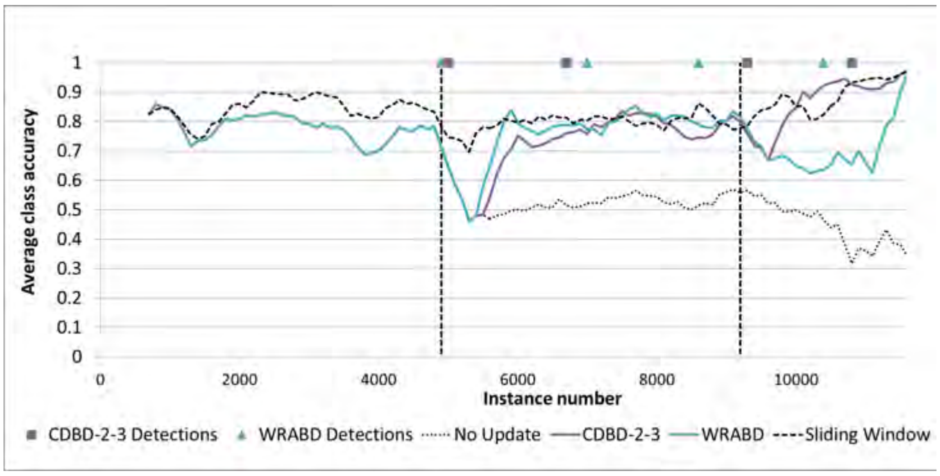


Figure 6.12: CDBD-2-3 average class accuracy over time on the 20NG$_{SIF}$ dataset.

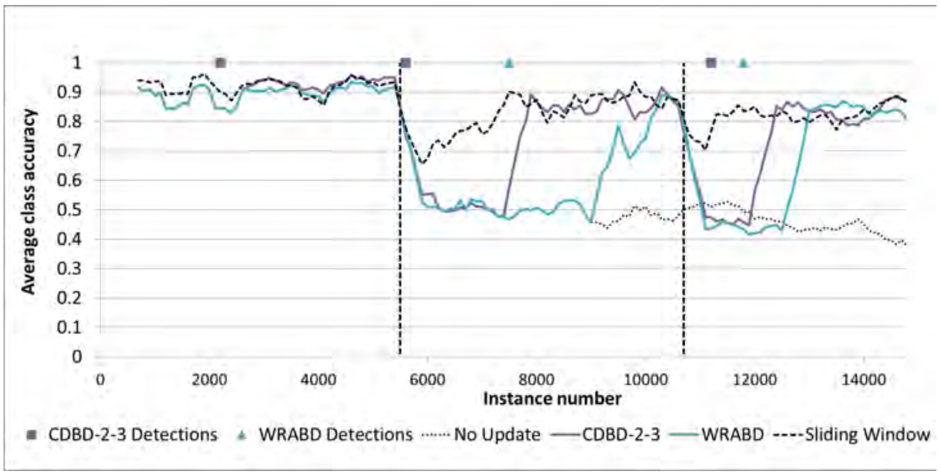


Figure 6.13: CDBD-2-3 average class accuracy over time on the Reuters$_{SIF}$ dataset.

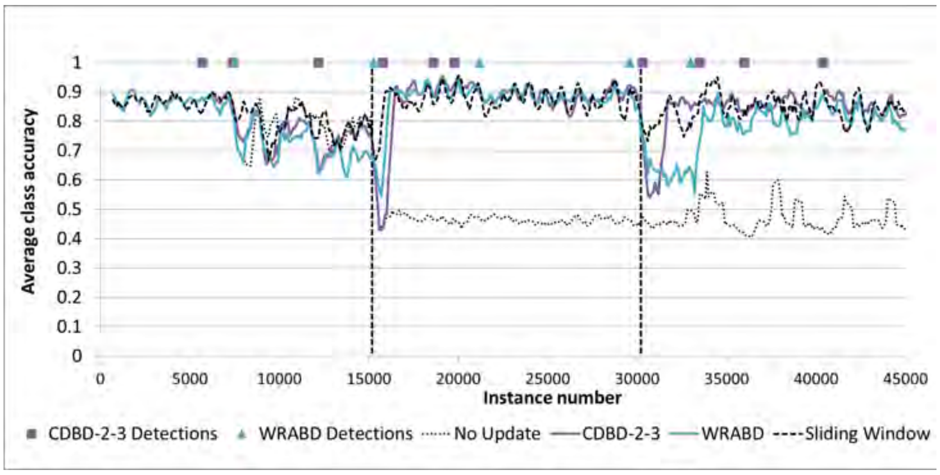


Figure 6.14: CDBD-2-3 average class accuracy over time on the NS$_{SIF}$ dataset.

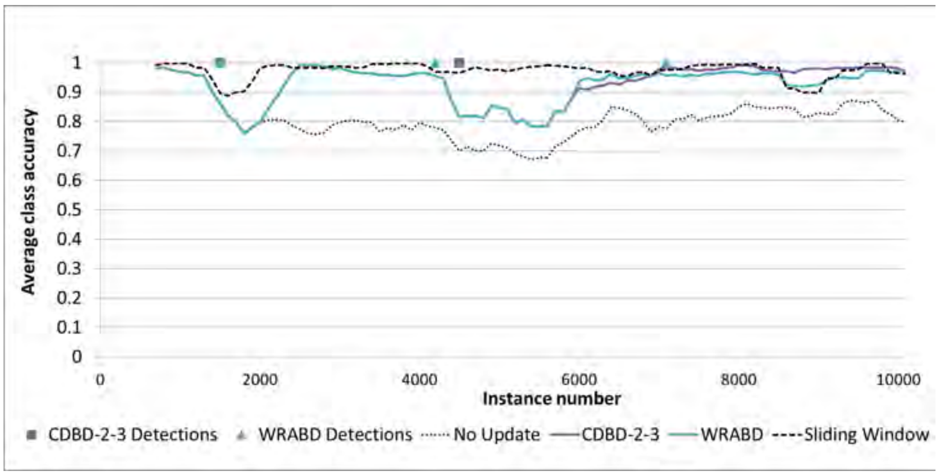


Figure 6.15: CDBD-2-3 average class accuracy over time on the Spam_1 dataset.

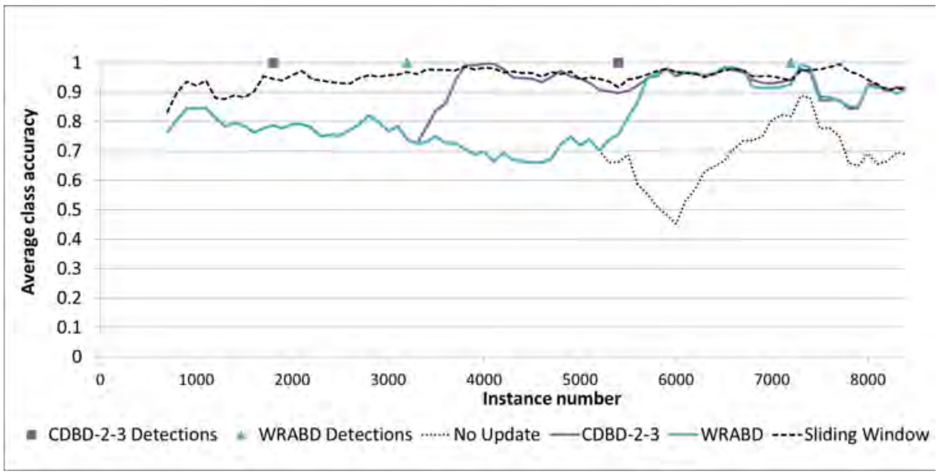


Figure 6.16: CDBD-2-3 average class accuracy over time on the Spam_2 dataset.

the first concept change the performance falls off dramatically. A concept drift detection is triggered almost immediately, however, and performance improves again. The visible delay between detection and performance improvement is partly due to the use of a moving average line and partly due to the time it takes to gather enough data to build a new training set. Figure 6.10 tells an almost identical story to Figure 6.9. WRABD performs very well in both diagrams as it accurately detects the change in concept at both shift points. In Figure 6.11 both WRABD and CDBD-2-3 flag a few changes in concept in the NS$_{SBF}$ data outside the artificially induced concept change. In this case, however, it might be due to natural concept drift within the data. At around the same period the performance of the sliding window and the no update approaches also seem to undergo a series of performance fluctuations, which would suggest that the nature of the data is changing.

Figures 6.12 to 6.14 show that the performance of the four approaches on imbalanced versions of the same three datasets is not quite as impressive, as both CDBD-2-3 and WRABD flag a number of false positives, and in some cases once detections are made it takes a considerable amount of time before the classification performance improves. This is because the class imbalance in the data means it takes a large number of batches to build a new balanced training set with which to update the classifier. The amount of labelled data required in these cases is also much higher than for the balanced datasets for the same reason. In all cases, however, performance is still considerably better than the no update approach, comparable to WRABD, and in most cases close to that of the sliding window approach. These results reinforce the conclusion that CDBD can handle concept drift without using a large numbers of labelled instances.

The classifier performances shown in Figures 6.15 and 6.16 for the real Spam$_1$ and Spam$_2$ datasets are a little harder to interpret as the nature of the concept drift present is unknown. The difference between the performance of the sliding window approach and no update approach clearly show that concept drift is present. As this is real data this drift is likely to be due to a mixture of feature and conditional change, so it is interesting that the CDBD approach is able to detect drift and successfully respond to it, as CDBD is only designed to handle feature change. The response is far from perfect, however, and the adaptation mechanism used by CDBD and WRABD suffers from the fact that it can take an excessively large number of batches before an updated balanced training set can be created. During this time the out-of-date classifier is used to classify the stream, which is suboptimal. An improvement to the rebuild process could decrease the amount of labelled data used and ensure that a classifier trained on up-to-date data is used earlier.

The performance of the perfect detection approach is included in Table 6.4 as an indication of the smallest amount of labelled data a triggered rebuild approach needs, while still handling concept drift effectively. Any differences between the amount of data used by the CDBD approaches and perfect detection show the impact of false positives, while the significantly larger amounts of labelled data required for the 20NG$_{SIF}$, Reuters$_{SIF}$, NS$_{SIF}$, Spam$_1$ and Spam$_2$ datasets show the impact of class imbalance.

Taken together the results of the evaluation experiments described above show that CDBD, a triggered rebuild concept drift detection approach that is based on classifier output and does not need full access to true class labels, can handle concept drift as effectively as WRABD, a triggered rebuild approach that requires all the instances to be labelled. While CDBD does not perform as effectively as the sliding window approach, the potential that triggered rebuild approaches have for handling concept drift in scenarios constrained by high labelling cost is clearly evident, particularly on the larger datasets.

6.4 Conclusion

In this chapter CDBD, a triggered rebuild approach to handling concept drift was presented. CDBD uses a concept drift indicator which is based on the divergence between the classifier confidence scores in a reference window and a moving window. The advantage of CDBD over most other triggered detection approaches is that the CDBD indicator does not require labelled data to calculate.

The first experiment showed that the CDBD indicator responds to changes in concept. It also showed that the signal can be noisy, which necessitates using Western Electric rules to decide when to rebuild the classifier. The rules which required two out of three (2/3) or three out of five (3/5) values to be above the threshold achieve the best balance between detecting changes and false positives (for more information about how the triggers were compared, please see Appendix E.2).

The second experiment showed that the CDBD indicator coupled with a trigger and adaptation mechanism was able to detect changes in concepts and adapt the classifier to handle sudden concept shifts. The evaluation results also indicate that CDBD works on data streams with a varying class distribution, but that it works even better on data where each batch has a fixed class distribution. Another important finding is that CDBD performed more effectively than a similar triggered rebuild approach that require full access to actual class labels, while using only a fraction of the labelled data in rebuilding the classifier. It was also noted that the rebuild policy used in CDBD is sub-optimal and an interesting area for future research.

However, both CDBD and our other concept drift handling approach DVS have so far only been evaluated against benchmarks which require a fully labelled data stream. In the next chapter DVS and CDBD will be compared to each other and competing approaches which also attempt to handle concept drift while trying to reduce the amount of labelled data used.

Chapter 7

Evaluating Concept Drift Handling Approaches using Multiple Criteria

Chapter 1 introduced two high level approaches to handling concept drift, continuous rebuild approaches and triggered rebuild approaches. The two high level approaches differ in when they adapt the classifier: continuous rebuild approaches are characterised by the classifier being continuously adapted whereas triggered rebuild approaches only adapt the classifier when a sizeable change in concept is suspected. However, most concept drift handling approaches, regardless of when they adapt the classifier, rely on a fully labelled data stream.

In previous chapters Decision Value Sampling (DVS), a novel continuous rebuild approach and Classifier Distribution Batch Detection (CDBD), a novel triggered rebuild approach have both been shown to be capable of handling concept drift without relying on a fully labelled data stream. The fact that either approach is viable raises the interesting question of which approach is the most appropriate for handling concept drift with a reduced amount of labelled data. The comparisons carried out so far have mainly focused on how DVS and CDBD perform in terms of average class accuracy. A more comprehensive evaluation of continuous and triggered rebuild approaches requires that the appropriateness of an approach is judged on two criteria, the average class accuracy obtained by the approach and the fraction of labels used.

Section 4.3.1 described the Multi-criteria Decision Analysis (MCDA) approach Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) which can be used to combine the two criteria into a single measure and also has the ability to weight the two criteria differently to reflect scenarios where one criteria is more important than the other. The evaluation performed in this chapter will therefore use TOPSIS when comparing approaches on multiple criteria.

TOPSIS does not specify what weights should be used, as the appropriate set of weights is completely dependent on the classification task at hand. To establish suitable weights the cost of acquiring labelled data and the cost of misclassification must be determined. In many real life scenarios both of these costs can be quantified. For example, in the case of fraud detection the cost of investigating suspected cases of fraud, and the cost of the fraud going unnoticed can be estimated based on experience. Once the costs have been established they can be normalised and used with TOPSIS to evaluate concept drift handling approaches.

Another aspect of the evaluation so far is that DVS and CDBD, have been compared to benchmark approaches which require a fully labelled data stream, but not against other approaches which attempt to handle concept drift in a partially labelled data stream, such as Variable Uncertainty with Randomness (VUR) (Zliobaite *et al.*, 2011) and Confidence Range Batch Detection (CRBD) (Lanquillon, 1999). This chapter will continue the evaluation by using TOPSIS to compare DVS, CDBD, VUR and CRBD against each other and baseline concept drift handling approaches to establish which approach is the most appropriate for handling concept drift in the context of expensive labels. There will also be an attempt, if possible, to extrapolate from the results a more expansive comment on the strengths, weaknesses and applicability of continuous and triggered rebuild approaches in general.

The remainder of this chapter is organised as follows Section 7.1 will list and briefly discuss the approaches which will be compared. Section 7.2 will deal with the evaluation of the two new approaches VUR and CRBD, while Section 7.3 will be an evaluation of various concept drift handling approaches using TOPSIS to combine average class accuracy and fraction of labels used into one evaluation metric. The chapter will finish with Section 7.4, a brief conclusion.

7.1 Approaches

In some scenarios either a continuous or triggered rebuild approach is suitable, for example concept drift caused by conditional change does not manifest itself in a way which can be detected without some labelled data. This makes triggered approaches which do not use an indicator based on labelled data unsuitable for this type of problem. However, there are also scenarios where a triggered approach might result in a larger label saving, such as a data stream which experiences infrequent sudden concept shifts, yet is stable in between these shifts. In this case a triggered rebuild approach might use less data as a continuous rebuild approach requires some labels regardless of whether the concept is changing or not, whereas a triggered rebuild approach only requires labelled data when a change in concept is suspected. The experiments performed in this chapter are all grounded in a scenario where either a continuous or triggered rebuild approach could be used.

The approaches evaluated in this chapter are as follows:

- No Update
- Error rate based detection (WRABD) (Kuncheva, 2009)
- Fixed size sliding window
- Random sampling[1]
- Variable Uncertainty with Randomness[1] (VUR) (Zliobaite *et al.*, 2011)

[1]using a labelling budget, b of 0.05, 0.10, and 0.15

- Decision Value sampling[1](DVS)

- Confidence Range Batch Detection (CRBD) (Lanquillon, 1999)

- Classifier Distribution Batch Detection[2] (CDBD)

No update, WRABD and sliding window are included as representative baselines. No update gives an indication of what kind of accuracy can be expected if no labelled data is used, whereas WRABD and sliding window show the accuracies that can be obtained by a triggered and continuous rebuild approach respectively, assuming labelling cost is not a factor.

Random sampling will be included in the comparison to show how well a naive sampling approach coupled with a sliding window can perform.

VUR, which is described in more detail in Section 3.4.3.1, is a sliding window approach combined with an active learning based sampling strategy, and is similar to DVS. VUR will be included in the comparison as it is a state of the art concept drift handling approach which aims to handle concept drift while using as little labelled data as possible.

CRBD, which is described in more detail in Section 3.4.3.2, is the approach in the literature most closely related to CDBD as it is also a triggered rebuild approach with an indicator based on classifier output. CRBD estimates a class confidence range from the training data. A change in concept is flagged if in subsequent batches of data the number of predictions in that range exceeds the detection threshold.

Some of the approaches used in the experiments in this chapter have an

[2]using a 1/1, 2/3 and 3/5 trigger

algorithm parameter which influences how much labelled data an approach uses. If the approach uses a parameter which influences its label usage then the parameter value(s) recommended by the developers of the approach will be used. In the case of the labelling budget parameter for DVS and the trigger parameter for CDBD the values used are based on a series of experiments performed by the author and described in Appendix E.2.

The average class accuracy and labels used evaluation metrics for all approaches except VUR and CRBD are available from Sections 5.3 and 6.3.3, so the next section will deal with the experiment used to collect the average class accuracy and fraction of labels used by VUR and CRBD.

7.2 Benchmarking the State of the Art Approaches

This experiment aims to evaluate the two state of the art approaches, VUR and CRBD, in terms of average class accuracy and fraction of labels used. This section will cover the methodology used in the experiment and an analysis of the evaluation results.

7.2.1 Methodology

The evaluation will be performed on drift induced datasets with feature change and real datasets with an unknown concept drift cause. This allows for a fair comparison as the continuous rebuild approaches listed in the previous section can handle both conditional and feature change, whereas the triggered rebuild approaches are limited to feature change. The feature change datasets will exhibit sudden rather than gradual change, as triggered rebuild techniques perform best on sudden concept shifts whereas continuous rebuild approaches work well on both gradual and sudden changes. So the datasets used in this evaluation are the same as the datasets used in the CDBD detection and rebuild experiment from Chapter 6, i.e. 20NG$_{SBF}$, 20NG$_{SIF}$, Reuters$_{SBF}$, Reuters$_{SIF}$, NS$_{SBF}$, NS$_{SIF}$, Spam$_1$ and Spam$_2$.

The general experiment methodology is very similar to the methodology used to evaluate VUR and CRBD in the previous chapter: train the initial classifier on the training data then allow the approach to process the data, noting the average class accuracy obtained by the classifier, and the labels required by the approach. There are however some differences in the methodology used by the approaches when viewed on a more low level, which will be discussed in the next section.

7.2.1.1 Variable Uncertainty with Randomness

VUR will be evaluated on a modified version of the *ActiveClassifier* in the Massive Online Analysis (MOA) (Bifet *et al.*, 2010) framework. The methodology used to evaluate VUR is very similar to the one used to evaluate DVS, the major difference is that VUR processes instances one by one, whereas DVS processes instances in batches. The first step is to train the initial classifier on the training data. An SVM was used in these experiments as they are very suitable on text data (Dumais *et al.*, 1998; Joachims, 1998; Yang & Liu, 1999). After the initial classifier has been trained the unlabelled instances are processed sequentially. When a new unlabelled instance arrives the classifier predicts its label, then makes a decision if that instance should be sampled or not, using the sampling strategy covered in Section 3.4.3.1. If an instance is sampled it is labelled and added to the training window. The error rate, calculated on the sampled instances, is used to adjust the window size.

The sampling strategy is non-deterministic (due to the random component) so VUR will be run ten times and the mean of the ten average class accuracies and fraction of labels used[1] metrics will be considered representative values for the approach. The experiment will be run three times on each dataset with a labelling budget of 0.05, 0.10 and 0.15 to match the labelling budget used by DVS (as the authors do not recommend a particular value for the labelling budget).

7.2.1.2 Confidence Range Batch Detection

CRBD was briefly discussed in Section 3.4.3.2. It is the approach most closely related to CDBD as it is also a triggered rebuild approach with an indicator based on classifier output. The methodology used to evaluate this approach is very similar to the one used to evaluate CDBD. The classifier is first trained on the initial training data. The same classifier will be used as in (Lanquillon, 1999), a single prototype classifier, as this approach requires a classifier which can produce a confidence score for each class, where the

[1]The labelling budget guides the probability of sampling so that, on average, the fraction sampled over the full data stream is equal to b. However, the labels used by VUR can be slightly less than allowed by the labelling budget due to the probabilistic nature of the approach.

scores do not add up to unity[1]. After the initial classifier has been trained, and the confidence range estimated, the unlabelled instances are batched together and presented for classification. CRBD then uses the detection approach detailed in Section 3.4.3.2 to examine each batch of data for a change in concept. However, initial experiments showed that this approach did not work because the initial training data on which the confidence range was estimated had a very different class distribution to the test data (as the initial training data has a balanced class distribution). This problem probably never arose in the evaluation performed in (Lanquillon, 1999) as the datasets used had a fixed class distribution. The approach was modified to overcome this by estimating the range on ten batches after the training data. The classifier is adapted when a change in concept is detected in the same way as CDBD, as exploratory experiments showed that just updating the classifier with data from the place where the detection took place (as specified in (Lanquillon, 1999)) produced a highly skewed training set which led to poor classification accuracy and detection results.

The modified version of CRBD will be run on each of the datasets and the classification accuracy and fraction of labels used will be noted.

[1]The single prototype classifier returns the distance between an unlabelled instance and each class prototype, and the distances do no add up to unity, unlike an SVM. Some early experiments on static text datasets (which includes the experiment used to illustrate TOPSIS in Section 4.3.1) showed that a single prototype classifier achieves a classification accuracy comparable to an SVM.

7.2.2 Results

Both VUR and CRBD attempt to handle concept drift without requiring a fully labelled data stream. The evaluation results will be analysed separately to attempt to determine how well the approaches performed.

7.2.2.1 Variable Uncertainty with Randomness

Table 7.1 shows the average class accuracy obtained by VUR on each of the eight datasets. The table also includes the average class accuracies obtained by DVS on the same datasets to allow a comparison between the two approaches. Each row in the table shows the average class accuracy on that dataset using a given approach and labelling budget. The average class accuracies obtained by VUR shows that the approach is handling concept drift on all datasets. The average class accuracy improves as the labelling budget is increased, just like DVS. Comparing the two sets of average class accuracies shows that the average class accuracies obtained by VUR are comparable to the ones obtained by DVS in Chapter 5.

Table 7.1: Average class accuracy of DVS and VUR for different labelling budgets.

	b = 0.05		**b = 0.10**		**b = 0.15**	
	DVS	**VUR**	**DVS**	**VUR**	**DVS**	**VUR**
20NG$_{SBF}$	0.72	0.71	0.77	0.75	0.76	0.77
20NG$_{SIF}$	0.77	0.73	0.81	0.76	0.82	0.77
Reuters$_{SBF}$	0.71	0.72	0.73	0.75	0.76	0.79
Reuters$_{SIF}$	0.72	0.69	0.78	0.74	0.80	0.76
NS$_{SBF}$	0.83	0.81	0.86	0.84	0.86	0.83
NS$_{SIF}$	0.84	0.75	0.87	0.79	0.87	0.79
Spam$_1$	0.97	0.92	0.97	0.94	0.97	0.95
Spam$_2$	0.93	0.85	0.93	0.89	0.94	0.90

7.2.2.2 Confidence Range Batch Detection

Table 7.2 shows the average class accuracy obtained and fraction of labels used by CRBD on each of the eight datasets. The table also includes the average class accuracies obtained by CDBD-3-5 on the same datasets to allow a comparison between the two approaches. CDBD-3-5 was chosen for this comparison over CDBD-1-1 and CDBD-2-3 as evaluations performed in Appendix E.2 showed that the 3/5 trigger is the best out of the three when both average class accuracy and fraction of labels used are considered. Each row in the table shows the average class accuracy and fraction of labels used on that dataset using a given approach.

Table 7.2: The average class accuracy and fraction of labels used by CRBD and CDBD.

	CRBD		**CDBD-3-5**	
	Acc.	**Labels**	**Acc.**	**Labels**
$20NG_{SBF}$	0.72	0.15	0.76	0.10
$20NG_{SIF}$	0.73	0.04	0.76	0.09
$Reuters_{SBF}$	0.71	0.08	0.75	0.16
$Reuters_{SIF}$	0.74	0.25	0.78	0.19
NS_{SBF}	0.61	0.03	0.83	0.04
NS_{SIF}	0.58	0.01	0.83	0.09
$Spam_1$	0.70	0.22	0.90	0.21
$Spam_2$	0.76	0.41	0.84	0.45

The results show that CRBD obtains an average class accuracy comparable with the one obtained by CDBD on some datasets, yet does very poorly on other datasets. The labels used column shows that CRBD uses considerably less labelled data on many of the datasets. Both of these inconsistencies can be explained by the CRBD detection points.

Table 7.3 shows the instance number in the data stream where the notional perfect detector and CRBD detected changes in concept. The perfect detector triggers a classifier rebuild at each of the known concept drift points, i.e. one rebuild at the beginning of the second concept (C2) and one in the beginning of the third concept (C3) on each dataset[1] (it does not detect a change in C1, as this is the same concept as the one the initial classifier was trained on). The detection points listed for CRBD are the ones closest to the perfect detector detection points in that concept. CRBD misses a detection on three datasets, these are marked with a dash.

Table 7.3: The detection points for CRBD and the perfect detector.

	C2		**C3**	
	Perfect	**CRBD**	**Perfect**	**CRBD**
$20NG_{SBF}$	2300	2300	4400	4300
$20NG_{SIF}$	4800	-	9400	9200
$Reuters_{SBF}$	1600	1600	2900	-
$Reuters_{SIF}$	5500	5800	10700	9400
NS_{SBF}	8200	1700	16200	3500
NS_{SIF}	15200	6300	30200	-

On the $20NG_{SBF}$ dataset CRBD detects a change in concept exactly where it should in C2 and before the change has taken place in C3. A similar detection pattern is found on the $Reuters_{SIF}$ dataset. A slightly premature detection by CRBD might seem acceptable, however this means that the classifier is rebuilt on training data from both the new and old concept, which may lead to lower classification accuracy and worse detections. CRBD misses a detection on the $20NG_{SIF}$, $Reuters_{SBF}$ and the NS_{SIF} datasets, which leads to a smaller number of labels being used, but worse accuracy when compared to CDBD-3-5. However, the accuracy obtained by CRBD

[1]The Spam datasets are not included in this analysis as the drift points are unknown in those datasets.

on some datasets is not much lower than CDBD-3-5 even though CDBD-3-5 is better at detecting changes in concept. This is because on those datasets the concept changes again before the benefit obtained by rebuilding the classifier is consolidated. This is more obvious on small datasets where the time between changes in concept is smaller, and imbalanced datasets where the time it takes to rebuild the classifier is higher (as the rebuild mechanism attempts create a balanced training set). However, the classification accuracy difference between CRBD and CDBD-3-5 on larger datasets, like NS$_{SBF}$ and NS$_{SIF}$, and highly imbalanced datasets like Spam$_1$ and Spam$_2$ shows the cost of bad detections.

Based on Table 7.2 it might seem like CDBD-3-5 is a better concept drift handling algorithm than CRBD, however, this is only the case if average class accuracy is the only evaluation metric used. The next section will therefore compare CRBD, CDBD and other concept drift handling approaches on an evaluation metric which combines both average class accuracy and labelling cost into one measure.

7.3 Evaluating Concept Drift Handling Approaches using Multiple Criteria

This experiment aims to evaluate the approaches listed in Section 7.1 based on two criteria: average class accuracy and fraction of labels used. Because two criteria are being considered, the suitability of an approach is fully dependent on the weight placed on each criterion. Evaluating the approaches using TOPSIS requires the domain dependant costs (such as misclassification and labelling costs) to be expressed as TOPSIS weights. This can be achieved by first expressing the costs as a ratio. For example, in the context of this experiment a ratio of 3 : 1 implies that when comparing two approaches one approach would have to obtain 3% higher average class accuracy than the other for them to be equal, if the other approach uses 1% less labelled data. The ratio can then be written as a fraction and used as TOPSIS weights. In the previous example the TOPSIS weights would be 0.75 and 0.25 respectively.

The experiment in this section will not evaluate the approaches based on a particular set of costs, but will instead use the TOPSIS weights to simulate many different labelling cost scenarios. For example, weights heavily skewed towards labelling cost simulates a scenario where labelled data is very expensive to obtain, and conversely if the weights are heavily skewed towards average class accuracy then a high average class accuracy is desirable and the fraction of labelled data used is considered unimportant.

7.3.1 Methodology

The first step in these experiments is to gather the two evaluation metrics for each approach, in this case average class accuracy and fraction of labels used. The evaluation metrics will be collected from various experiments performed so far. The data for no update, sliding window, random sampling and DVS will be taken from Section 5.3, WRABD and CDBD from Section 6.3.3, and the two state of the art approaches, VUR and CRBD, from Section 7.2.2.

TOPSIS will then be applied to the results with equal weights for average classification accuracy and fraction of labels used, creating a TOPSIS score for each combination of dataset and approach. This score will then be used to rank each dataset from best to worst. The average rank will be calculated and used in a Friedman test to establish whether or not there is a statistically significant difference in the average ranks. If a statistically significant difference is found then the Nemenyi test will be used to ascertain which pair, or pairs, of approaches show a statistically significant difference in performance. However, this only provides a conclusion predicated on the assumption that the average class accuracy and labelling cost are equally important. The final part of the experiment is therefore to examine how the average ranks change under different weighting scenarios. This will be achieved by varying the labelling cost weight from zero to one, while setting the accuracy weight to $1 - \text{labellingCostWeight}$. The two weights are linearly correlated as TOPSIS requires the weights to sum to one. Table 7.4 lists the weights that will be used in this experiment and shows how they relate to a labelling cost scenario.

7.3.2 Results

The TOPSIS scores presented in Table 7.5 are calculated using the procedure described in Section 4.3.1. The TOPSIS scores are based on equal weights for average class accuracy and labelling cost. The approaches examined span from no update (abbreviated as "NU") to the fixed size sliding window (abbreviated as "SW").

Table 7.4: Weights for the TOPSIS multi-criteria experiment.

Labelling cost	Labelling weight	Accuracy weight
Low	0.0	1
⋮	0.1	0.9
	0.2	0.8
	0.3	0.7
	0.4	0.6
	0.5	0.5
	0.6	0.4
	0.7	0.3
	0.8	0.2
⋮	0.9	0.1
High	1.0	0.0

The TOPSIS scores suggests that there are two distinct groupings of approaches. The two baseline approaches which require a fully labelled data stream, WRABD and sliding window, obtained low TOPSIS scores as they are heavily penalised for using all the labels. The other approaches seem to fare a bit better, earning a more respectable TOPSIS score ranging from 0.55 to 0.96.

Table 7.6 simplifies the comparison by ranking the approaches on each dataset, based on their TOPSIS score. The last row gives the average rank of each approach. The average ranks show that given equal weights, RS-0.05 is the best approach, closely followed by DVS-0.05. This result makes sense as this choice of weights favours approaches which use little or no data. The fact that RS-0.05 is better than DVS-0.05 can be easily explained, as the results from Chapter 5 showed that random sampling is often better than DVS on a 0.05 budget, while DVS is better when the label budget is above 0.05.

Table 7.5: TOPSIS scores for continuous and triggered rebuild approaches using equal weights.

	NU	**Random sampling**			**DVS**			**VUR**			**CDBD**			**CRBD**	**WRABD**	**SW**
		0.05	**0.10**	**0.15**	**0.05**	**0.10**	**0.15**	**0.05**	**0.10**	**0.15**	**1/1**	**2/3**	**3/5**			
$20NG_{SBF}$	0.94	0.94	0.90	0.85	0.94	0.90	0.85	0.93	0.90	0.85	0.89	0.90	0.90	0.84	0.04	0.06
$20NG_{SIF}$	0.90	0.94	0.90	0.85	0.93	0.90	0.85	0.93	0.89	0.85	0.72	0.84	0.90	0.94	0.07	0.10
$Reuters_{SBF}$	0.91	0.92	0.89	0.85	0.92	0.89	0.85	0.93	0.89	0.85	0.84	0.84	0.83	0.90	0.08	0.09
$Reuters_{SIF}$	0.89	0.92	0.89	0.85	0.91	0.89	0.84	0.91	0.88	0.84	0.67	0.77	0.81	0.75	0.05	0.11
NS_{SBF}	0.89	0.95	0.90	0.85	0.95	0.90	0.85	0.95	0.90	0.85	0.88	0.94	0.96	0.89	0.09	0.10
NS_{SIF}	0.89	0.95	0.90	0.85	0.94	0.90	0.85	0.93	0.89	0.85	0.83	0.84	0.91	0.88	0.10	0.11
$Spam_1$	0.93	0.95	0.90	0.85	0.95	0.90	0.85	0.95	0.90	0.85	0.55	0.76	0.79	0.76	0.09	0.10
$Spam_2$	0.90	0.95	0.90	0.85	0.94	0.90	0.85	0.93	0.90	0.85	0.56	0.58	0.55	0.58	0.04	0.10

Table 7.6: Ranks for continuous and triggered rebuild approaches based on TOPSIS scores using equal weights.

	NU	**Random sampling**			**DVS**			**VUR**			**CDBD**			**CRBD**	**WRABD**	**SW**
		0.05	**0.10**	**0.15**	**0.05**	**0.10**	**0.15**	**0.05**	**0.10**	**0.15**	**1/1**	**2/3**	**3/5**			
$20NG_{SBF}$	1	3	5	12	2	7	13	4	6	11	10	9	8	14	16	15
$20NG_{SIF}$	7	1	6	10	3	8	11	4	9	12	14	13	5	2	16	15
$Reuters_{SBF}$	4	2	8	11	3	7	10	1	6	9	12	13	14	5	16	15
$Reuters_{SIF}$	4	1	5	8	2	6	9	3	7	10	14	12	11	13	16	15
NS_{SBF}	10	3	6	12	2	7	13	4	8	14	11	5	1	9	16	15
NS_{SIF}	8	1	5	10	2	6	11	3	7	12	14	13	4	9	16	15
$Spam_1$	4	1	5	8	2	6	9	3	7	10	14	12	11	13	16	15
$Spam_2$	4	1	5	8	2	6	9	3	7	10	13	11	14	12	16	15
AVG	5.25	1.63	5.63	9.88	2.25	6.63	10.63	3.13	7.13	11.00	12.75	11.00	8.50	9.63	16.00	15.00

Another interesting result is that no update does so well. It seems that what it lacks in accuracy it makes up for in label frugality. This also emphasises one of the most important aspects of handling concept drift in the context of expensive labels: that the appropriateness of an approach is highly dependent on the relative importance placed on average class accuracy and fraction of labels used. It could be argued that in the equal weights scenario the no update approach is very attractive as its TOPSIS scores are comparable to RS-0.05 and DVS-0.05, but unlike those approaches it does not need to be continuously adapted.

The Friedman test shows that there is a statistically significant difference in average ranks in Table 7.6 (at the 95% confidence level). Table 7.7 shows the absolute difference in average rank between all of the approaches. The statistically significant differences, based on the Nemenyi test at the 95% confidence level are underlined. The table shows that, DVS-0.05, RS-0.05 and VUR-0.05 are statistically significantly better than both WRABD and sliding window. DVS-0.05 is also statistically significantly better than CDBD-1-1, CDBD-2-3 and the continuous rebuild approaches using a labelling budget of 0.15 (i.e. DVS-0.15, RS-0.15 and VUR-0.15). The results indicate that when average class accuracy and fraction of labels used are weighted equally, approaches which use a small amount of labelled data, yet obtain a sizeable accuracy increase over no update (like RS-0.05 and DVS-0.05) are the best approaches. However, these results only show the ranks when equal weights are used. This chapter deals with evaluating the concept drift handling approaches over a number of labelling cost scenarios, so the remainder of

Table 7.7: Absolute difference in rank between concept drift handling approaches.

	DVS			**Random sampling**			**VUR**			**CDBD**			**CRBD**	**WRABD**	**SW**
	0.05	0.10	0.15	0.05	0.10	0.15	0.05	0.10	0.15	1/1	2/3	3/5			
NU	3.63	0.38	4.63	3.00	1.38	5.38	2.13	1.88	5.75	7.50	5.75	3.25	4.38	10.75	9.75
DVS-0.05		4.00	8.25	0.63	5.00	9.00	1.50	5.50	9.38	11.13	9.38	6.88	8.00	14.38	13.38
DVS-0.10			4.25	3.38	1.00	5.00	2.50	1.50	5.38	7.13	5.38	2.88	4.00	10.38	9.38
DVS-0.15				7.63	3.25	0.75	6.75	2.75	1.13	2.88	1.13	1.38	0.25	6.13	5.13
RS-0.05					4.38	8.38	0.88	4.88	8.75	10.50	8.75	6.25	7.38	13.75	12.75
RS-0.10						4.00	3.50	0.50	4.38	6.13	4.38	1.88	3.00	9.38	8.38
RS-0.15							7.50	3.50	0.38	2.13	0.38	2.13	1.00	5.38	4.38
VUR-0.05								4.00	7.88	9.63	7.88	5.38	6.50	12.88	11.88
VUR-0.10									3.88	5.63	3.88	1.38	2.50	8.88	7.88
VUR-0.15										1.75	0.00	2.50	1.38	5.00	4.00
CDBD-1-1											1.75	4.25	3.13	3.25	2.25
CDBD-2-3												2.50	1.38	5.00	4.00
CDBD-3-5													1.13	7.50	6.50
CRBD														6.38	5.38
WRABD															1.00

this section will therefore look at how the average ranks change when the weights are varied.

Table 7.8 shows the average rank of each approach while varying the labelling cost weight (the average class accuracy weight moves correspondingly but does not need to be shown as it is linearly correlated with the labelling cost weight). Each row shows the average rank (on all datasets) obtained by an approach for a given labelling cost weight. The first row shows the ranks when the amount of labelled data used is not considered (the assumption made by most concept handling techniques). The opposite situation is when the labelling cost weight is set to one, i.e. a scenario where the accuracy is unimportant and the only consideration is the amount of labelled data used. The last row shows the average of the average ranks, which gives an indication of how well an approach does over all of the different weight values.

Figure 7.1 shows a visual representation of Table 7.8 where the average rank over various labelling cost weights is plotted.

The graph shows that approaches with a large labelling budget have the lowest average rank when the TOPSIS scores are calculated using weights heavily skewed towards accuracy. Approaches which use less labelled data receive a more competitive average rank as the weights are adjusted to place more importance on the labelling cost. When the label weight is set to one (and therefore average class accuracy weight is zero) the approaches with the same labelling budget converge on the same average rank, which makes sense as they have the same budget and their average rank is solely based on the labelling budget.

Table 7.8: The average rank of each approach while varying the labelling cost weight.

Labelling weight:	**NU**	**DVS**			**Random sampling**			**VUR**			**CDBD**			**CRBD**	**WRABD**	**SW**
		0.05	**0.10**	**0.15**	**0.05**	**0.10**	**0.15**	**0.05**	**0.10**	**0.15**	**1/1**	**2/3**	**3/5**			
0	15.75	9.13	3.63	2.88	11.75	7.38	5.50	13.75	9.88	7.38	7.38	7.75	9.00	14.00	9.25	1.63
0.1	14.25	6.00	2.63	3.25	8.63	5.50	5.38	11.13	7.88	6.25	8.75	6.88	6.50	13.38	15.63	14.00
0.2	12.63	4.25	3.00	6.13	6.25	4.75	7.38	8.75	6.13	8.00	10.38	8.00	6.50	12.88	16.00	15.00
0.3	11.38	1.88	3.63	8.75	2.88	5.00	9.50	4.75	6.13	10.13	11.88	10.13	7.63	11.38	16.00	15.00
0.4	7.88	1.50	5.00	9.38	2.13	6.13	10.13	3.38	6.63	10.50	12.50	10.88	8.38	10.63	16.00	15.00
0.5	5.25	1.63	5.63	9.88	2.25	6.63	10.63	3.13	7.13	11.00	12.75	11.00	8.50	9.63	16.00	15.00
0.6	3.25	2.00	6.25	9.88	2.50	7.13	10.63	3.38	7.63	11.00	12.75	11.25	8.88	8.50	16.00	15.00
0.7	1.25	2.63	6.25	9.88	3.13	7.13	10.63	4.00	7.63	11.00	12.75	11.50	8.88	8.38	16.00	15.00
0.8	1.00	2.88	6.25	9.88	3.38	7.13	10.63	4.25	7.63	11.00	12.75	11.50	9.13	7.63	16.00	15.00
0.9	1.00	2.88	6.25	9.88	3.38	7.13	10.63	4.25	7.63	11.00	12.75	11.50	9.13	7.63	16.00	15.00
1	1.00	3.50	7.00	10.50	3.50	7.00	10.50	3.50	7.00	10.50	12.75	11.50	9.13	7.63	15.50	15.50
AVG	6.78	3.48	5.05	8.20	4.52	6.44	9.23	5.84	7.39	9.80	11.58	10.17	8.33	10.15	15.31	13.74

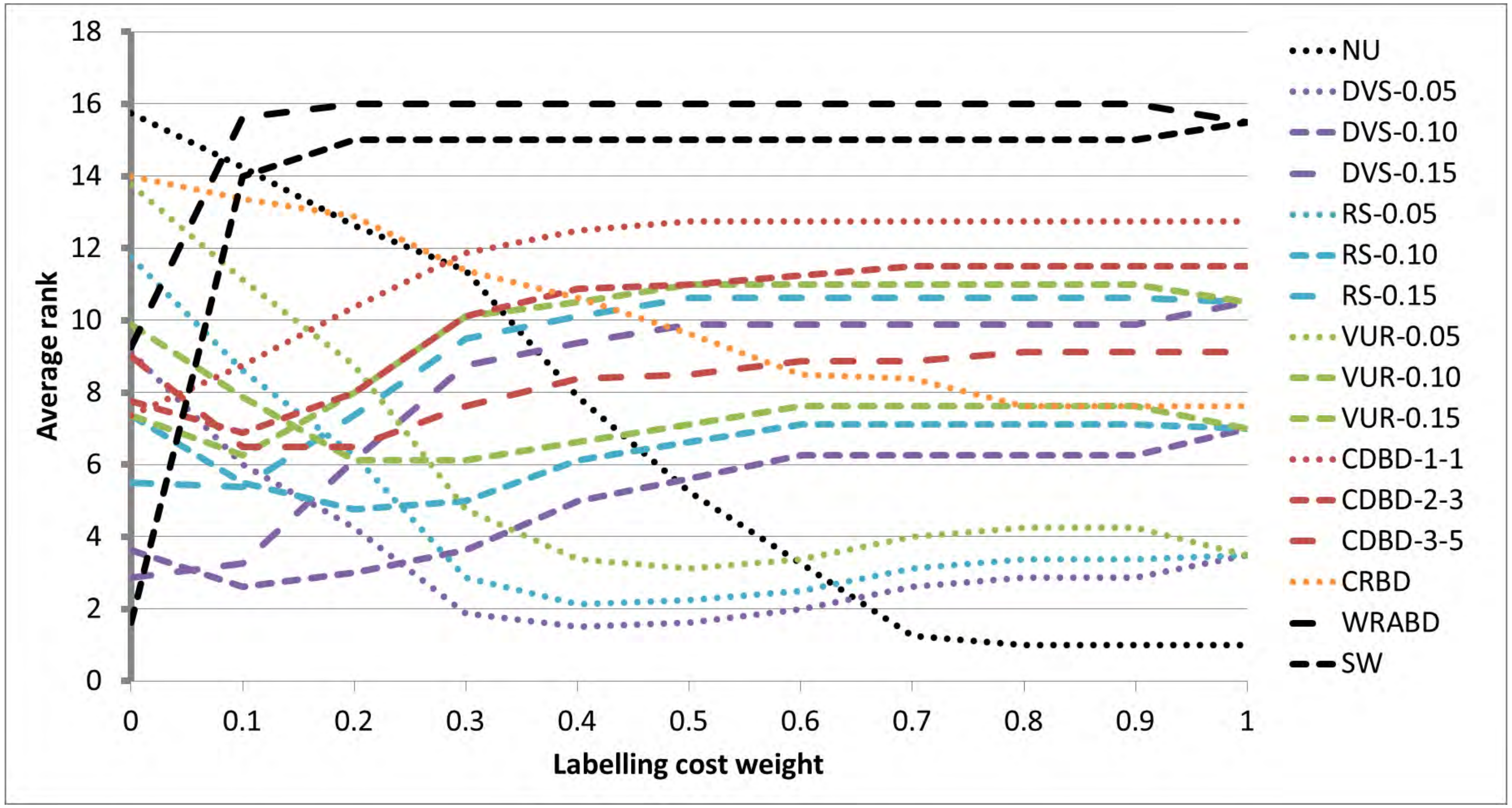


Figure 7.1: The average rank over various labelling cost weights.

The graph contains a large number of series, making it cumbersome to read. However, all the approaches have been included in the graph for completeness, but the focus will be placed on the approaches with the lowest average rank. No update is the best concept drift handling approach when the cost of acquiring labelled data is very high. In this experiment no update is the best concept drift handling approach when the labelling cost weight is above 0.65. Conversely, sliding window is the best approach when there is no labelling cost (the labelling cost weight is zero). However, in the labelling weight cost range of 0.05 to 0.65, DVS is the best concept drift handling approach as long as the correct labelling budget parameter is used. In the case of this experiment the optimal labelling budget is roughly 0.05 for a labelling cost weight between 0.25 and 0.65. When the labelling cost weight is between 0 and 0.25 then a labelling budget of 0.10 is most apt[1].

The last part of this analysis will attempt to provide a few high level conclusions based on the results from this section and Chapters 5 and 6. The first conclusion is that unless the cost of acquiring labelled data is very high it is worth attempting to handle concept drift. The results also indicate that even a small amount of labelled data can drastically improve classification accuracy over not updating the classifier at all. This is particularly true for DVS which shows a lot of potential for handling concept drift with a small amount of labelled data. The experiment results show that DVS obtained the best trade-off between average class accuracy and labels used for nearly all

[1]DVS-0.15 is better than the suggested DVS-0.10 at a labelling cost weight of about 0.05, however, the difference is so small that it simplifies the parameterisation of DVS to recommend the use of DVS-0.10 even in this case.

labelling weight scenarios tested. The other continuous rebuild approaches, random sampling and VUR also performed well with average ranks very comparable to the average rank obtained by DVS. This suggests that the act of continuously adapting the classifier with recent training data is the key property which makes sliding window approaches so successful, yet the idea of a sampling strategy should not be discarded as the results from Chapter 5 show that using a targeted sampling strategy gives slightly better results than random sampling.

The results also show that triggered rebuild approaches can be used to handle concept drift unless the labelling weight is very high (in the case of Figure 7.1 less than 0.40). CDBD-3-5 provides the best trade-off between average class accuracy obtained and fraction of labels used out of the triggered rebuild approaches, when the labelling cost justifies handling concept drift.

The comparison between continuous and triggered rebuild approaches raises an interesting question about the strengths and weaknesses of each type of approach. There might not be enough concept drift handling approaches used in these experiments to reliably extrapolate a more general conclusion about continuous and triggered rebuild approaches, however, some important points which are likely to be broadly applicable will be highlighted, as the approaches evaluated are representative of their respective types.

Continuous rebuild approaches benefit from being able to handle small, gradual change as it occurs, whereas triggered rebuild approaches need to wait until the change in concept accumulates enough to trigger a rebuild.

Another strength shared by continuous rebuild approaches is that they are able to handle conditional, feature and dual change, whereas triggered rebuild approaches based on a signal not derived from labelled data are only able to handle feature and dual change.

However, this does not mean that triggered rebuild approaches should not be considered. In the case where changes in concept occur suddenly and infrequently triggered rebuild approaches have the potential to use appreciably less labels than a continuous rebuild approach. This is supported by the CDBD-3-5 results on the NS$_{SIF}$ and NS$_{SBF}$ datasets.

Overall continuous rebuild approaches seem to show the most potential, particularly DVS, which not only handles concept drift, but is also conceptually simple and only requires the setting of one parameter.

7.4 Conclusion

The first part of this chapter evaluated VUR and CRBD, two state of the art approaches which both aim to handle concept drift using a limited amount of labelled data. VUR is a continuous rebuild approach, which, like DVS uses a sliding window and active learning to handle concept drift. CRBD is a triggered rebuild approach, which, like CDBD constructs an indicator from the output of the classifier and rebuilds the classifier when a sizeable change in the indicator values takes place.

The evaluation showed that VUR was capable of handling concept drift on a variety of datasets, while CRBD could handle concept drift on some datasets, but missed detections on others.

The second part of the chapter dealt with comparing both continuous and triggered rebuild approaches. The evaluation so far had been limited to a comparison based on either average class accuracy or fraction of labels used, which prevents the comparison between approaches which use an unequal amount of labelled data. In this comparison the two evaluation metrics were combined into one evaluation metric using TOPSIS.

The results showed that the best approach was highly dependent on what accuracy and labelling cost weights were used in the TOPSIS score calculation. When equal weights were used approaches which used a small amount of labelled data, like DVS-0.05 and RS-0.05 were statistically significantly better than both the fully labelled data stream benchmark approaches WRABD and sliding window, and the triggered rebuild approaches CDBD-1-1 and CDBD 2-3.

The weights were adjusted to investigate how the best approach changes as the weights change. When the labelling cost was above 0.65 no update was the best approach, and conversely when the labelling cost was close to one, the fixed size sliding window approach was the best approach. DVS was the best approach when the labelling cost weight was in the range 0.05 to 0.65. In fact, continuous rebuild approaches with a low labelling budget were better than the triggered rebuild approaches in most cases. However, triggered rebuild approaches showed potential on large datasets with infrequent changes in concept, such as NS_{SIF} and NS_{SBF}.

The next chapter will summarise the findings and contributions from this, and the proceeding chapters.

Chapter 8

Conclusion

Machine learning has been very successful in a wide array of prediction problems, yet applying it to large data streams can be complicated by concept drift. There is a growing body of research dealing with how concept drift can be handled but most of it is underpinned by the assumption that the data stream is fully labelled, or that it is possible to obtain the true label of an instance shortly after classification at a negligible cost. The aim of this thesis is to examine ways in which concept drift can be handled when this pivotal assumption does not hold. More specifically, when labelled data can be obtained, but the cost of doing so is high.

Two distinct ways of dealing with this can be found in the literature: continuous rebuild approaches which use semi-supervised learning, and triggered rebuild approaches. Decision Value Sampling (DVS), a semi-supervised approach was shown to be capable of handling concept drift when only a small fraction of the instances in the data stream are labelled. Confidence Distribution Batch Detection (CDBD), is a triggered rebuild approach capable of detecting changes in concept without requiring labelled data and can be used in conjunction with an adaptation mechanism to handle concept drift.

Finally, an empirical evaluation of both continuous and triggered rebuild approaches was also performed. It, coupled with the results from the previous evaluations showed that semi-supervised rebuild approaches with a low labelling budget were better than the triggered rebuild approaches in most cases, but triggered rebuild approaches should be considered on large datasets.

The remainder of this chapter will be used to summarise how this thesis contributes towards concept drift research and suggest interesting areas for future study.

8.1 Summary of Contributions and Achievements

The fact that most concept drift handling approaches assume the availability of labelled data has been referred to throughout this thesis. The importance of developing concept drift handling approaches that do not rely on that assumption has also been highlighted. The purpose of this section is to summarise how this thesis is contributing towards that goal, and more generally towards concept drift research as a whole. Specific contributions include:

- *A process for generating text datasets exhibiting highly controllable concept drift* – Section 4.1 described some of the issues associated with naturally occurring data exhibiting concept drift and motivated the need for a process which can generate datasets exhibiting controlled concept drift. The framework described allows for the generation of drift induced datasets where the change type, change cause and class distribution can be controlled while maintaining some of the nuances of real data. The method was applied to text documents, but it is likely that the general procedure is versatile enough to be applied to other types of data, such as financial data.

- *A novel continuous rebuild concept drift handling approach* – Chapter 5 introduced DVS, a continuous rebuild approach that combines active learning and a sliding window to handle concept drift. The evaluation described in that chapter showed that DVS was capable of handling both conditional and feature change, both gradual and sudden concept drift and both balanced and imbalanced class distributions. This suggests that DVS is a good choice of concept drift handling approach when little is known about the characteristics of the concept drift that will be encountered. The evaluation also showed that most of the gains in classification accuracy were obtained at the lower labelling budgets.

- *The CDBD signal, a novel way of measuring feature change* – Chapter 6 showed how an indicator believed to be correlated with changes in concept can be created by measuring the difference in a classifier's confidence output. The CDBD signal was shown to be well correlated with changes in concept, both by comparing the signal values before and after changes in concept, and by calculating the detection accuracy of a trigger applied to the signal.

- *A novel triggered rebuild concept drift handling approach* – Chapter 6 also showed how the CDBD indicator can be combined with a trigger and adaptation mechanism to handle changes in concept. CDBD only requires labelled data when a change in concept is detected, and has been shown to accurately detect changes in concept in text data streams. This represented a sizeable saving in labelled data, particularly on large datasets with infrequent changes in concept.

- *A methodology for comparing concept drift handling approaches in the context of expensive labels* – Chapter 7 showed how TOPSIS can be used to evaluate concept drift handling approaches by combining average class accuracy and fraction of labels used into one evaluation metric. The usefulness and versatility of this approach was demonstrated on a collection of representative concept drift handling approaches. Arguably the most important attribute of an MCDA approach like TOPSIS is that it allows the relative importance of the criteria to be adjusted. This is of particular interest in the context of expensive labels as our evaluation showed that the appropriateness of a given approach was highly dependant on the weight given to the labelling cost.

- *An empirical evaluation of representative continuous and triggered rebuild concept drift handling approaches* – Chapter 7 also compared representative continuous and triggered rebuild approaches. The results showed that continuous rebuild approaches are better than a triggered rebuild approaches in most circumstances, but triggered rebuild approaches might be more appropriate on large datasets which exhibit infrequent feature change.

8.2 Open Problems and Future Work

The contributions listed in the previous section are centred around the problem of expensive labels in concept drift handling. However, the exploration of the problem unearthed more questions than can be addressed in one thesis. This section aims to list some interesting directions for future research in the area of concept drift handling with expensive labels. Most of the suggested directions have been discovered due to the work on DVS and CDBD, but are largely applicable to most concept drift handling approaches attempting to handle concept drift using a limited amount of labelled data.

8.2.1 Investigating Other Uses for Change in Concept Indicators

In Chapter 6 the CDBD indicator was shown to be able to distinguish between two concepts. It was also shown that the CDBD indicator combined with a trigger and adaptation mechanism could handle concept drift. However, the CDBD indicator might also be useful outside of a triggered rebuild framework, for example to dynamically adjust algorithm parameters.

One way the CDBD signal could be used in this manner is to adjust the labelling budget in DVS and Variable Uncertainty with Randomness (VUR) so that more instances are sampled when the signal suggests that the concept is changing, and less (or none) when the concept is believed to be stable. This change to the sampling strategy could result in a sizeable reduction in the number of labels used by the approach, particularly if changes in concept occur infrequently.

The CDBD signal need not be confined to just continuous rebuild approaches using active learning. In fact, the CDBD signal could be used in a similar way to how the error rate is used in many common concept drift handling approaches. For example, it could be used to adjust the window size of a sliding window or form a part of an ensemble weighting scheme.

8.2.2 Decreasing the Adaptation Lag

CDBD often takes an excessively large number of batches before an updated balanced training set can be created, during which the out-of-date classifier is still being used. This results in a noticeable lag between when a change in concept is flagged and when the classification accuracy improves.

A simple modification to CDBD would be to start incrementally adapting the classifier straight after a change in concept is detected. A classifier agnostic way to achieve this is to use a sliding window approach which discards old data in the training window until all of the instances in the training window have been replaced. One of the large advantages of incrementally updating the classifier is that the classifier can immediately start learning the new concept, rather than having to wait until the new training window is formed. However, any changes to the adaptation approach must ensure that the ability to handle data with a high class imbalance is not hampered.

8.2.3 Handling Class Imbalance

DVS and CDBD handle class imbalance using a form of undersampling, as they discard instances from the majority class to balance the class distribution in the training window. An interesting direction for future work would be to incorporate a more sophisticated class imbalance approach, such as SMOTE (Chawla *et al.*, 2002), which undersamples the majority class and artificially generates instances of the minority class. Improving the way class imbalance is handled might enable the training window size to be reduced without the loss of classification accuracy, which, in turn should decrease the detection lag discussed in the previous section and more importantly reduce the amount of labelled data required (as a smaller training window requires less labelled data). There are concept drift approaches which are specifically designed to handle class imbalance, however, to the best of our knowledge, DVS and CDBD are the only concept drift handling approaches which are designed to handle concept drift in an imbalanced data stream which is not fully labelled.

Another interesting direction for future work is to use a sampling strategy to attempt to balance the dataset. *Positive biased sampling* (Lindstrom *et al.*, 2010b) seems like a good starting point. Positive biased sampling aims to identify two types of instances: those which will refine the decision boundary and those which will balance the class distribution. The sampling strategy attempts to sample an equal number of instances of both types by sampling instances close to the decision boundary and instances which are likely to be of the minority class (based on their distance and orientation to the decision boundary). The approach is designed for static datasets, but could probably be adapted for use in a concept drift handling approach like DVS.

8.2.4 Improving the Selection Strategy

Chapter 6 introduced a sampling strategy which selects a number of instances from each batch based on their proximity the classifier's decision boundary. There are however improvements which could be made to how the instances are selected.

One interesting modification to the sampling strategy would be to incorporate instance similarity into the sampling process. This modification is based on the intuition that instances being sampled based on distance to the decision boundary might be very similar, and the refinement of the decision boundary might benefit more from instances which differ from the already sampled instances. The approach presented in (Zliobaite, 2011) incorporates instance similarity into the selection strategy, but requires a fully labelled data stream.

Another way to ensure more diversity among the sampled instances is to first select a number of instances near the decision boundary, then apply clustering to those instances and sample no more than one instance per cluster. Hu *et al.* (2010) introduced a sampling strategy like this, but only applied it to a static dataset. An interesting direction for future work would be to apply a similar selection strategy to a concept drift handling approach like DVS.

8.3 Final Thoughts

This section will provide some final thoughts on the thesis as a whole, and some of the lessons learned. An early goal was to create a unified and comprehensive approach to handling concept drift in the context of expensive labels. It turned out to be an elusive goal as the appropriateness of a given approach was highly dependent on the specifics of the classification task at hand. One obvious example is how the appropriateness of an approach varies depending on the relative importance placed on obtaining a high classification accuracy and using as little labelled data as possible. If the classification task demands a high classification accuracy then a sliding window seems to be the most appropriate approach, conversely if a low label usage is the most important criterion, then a continuous rebuild approach with a low labelling budget, or even not updating the classifier, is the best approach.

Prohibitively expensive labels is one of the main assumptions made in this thesis, however every stage of the design, development and evaluation of a concept drift handling approach is predicated on assumptions about the classification task the approach will be applied to. One of the most fundamental task dependant decisions is what type of concept drift handling should be used, instance selection, model parameters or ensembles. Implementations of all three types have shown to be capable of handling concept drift and each type has inherit advantages and disadvantages associated with it. So the choice of high level concept drift handling approach comes down to a careful balancing of the positive and negative aspects of that particular type of approach.

DVS and CDBD are both types of instance selection approaches. Instance selection was chosen as it is classifier agnostic and not overly computationally expensive. However, if interoperability is not important then a concept drift handling approach can be developed exploiting properties of specific classifiers, such as instance weighting an SVM or pruning out-of-date branches of decision trees. On the other hand, if computational cost is not an issue, for example if the approach is run on a powerful server, or distributed over multiple servers, then an ensemble based approach might be more suitable.

Similar task-dependant design decisions were made throughout the thesis. Some by choice, and some by necessity due to the data the approaches were being evaluated on. DVS was designed to be robust and handle most concept drift types and causes, whereas CDBD was specifically focused on infrequent feature change. Neither approach considered re-occurring concepts (which could potentially save a lot of labelled data), but they both handle noisy, balanced and imbalanced data.

An example of a design decision in development phase is the choice of classifier. DVS and CDBD both use an SVM as they have been shown to be suitable for many classification tasks, including text classification. Yet other classifiers could also have been considered, to show the versatility of DVS and CDBD.

Most of the evaluation decisions were focused around what data should be used to evaluate the two approaches. It was considered important to ground the evaluations in a domain which is likely to experience the issues the thesis addresses, such as concept drift and expensive labels. Information filtering was identified as a suitable domain, however, other domains such as the financial domain could also have been explored. In fact, additional evaluations on a larger number, and more varied datasets, would help to further validate the results of the experiments performed in this thesis.

Another evaluation-related issue shared by both DVS and CDBD is the reliance on well tuned algorithm parameters. DVS is intentionally simple, requiring only one parameter to be set, the labelling budget. It should however be noted that in many cases the labelling budget is dictated by what the business can afford. CDBD on the other hand has more parameters to set, including the bins to use, the detection threshold, and the trigger. CDBD also has no way of predicting how much labelled data it will require before it is used, which might make it unsuitable when the labelling requirement of the approach needs to be known before deployment. The use of algorithm parameters give the approaches some flexibility, but also complicates their use, as a separate, fully labelled dataset might be needed to empirically establish the optimal parameter value(s) in some cases.

The final evaluation issue worth raising is the range of approaches that

DVS and CDBD were compared to. The benchmark approaches which use a fully labelled data stream seem appropriate for comparison, even though the inclusion of an ensemble approach might also have been interesting. The partially labelled data streams approaches that DVS and CDBD were compared to were, in our opinion, the most appropriate approaches. Nevertheless, comparing the DVS and CDBD to more partially labelled data streams approaches would further strengthen the conclusions drawn from this work.

Concept drift handling in the context of expensive labels is still a budding area of research which warrants further exploration. Our evaluations have shown that, although DVS and CDBD are predicated on some assumptions, both very capable concept drift handling approaches when those assumptions are met and successfully address the task of handling concept drift when labels are expensive. The author is also of the opinion that both continuous and triggered rebuild approaches can deal with the problem of prohibitively expensive labels, with the choice of approach being dependant on the specifics of the classification task at hand. Hopefully this work can serve as a stepping stone for further research in this very promising area of concept drift handling.

Appendix A

Notation

x: an instance

X: a set of instances

x_i: instance i in a dataset

x_{ij}: feature j of instance i

N: number of instances

d: number of features

y: an output (referred to as a class or label in a classification task)

Y: the set of all outputs

y_i: class i in set Y

(x, y): a labelled instance

V: vocabulary of a collection of documents

$P(x)$: the probability of x

$P(y|x)$: the probability of y given x

μ: the mean

σ: the standard deviation

Appendix B

Abbreviations

AI	Artificial Intelligence	p. 1
AL	Active Learning	p. 52
BOW	Bag-Of-Words	p. 27
CDBD	Confidence Distribution Batch Detection	p. 6
CRBD	Confidence Range Batch Detection	p. 147
CRISP-DM	CRoss-Industry Standard Process for Data Mining	p. 12
DF	Document Frequency	p. 27
DT	Decision Tree	p. 16
DVS	Decision Value Sampling	p. 6
FLORA	Floating Rough Approximation	p. 41
IDF	Inverse Document Frequency	p. 27
KDD	Knowledge Discovery in Databases	p. 12
k-NN	*k* Nearest Neighbour	p. 19

MCDA	Multi-criteria Decision Analysis	p. 81
ML	Machine Learning	p. 1
MOA	Massive Online Analysis	p. 151
SVM	Support Vector Machine	p. 20
TDM	Term Document Matrix	p. 27
TF	Term Frequency	p. 27
TF-IDF	Term Frequency - Inverse Document Frequency	p. 27
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution	p. 83
VSM	Vector Space Model	p. 26
VUR	Variable Uncertainty with Randomness	p. 147
WEKA	Waikato Environment for Knowledge Analysis	p. 78
WRABD	Window Resize Algorithm for Batch Data	p. 133

Appendix C

Additional Material for Chapter 5

C.1 DVS Accuracy Over Time Graphs

This section contains the omitted average accuracy over time graphs from Chapter 5. For completeness it also include the three graphs which were presented previously in Section 5.3.

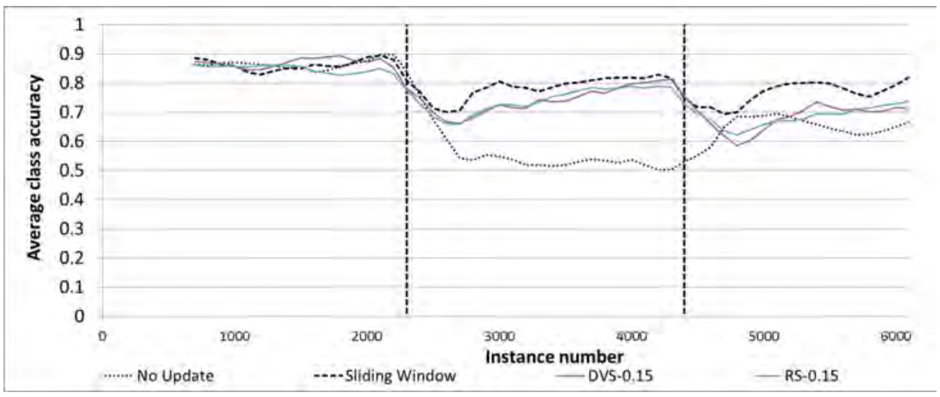


Figure C.1: DVS-0.15 and RS-0.15 average class accuracy over time on the $20NG_{SBF}$ dataset.

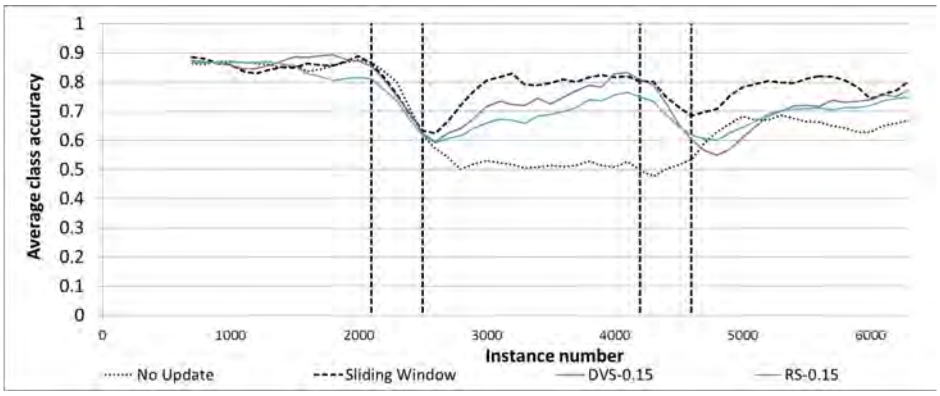


Figure C.2: DVS-0.15 and RS-0.15 average class accuracy over time on the $20NG_{GBF}$ dataset.

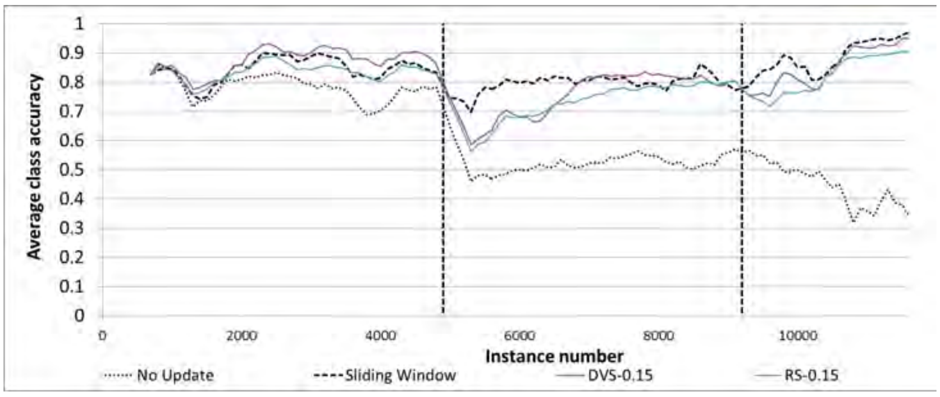


Figure C.3: DVS-0.15 and RS-0.15 average class accuracy over time on the $20NG_{SIF}$ dataset.

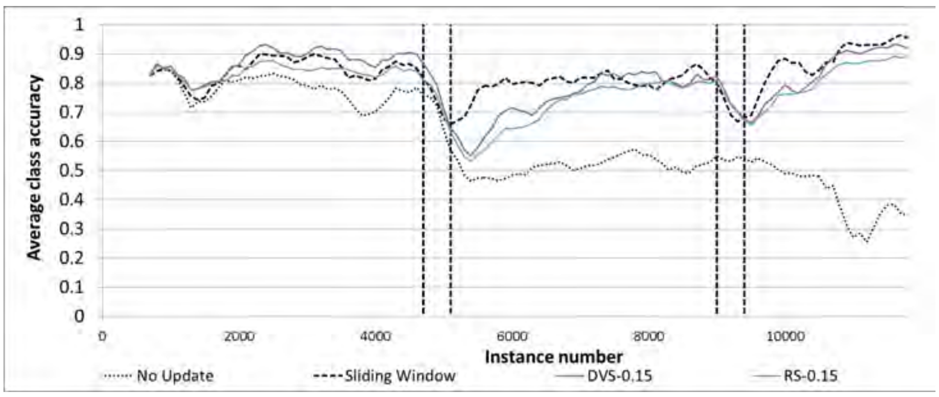


Figure C.4: DVS-0.15 and RS-0.15 average class accuracy over time on the $20NG_{GIF}$ dataset.

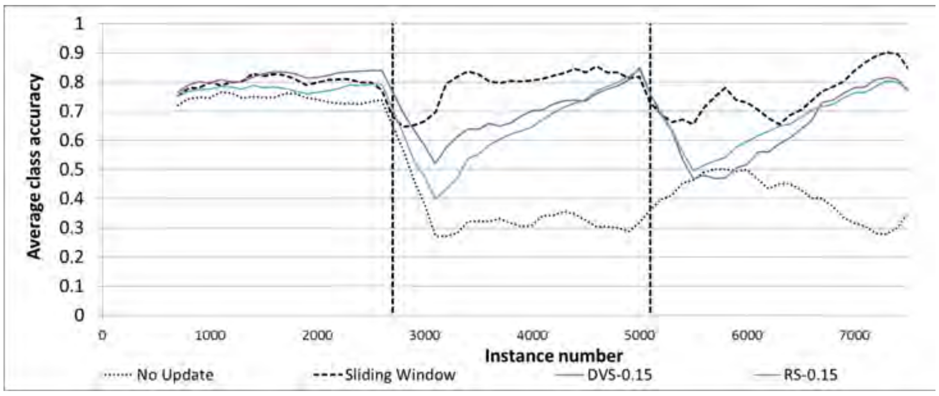


Figure C.5: DVS-0.15 and RS-0.15 average class accuracy over time on the $20NG_{SBC}$ dataset.

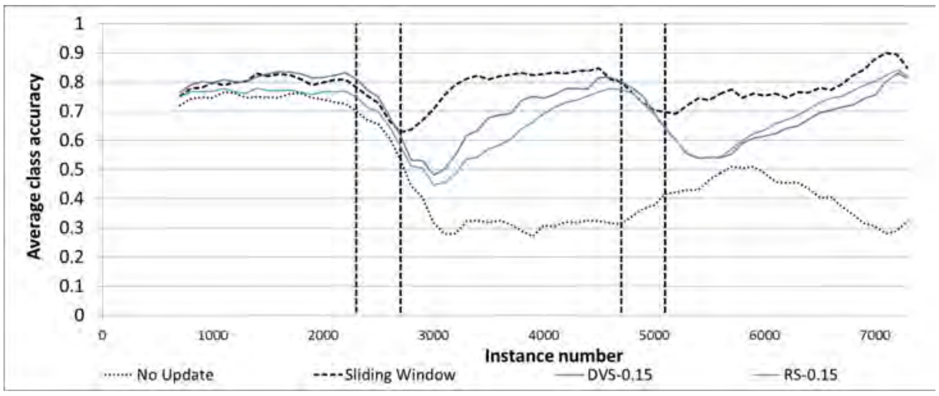


Figure C.6: DVS-0.15 and RS-0.15 average class accuracy over time on the $20NG_{GBC}$ dataset.

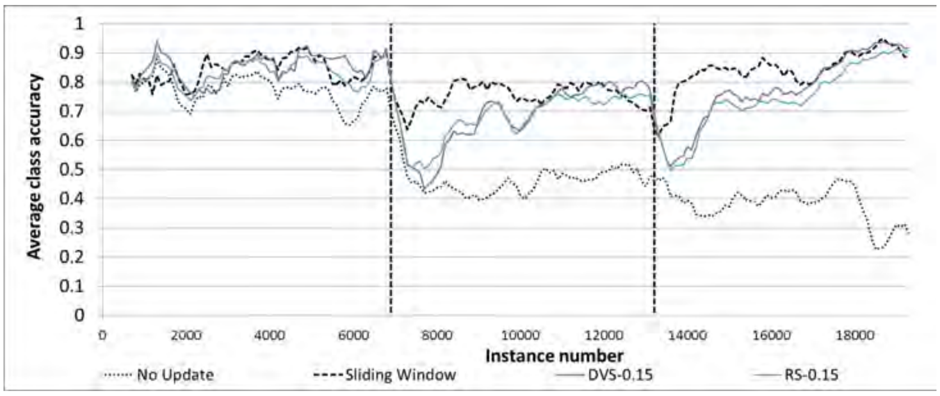


Figure C.7: DVS-0.15 and RS-0.15 average class accuracy over time on the 20NG$_{SIC}$ dataset.

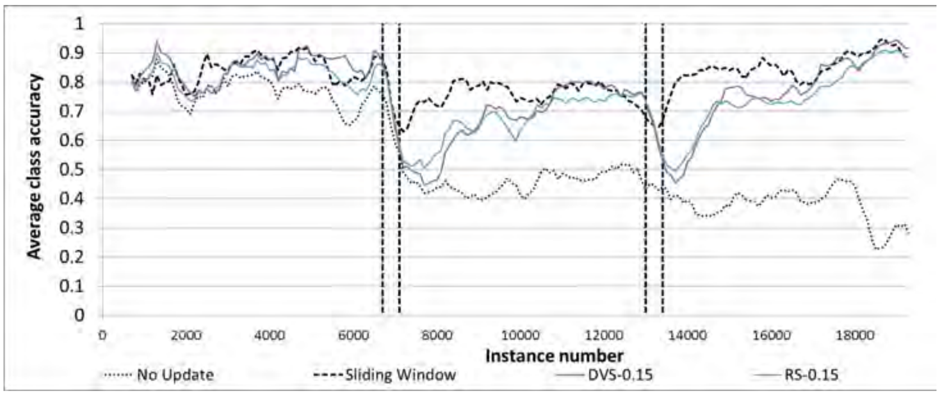


Figure C.8: DVS-0.15 and RS-0.15 average class accuracy over time on the 20NG$_{GIC}$ dataset.

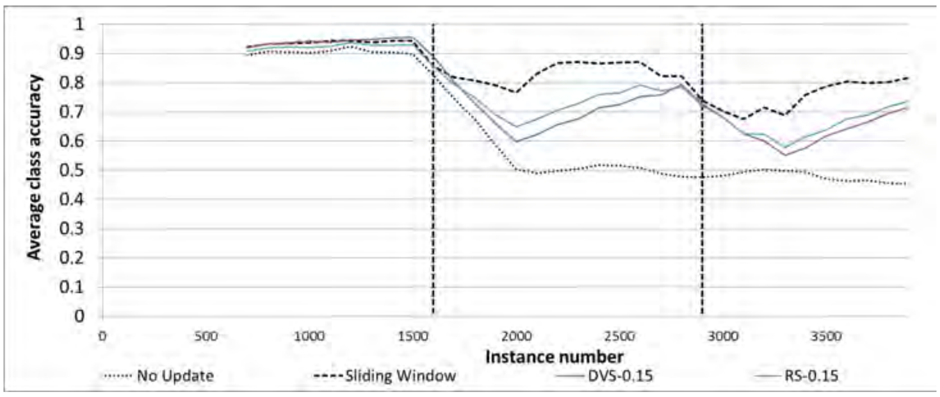


Figure C.9: DVS-0.15 and RS-0.15 average class accuracy over time on the Reuters$_{SBF}$ dataset.

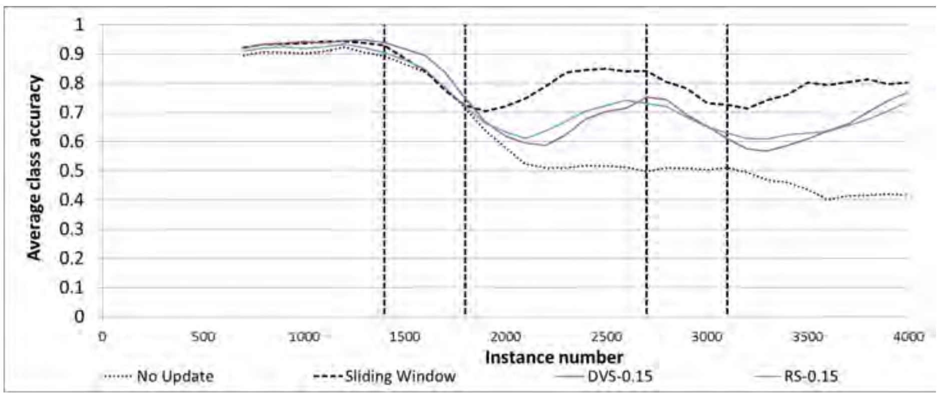


Figure C.10: DVS-0.15 and RS-0.15 average class accuracy over time on the Reuters$_{GBF}$ dataset.

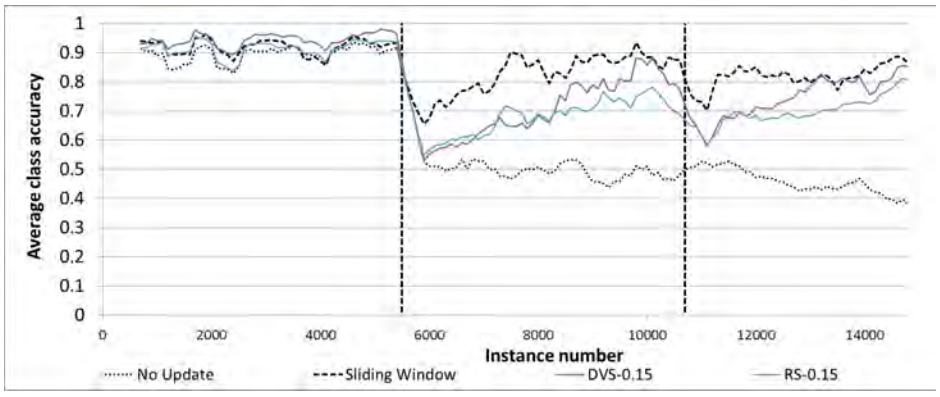


Figure C.11: DVS-0.15 and RS-0.15 average class accuracy over time on the Reuters$_{SIF}$ dataset.

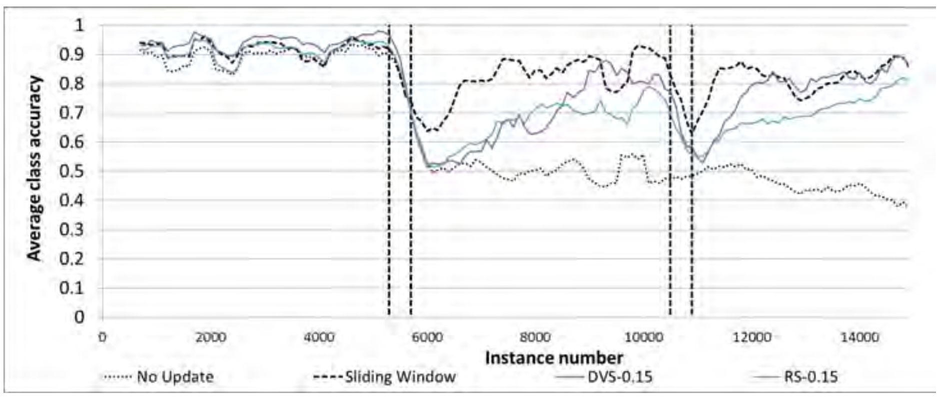


Figure C.12: DVS-0.15 and RS-0.15 average class accuracy over time on the Reuters$_{GIF}$ dataset.

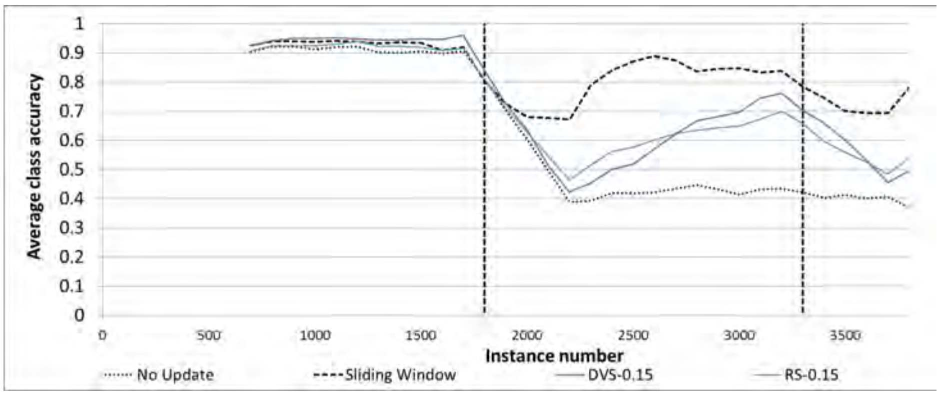


Figure C.13: DVS-0.15 and RS-0.15 average class accuracy over time on the Reuters$_{SBC}$ dataset.

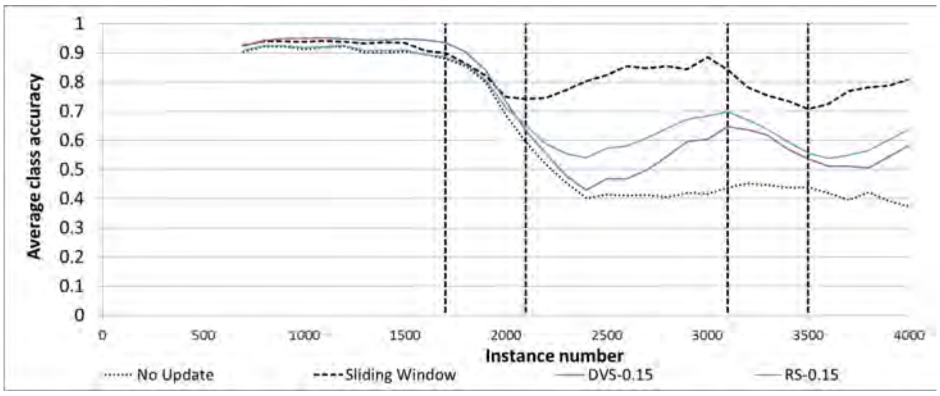


Figure C.14: DVS-0.15 and RS-0.15 average class accuracy over time on the Reuters$_{GBC}$ dataset.

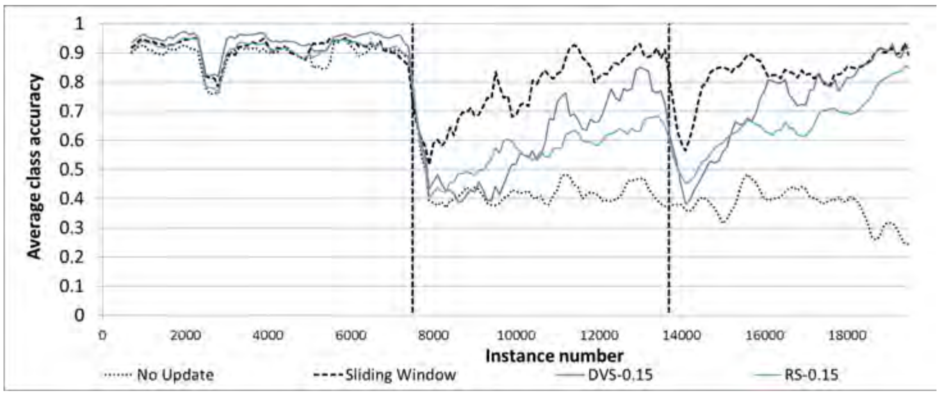


Figure C.15: DVS-0.15 and RS-0.15 average class accuracy over time on the Reuters$_{SIC}$ dataset.

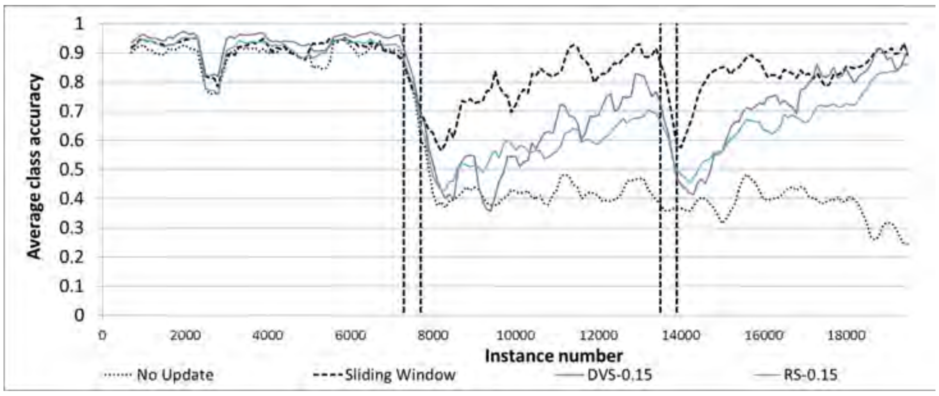


Figure C.16: DVS-0.15 and RS-0.15 average class accuracy over time on the Reuters$_{GIC}$ dataset.

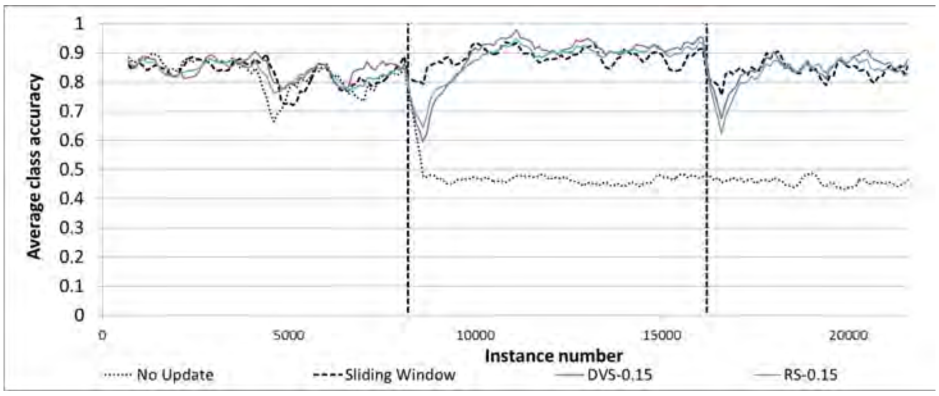


Figure C.17: DVS-0.15 and RS-0.15 average class accuracy over time on the NS$_{SBF}$ dataset.

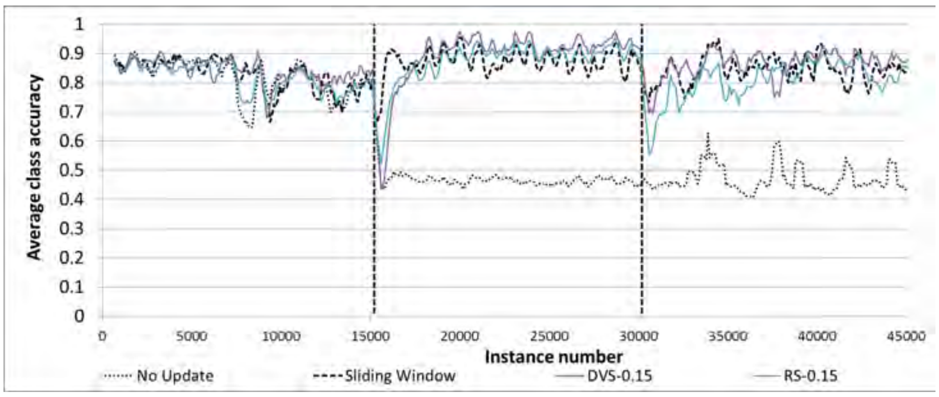


Figure C.18: DVS-0.15 and RS-0.15 average class accuracy over time on the NS$_{SIF}$ dataset.

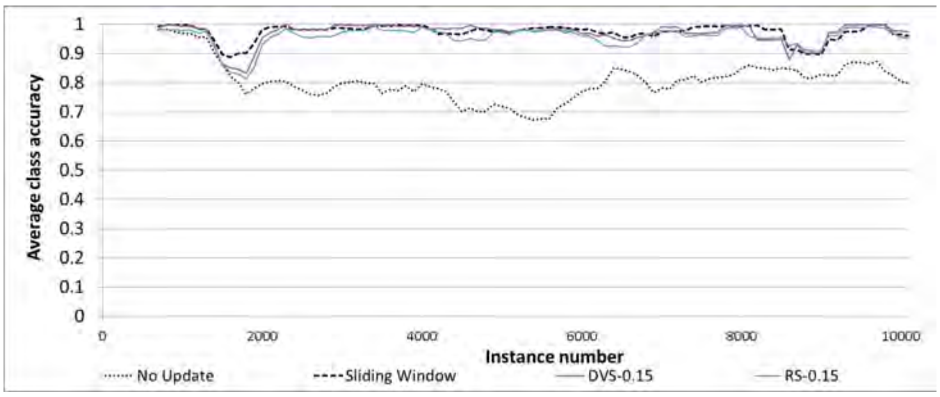


Figure C.19: DVS-0.15 and RS-0.15 average class accuracy over time on the Spam_1 dataset.

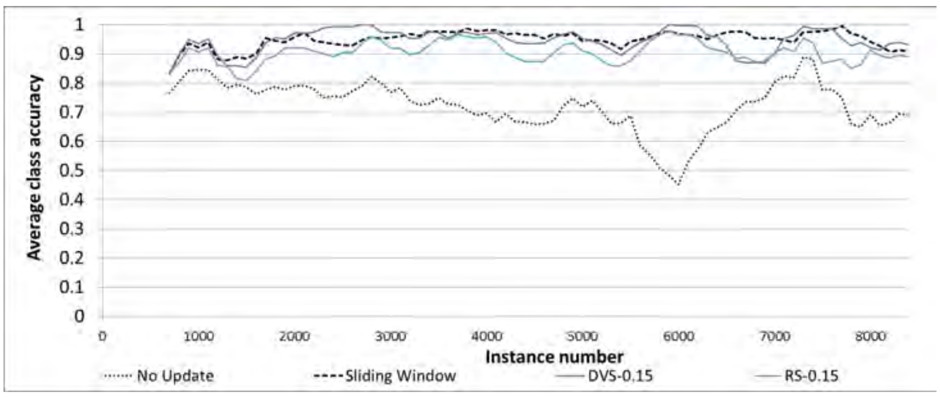


Figure C.20: DVS-0.15 and RS-0.15 average class accuracy over time on the Spam_2 dataset.

Appendix D

Additional Material for Chapter 6

D.1 CDBD Accuracy Over Time Graphs

This section contains the omitted average accuracy over time graphs from Chapter 6. For completeness it also include the CDBD-2-3 graphs which were presented previously in Section 6.3.3.

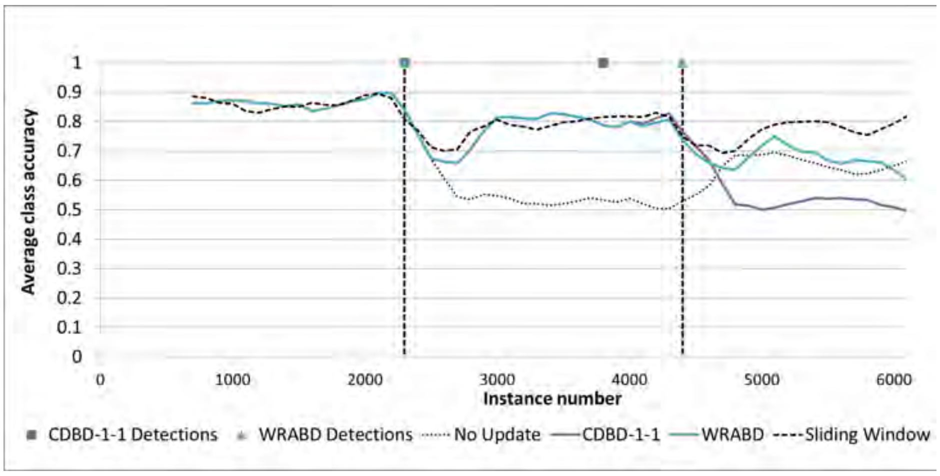


Figure D.1: CDBD-1-1 average class accuracy over time on the 20NG$_{SBF}$ dataset

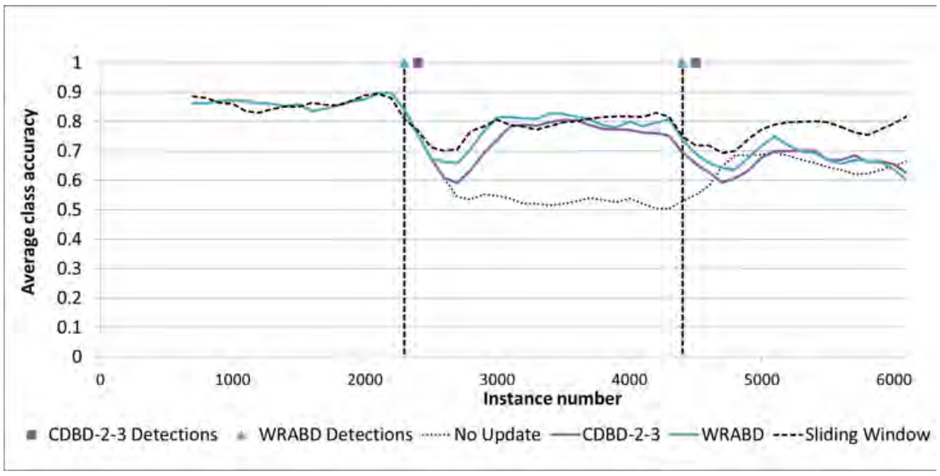


Figure D.2: CDBD-2-3 average class accuracy over time on the 20NG$_{SBF}$ dataset

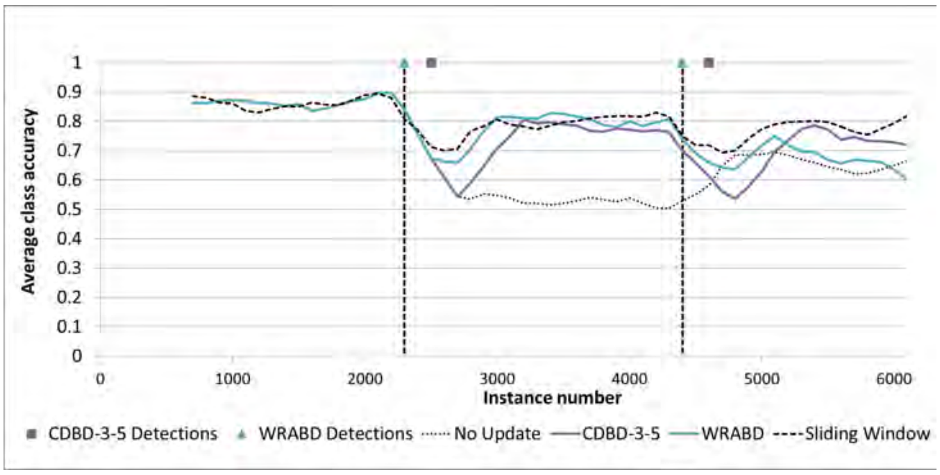


Figure D.3: CDBD-3-5 average class accuracy over time on the 20NG$_{SBF}$ dataset

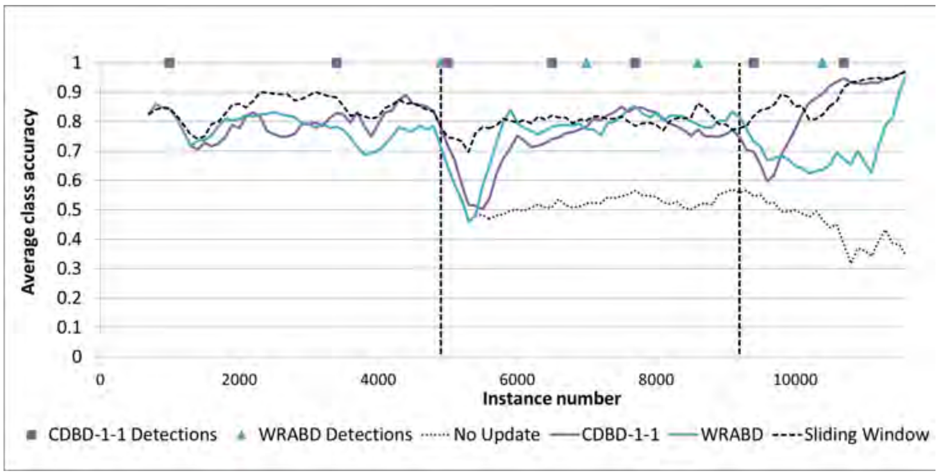


Figure D.4: CDBD-1-1 average class accuracy over time on the $20NG_{SIF}$ dataset

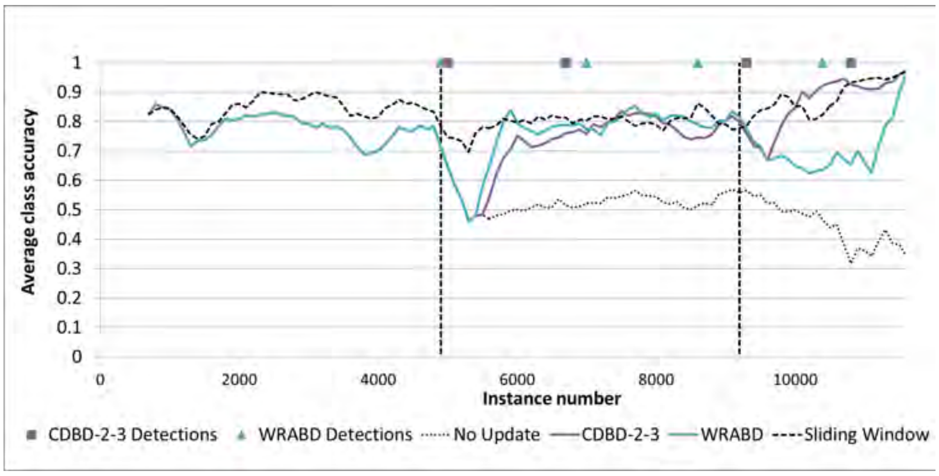


Figure D.5: CDBD-2-3 average class accuracy over time on the $20NG_{SIF}$ dataset

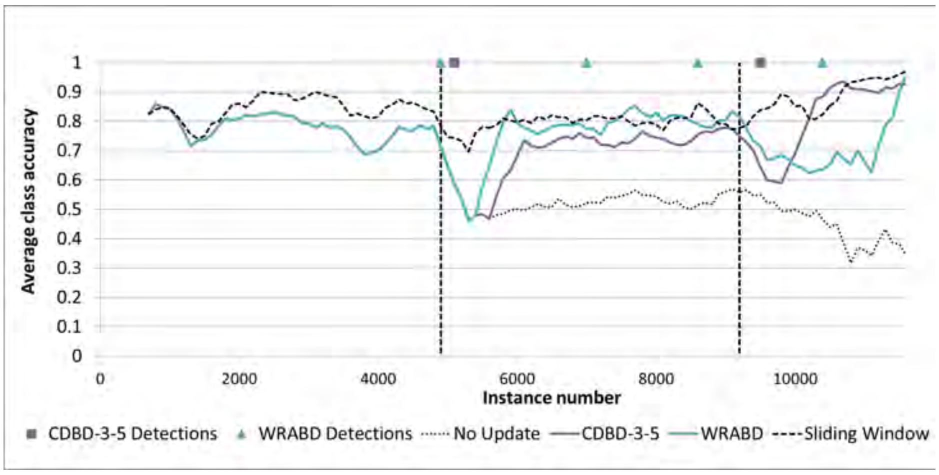


Figure D.6: CDBD-3-5 average class accuracy over time on the $20NG_{SIF}$ dataset

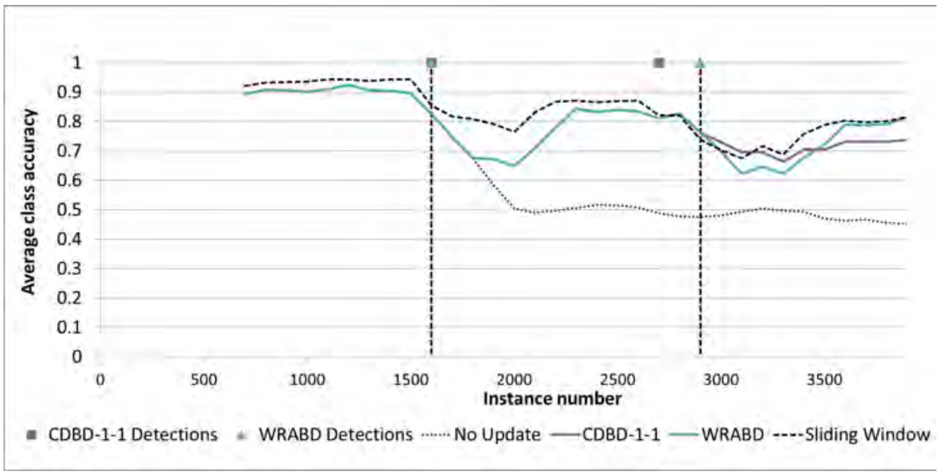


Figure D.7: CDBD-1-1 average class accuracy over time on the Reuters$_{SBF}$ dataset

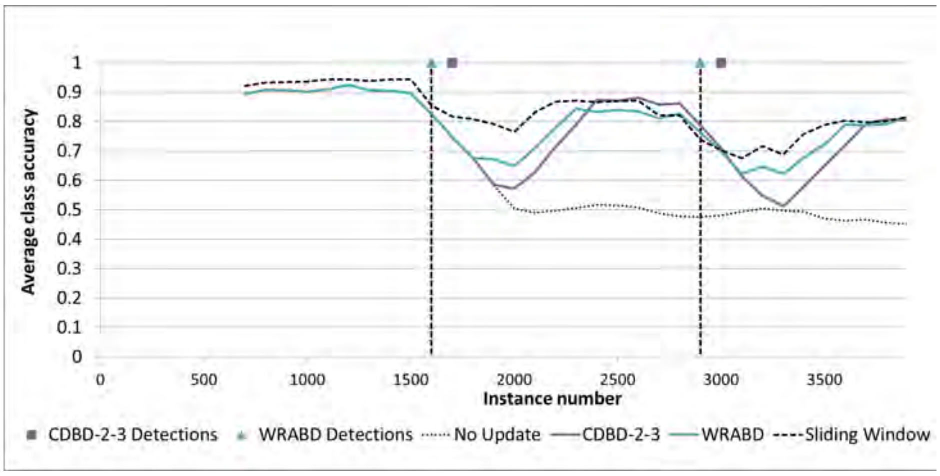


Figure D.8: CDBD-2-3 average class accuracy over time on the Reuters$_{SBF}$ dataset

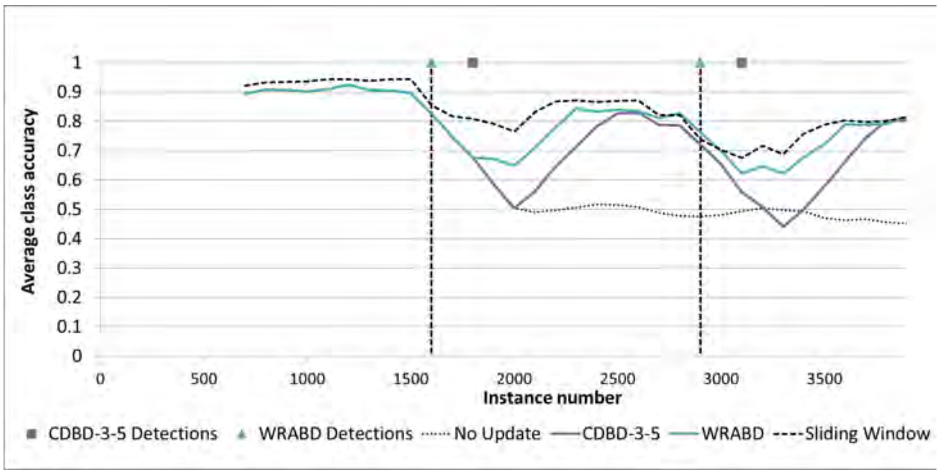


Figure D.9: CDBD-3-5 average class accuracy over time on the Reuters$_{SBF}$ dataset

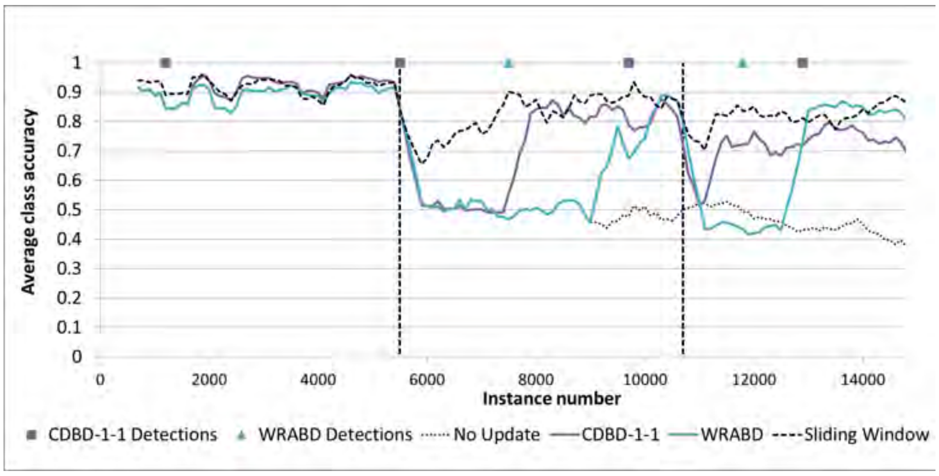


Figure D.10: CDBD-1-1 average class accuracy over time on the Reuters$_{SIF}$ dataset

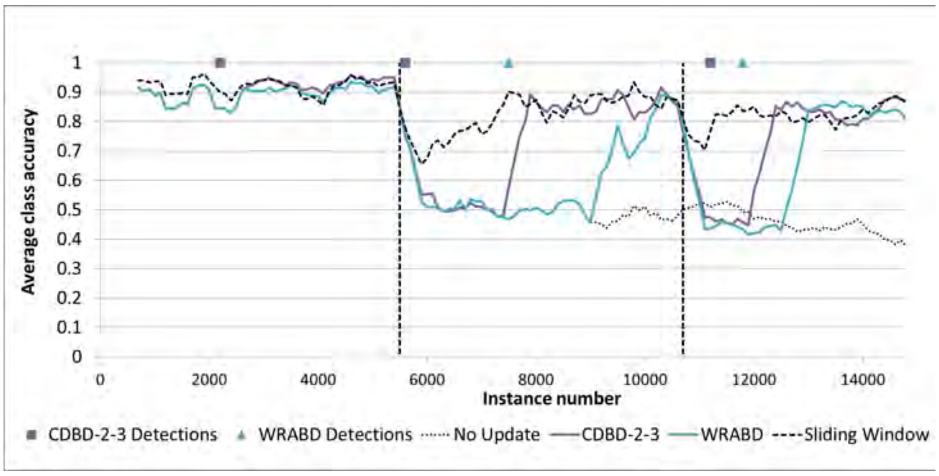


Figure D.11: CDBD-2-3 average class accuracy over time on the Reuters$_{SIF}$ dataset

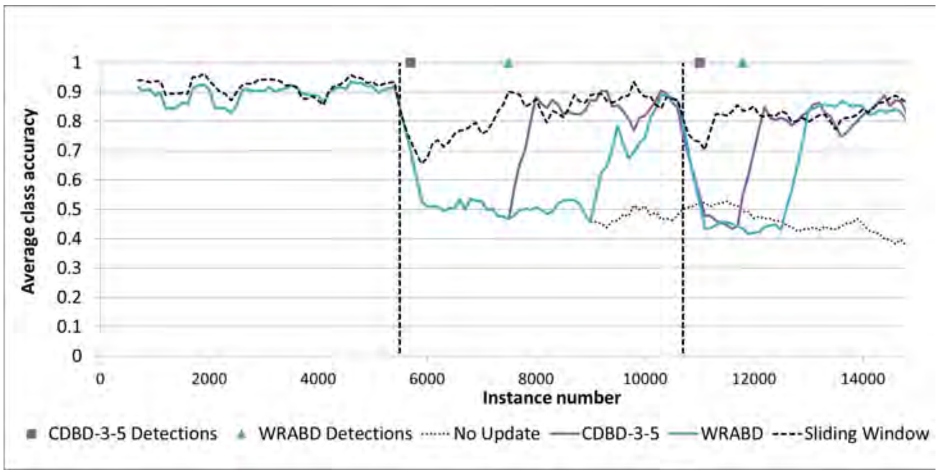


Figure D.12: CDBD-3-5 average class accuracy over time on the Reuters$_{SIF}$ dataset

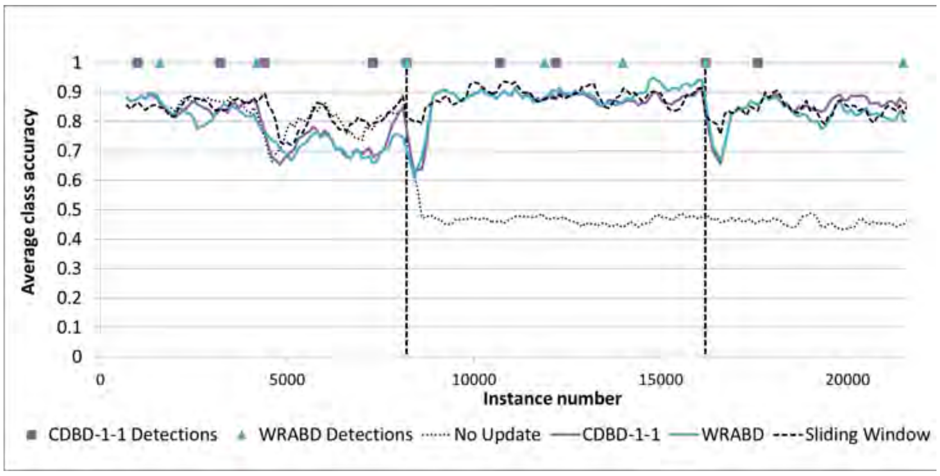


Figure D.13: CDBD-1-1 average class accuracy over time on the NS_{SBF} dataset

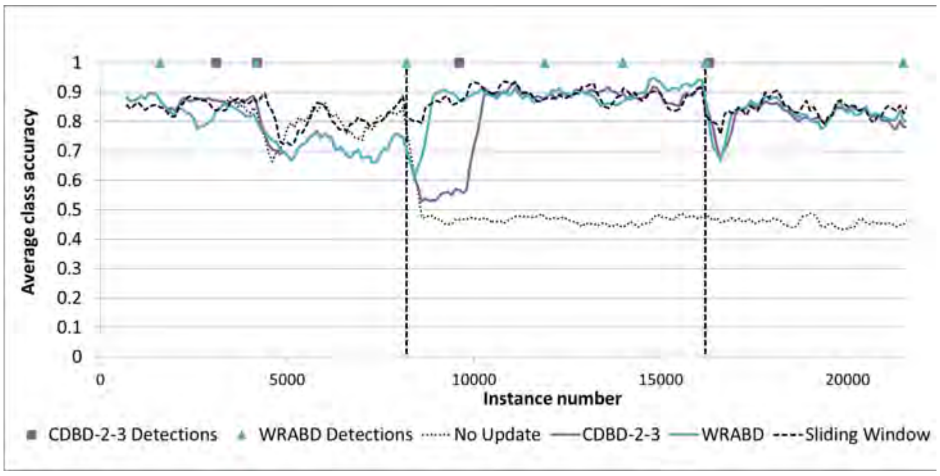


Figure D.14: CDBD-2-3 average class accuracy over time on the NS_{SBF} dataset

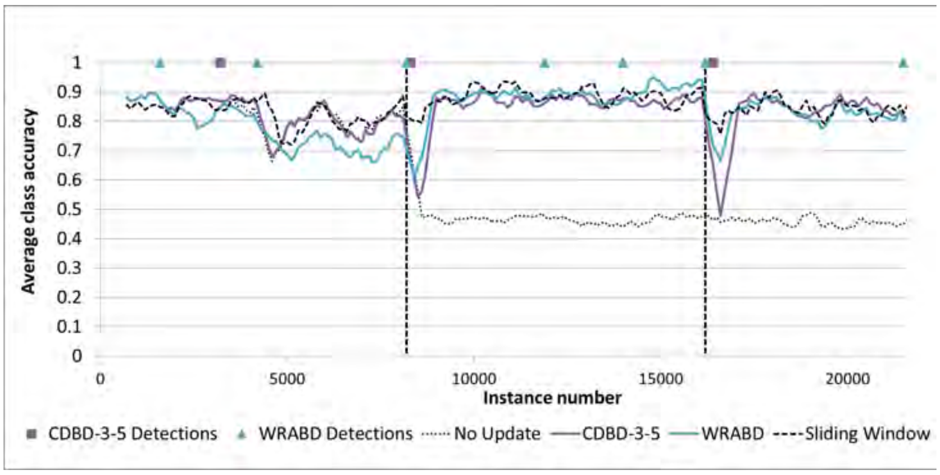


Figure D.15: CDBD-3-5 average class accuracy over time on the NS_{SBF} dataset

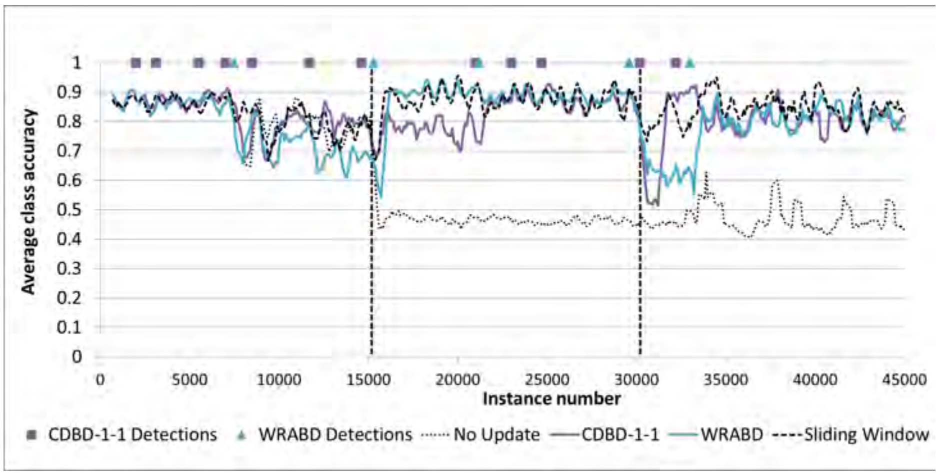


Figure D.16: CDBD-1-1 average class accuracy over time on the NS_{SIF} dataset

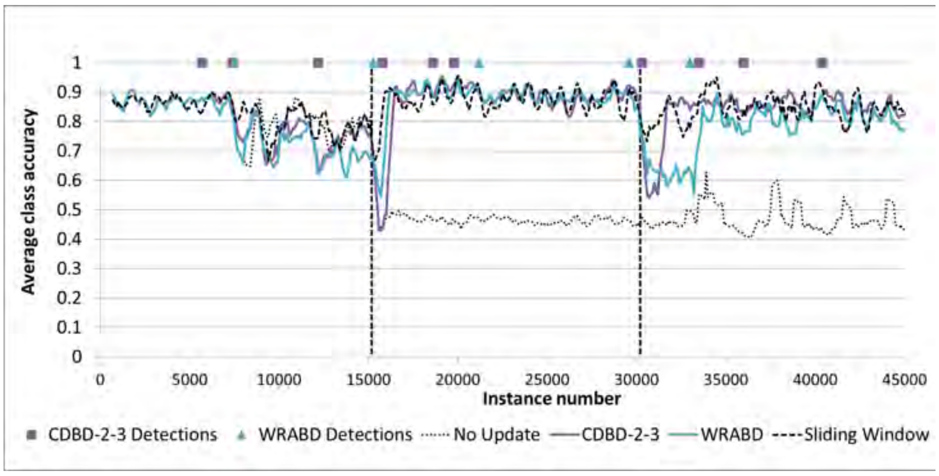


Figure D.17: CDBD-2-3 average class accuracy over time on the NS_{SIF} dataset

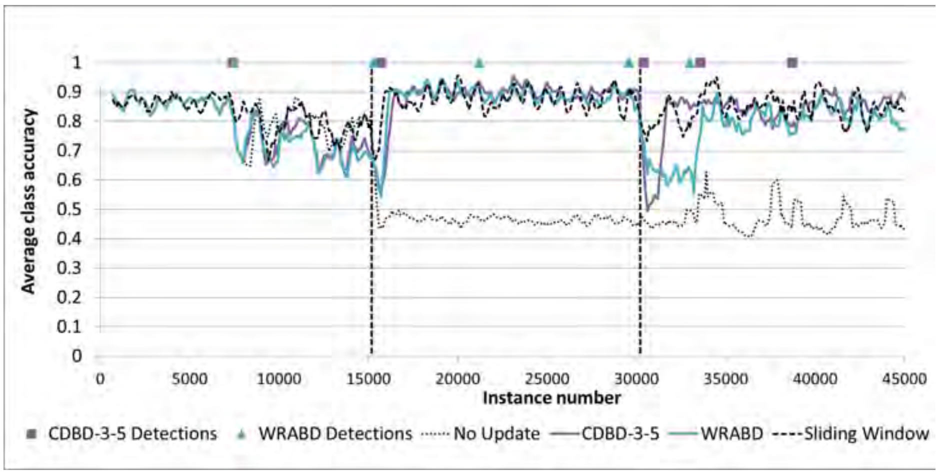


Figure D.18: CDBD-3-5 average class accuracy over time on the NS_{SIF} dataset

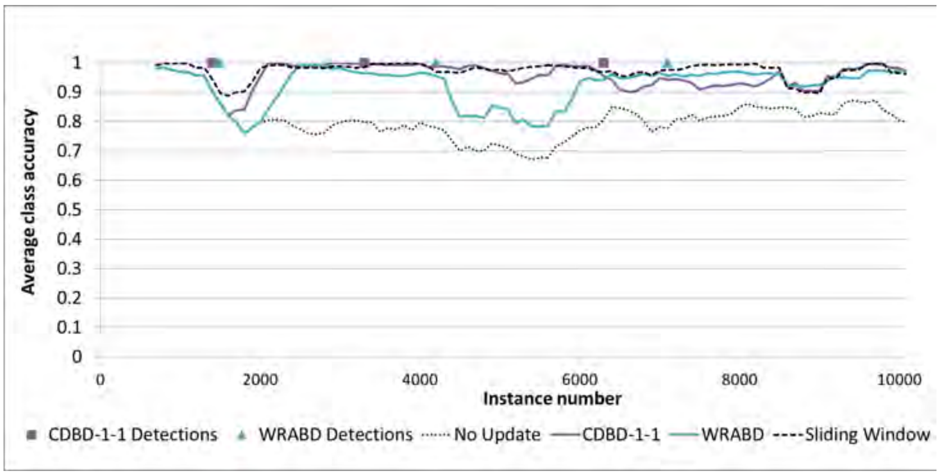


Figure D.19: CDBD-1-1 average class accuracy over time on the Spam$_1$ dataset

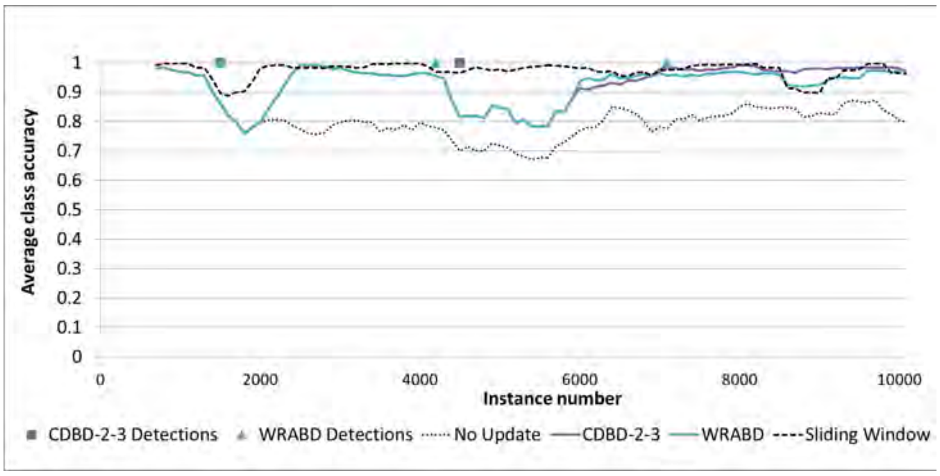


Figure D.20: CDBD-2-3 average class accuracy over time on the Spam$_1$ dataset

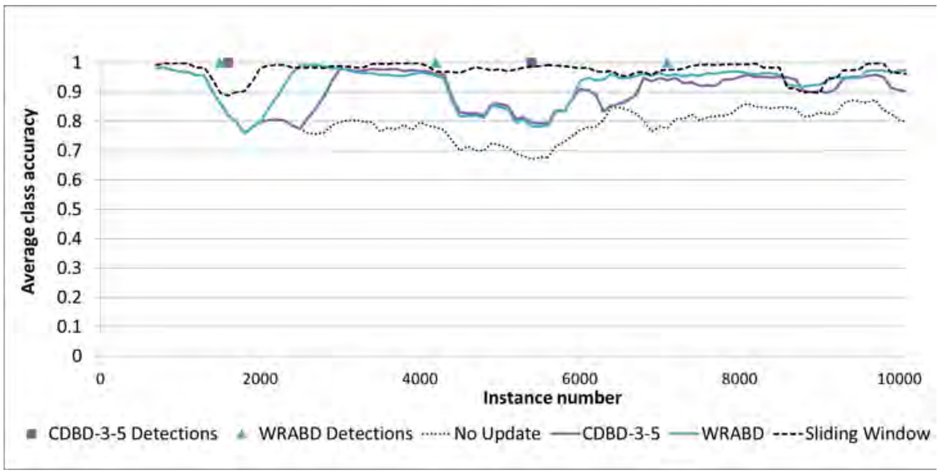


Figure D.21: CDBD-3-5 average class accuracy over time on the Spam$_1$ dataset

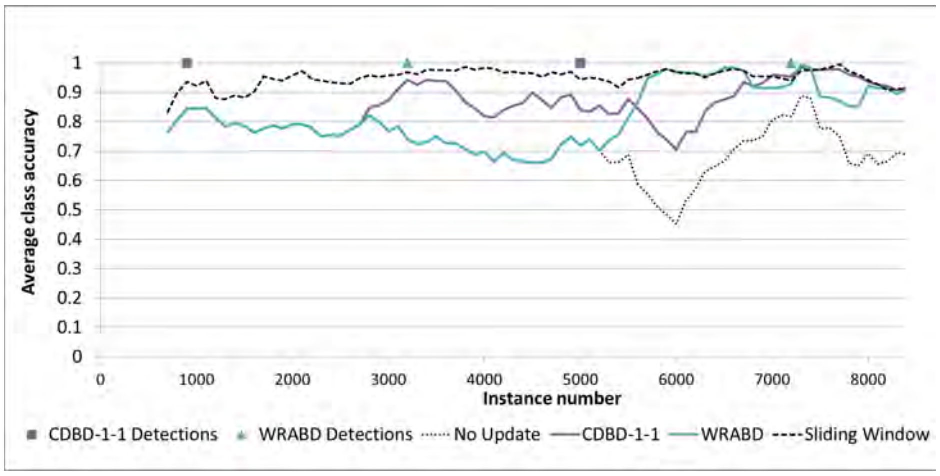


Figure D.22: CDBD-1-1 average class accuracy over time on the Spam_2 dataset

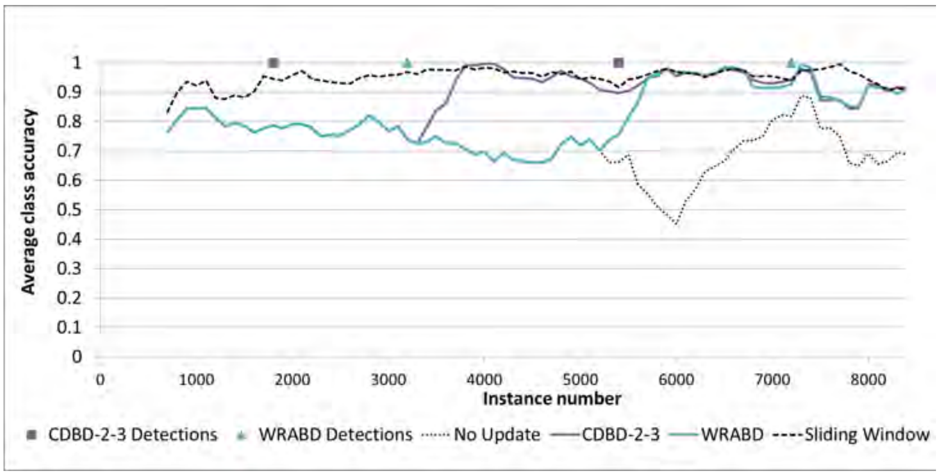


Figure D.23: CDBD-2-3 average class accuracy over time on the Spam_2 dataset

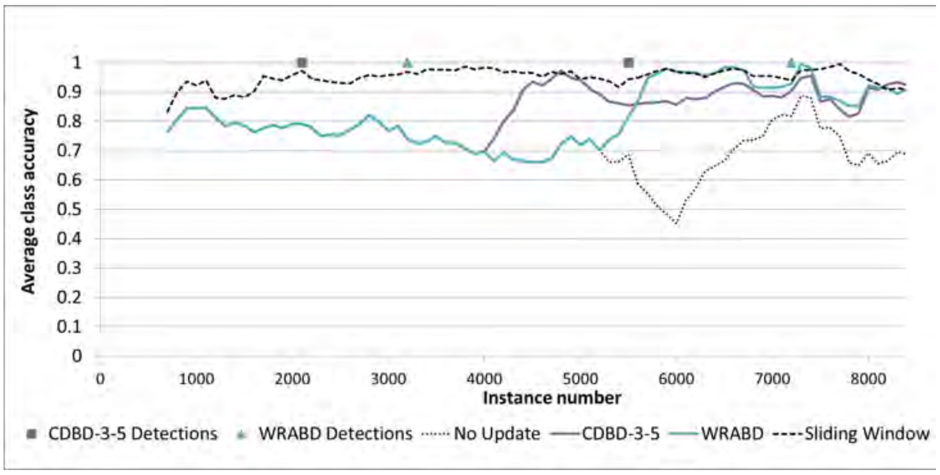


Figure D.24: CDBD-3-5 average class accuracy over time on the Spam_2 dataset

Appendix E

Additional Material for Chapter 7

E.1 Finding the Best Budget for DVS

This section contains supplemental material describing how the best budget for DVS was determined. The best labelling budget is wholly dependent on the relative importance placed on the two evaluation metrics average class accuracy and fraction of labels used. TOPSIS allows the two metrics to be weighted and combined into one measure, upon which the differently parameterised versions of DVS can be ranked.

The methodology used will follow the methodology outlined in Section 7.3.1, which can be summarised as: (1) calculate the TOPSIS score using equal weights, (2) rank the approaches based on TOPSIS score, (3) adjust the labelling cost weight from zero to one, while setting the accuracy weight to $1 - \text{labellingCostWeight}$. For each value of the two weights recalculate the TOPSIS score for each approach and rank the approaches based on their score.

Table E.1 shows step one, the TOPSIS score for each version of DVS on each dataset when equal weights are used. The average class accuracy and fraction of labels used for each approach are the values obtained in Chapter 5 (for more information about the experiment datasets and methodology see Section 5.2).

Table E.1: The TOPSIS score for differently parameterised versions of DVS.

	DVS-0.05	**DVS-0.10**	**DVS-0.15**	**DVS-0.25**	**DVS-0.50**	**DVS-0.75**
20NG$_{SBF}$	0.94	0.92	0.85	0.71	0.36	0.06
20NG$_{GBF}$	0.93	0.91	0.85	0.71	0.36	0.07
20NG$_{SIF}$	0.95	0.92	0.86	0.71	0.36	0.05
20NG$_{GIF}$	0.96	0.92	0.86	0.71	0.36	0.04
20NG$_{SBC}$	0.86	0.89	0.85	0.72	0.38	0.14
20NG$_{GBC}$	0.85	0.88	0.85	0.72	0.38	0.15
20NG$_{SIC}$	0.90	0.91	0.85	0.72	0.37	0.10
20NG$_{GIC}$	0.90	0.91	0.85	0.72	0.37	0.10
Reuters$_{SBF}$	0.92	0.90	0.85	0.71	0.36	0.08
Reuters$_{GBF}$	0.90	0.89	0.85	0.71	0.37	0.10
Reuters$_{SIF}$	0.91	0.91	0.85	0.72	0.37	0.09
Reuters$_{GIF}$	0.91	0.91	0.85	0.72	0.37	0.09
Reuters$_{SBC}$	0.89	0.87	0.84	0.71	0.37	0.11
Reuters$_{GBC}$	0.89	0.87	0.83	0.71	0.37	0.11
Reuters$_{SIC}$	0.89	0.90	0.85	0.71	0.37	0.11
Reuters$_{GIC}$	0.89	0.90	0.85	0.71	0.37	0.11
NS$_{SBF}$	0.97	0.93	0.86	0.71	0.36	0.03
NS$_{SIF}$	0.98	0.93	0.86	0.71	0.36	0.01
Spam$_1$	1.00	0.93	0.86	0.71	0.36	0.00
Spam$_2$	0.99	0.93	0.86	0.71	0.36	0.00

Table E.2 further clarifies the results by ranking all of the approaches from best to worst on each dataset, based on their TOPSIS score. The best labelling budget seems to be 0.05, based on the average rank. However, this conclusion is predicated on equal TOPSIS weights being used.

Table E.2: The rank for differently parameterised versions of DVS based on TOPSIS score.

	DVS-0.05	**DVS-0.10**	**DVS-0.15**	**DVS-0.25**	**DVS-0.50**	**DVS-0.75**
20NG$_{SBF}$	1	2	3	4	5	6
20NG$_{GBF}$	1	2	3	4	5	6
20NG$_{SIF}$	1	2	3	4	5	6
20NG$_{GIF}$	1	2	3	4	5	6
20NG$_{SBC}$	2	1	3	4	5	6
20NG$_{GBC}$	3	1	2	4	5	6
20NG$_{SIC}$	2	1	3	4	5	6
20NG$_{GIC}$	2	1	3	4	5	6
Reuters$_{SBF}$	1	2	3	4	5	6
Reuters$_{GBF}$	1	2	3	4	5	6
Reuters$_{SIF}$	2	1	3	4	5	6
Reuters$_{GIF}$	2	1	3	4	5	6
Reuters$_{SBC}$	1	2	3	4	5	6
Reuters$_{GBC}$	1	2	3	4	5	6
Reuters$_{SIC}$	2	1	3	4	5	6
Reuters$_{GIC}$	2	1	3	4	5	6
NS$_{SBF}$	1	2	3	4	5	6
NS$_{SIF}$	1	2	3	4	5	6
Spam$_1$	1	2	3	4	5	6
Spam$_2$	1	2	3	4	5	6
AVG	1.45	1.6	2.95	4	5	6

Table E.3 shows how the average ranks change as the TOPSIS weights are adjusted, giving an insight into what algorithm parameter values are the most appropriate for any given value of the weights. Each row shows the average rank (on all datasets) obtained by an approach for a given labelling cost weight. The first row shows the ranks when the amount of labelled data used is not considered (the assumption made by most concept handling techniques). The opposite situation is when the labelling cost weight is set to one, i.e. a scenario where the accuracy is unimportant and the only consideration is the amount of labelled data used.

The last row shows the average of the average ranks, which gives an indication of how well an approach does over all the different weight values. Figure E.1 shows a visual representation of Table E.3.

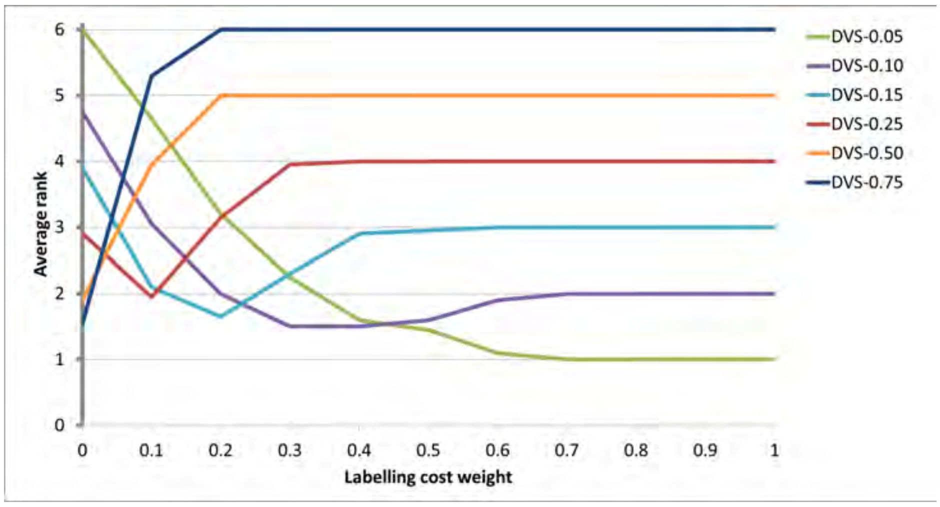


Figure E.1: The average rank for differently parameterised versions of DVS while varying the labelling cost weight.

The graph shows that when the TOPSIS scores are calculated using weights heavily skewed towards accuracy then approaches with a large labelling budget have the lowest average rank. Approaches with a small labelling budget receive a more competitive average rank as the weights are moved to place more importance on the labelling cost. The overall narrative from the ranks in Figure E.1 is that the appropriateness of a given budget is wholly dependant on the accuracy and labelling cost weights. However, Figure E.1 suggests that a labelling budget in the 0.05 to 0.15 range seems to be appropriate in most labelling weight scenarios. The importance of a low labelling budget can be explained by the average ranks, which show that

Table E.3: The average rank of each approach while varying the labelling cost weight.

Labelling cost weight:	**DVS-0.05**	**DVS-0.10**	**DVS-0.15**	**DVS-0.25**	**DVS-0.50**	**DVS-0.75**
0	6	4.75	3.9	2.9	1.9	1.55
0.1	4.65	3.05	2.10	1.95	3.95	5.30
0.2	3.20	2	1.65	3.15	5	6
0.3	2.25	1.50	2.30	3.95	5	6
0.4	1.60	1.50	2.90	4	5	6
0.5	1.45	1.60	2.95	4	5	6
0.6	1.10	1.90	3	4	5	6
0.7	1.0	2	3	4	5	6
0.8	1.0	2	3	4	5	6
0.9	1	2	3	4	5	6
1	1	2	3	4	5	6
AVG	2.2	2.21	2.8	3.63	4.62	5.53

the increase in classification accuracy obtained by increasing the labelling budget is offset by the increase in labels used (except in the scenario where the labelling cost weight is very low, in this case less than 0.1).

E.2 Finding the Best Trigger for CDBD

This section contains supplemental material describing how the best trigger for CDBD was determined. The best trigger is dependent on the relative importance placed on the two evaluation metrics average class accuracy and labels used. The number of labels used cannot be set using a parameter as it can in DVS, but the trigger sensitivity is one of the factors which affects how much labelled data CDBD will use. TOPSIS will be used to weight and combine the two metrics into one measure, upon which CDBD using 1/1, 2/3 and 3/5 triggers can be ranked.

The methodology used will follow the methodology outlined in Section 7.3.1, which can be summarised as: (1) calculate the TOPSIS score using equal weights, (2) rank the approaches based on TOPSIS score, (3) adjust the labelling cost weight from zero to one, while setting the accuracy weight to $1 - \text{labellingCostWeight}$. For each value of the two weights recalculate the TOPSIS score for each approach and rank the approaches based on their score.

Table E.4 shows step one, the TOPSIS score for each version of CDBD on each dataset when equal weights are used. The average class accuracy and fraction of labels used for each approach are the values obtained in Chapter 6 (for more information about the experiment datasets and methodology see Section 6.3).

Table E.5 shows CDBD with the three different triggers ranked by TOPSIS score. CDBD-3-5 is the best approach according to the average rank. However, this conclusion is predicated on equal TOPSIS weights being used.

Varying the weights between zero and one gives the average ranks listed in Table E.6. Figure E.2 shows a visual representation of Table E.6.

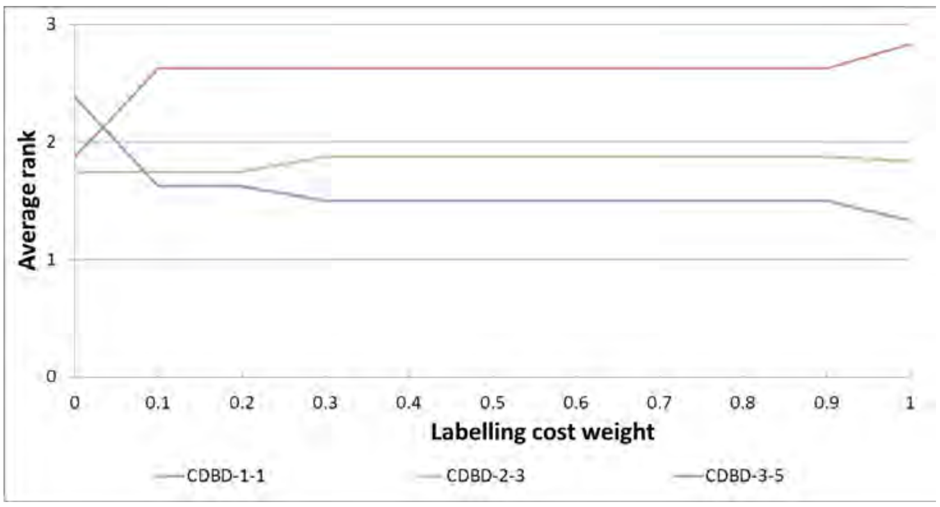


Figure E.2: The average rank of CDBD with different triggers while varying the labelling cost weight.

The average ranks remain static for most labelling cost weight values. The 2/3 trigger is the best trigger when the labelling cost is not considered (the labelling weight is zero). However, once the labelling cost weight goes above 0.10 the 3/5 trigger is the best trigger, closely followed by the 2/3 trigger. Overall the 3/5 trigger seems to be the best trigger, yet CDBD-1-1 and CDBD-2-3 might also be appropriate if the 3/5 trigger is too slow at detecting change.

Table E.4: TOPSIS score for CDBD using 1/1, 2/3 and 3/5 triggers.

	CDBD-1-1	**CDBD-2-3**	**CDBD-3-5**
20NG$_{SBF}$	0.00	0.73	1.00
20NG$_{SIF}$	0.04	0.64	0.96
Reuters$_{SBF}$	1.00	0.76	0.00
Reuters$_{SIF}$	0.00	0.70	0.98
NS$_{SBF}$	0.03	0.83	1.00
NS$_{SIF}$	0.00	0.14	1.00
Spam$_1$	0.07	0.86	0.93
Spam$_2$	0.37	1.00	0.00

Table E.5: CDBD using different triggers ranked by TOPSIS score.

	CDBD-1-1	**CDBD-2-3**	**CDBD-3-5**
20NG$_{SBF}$	3	2	1
20NG$_{SIF}$	3	2	1
Reuters$_{SBF}$	1	2	3
Reuters$_{SIF}$	3	2	1
NS$_{SBF}$	3	2	1
NS$_{SIF}$	3	2	1
Spam$_1$	3	2	1
Spam$_2$	2	1	3
AVG	2.63	1.88	1.50

Table E.6: The average rank of CDBD with different triggers while varying the labelling cost weight.

Labelling cost weight:	**CDBD-1-1**	**CDBD-2-3**	**CDBD-3-5**
0	1.88	1.75	2.38
0.1	2.63	1.75	1.63
0.2	2.63	1.75	1.63
0.3	2.63	1.88	1.50
0.4	2.63	1.88	1.50
0.5	2.63	1.88	1.50
0.6	2.63	1.88	1.50
0.7	2.63	1.88	1.50
0.8	2.63	1.88	1.50
0.9	2.63	1.88	1.50
1	2.83	1.83	1.33
AVG	**2.58**	**1.84**	**1.59**

References

Abdullah, M. & Ganapathy, V. (2000). Neural network ensemble for financial trend prediction. *TENCON 2000. Proceedings*, **3**, 157–161. 60

Ada, I. & Berthold, M. (2011). Unifying change –towards a framework for detecting the unexpected. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, 555 –559. 54, 63

Baena-García, M., Campo-Ávila, J.d., Fidalgo, R., Bifet, A., Gavaldà, R. & Morales-Bueno, R. (2006). Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*, 77–86. 3, 43, 48

Bifet, A. & Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *SIAM International Conference on Data Mining*, 443–448. 48

Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T. & Seidl, T. (2010). Moa: Massive online analysis, a framework for stream classification and clustering. *Journal of Machine Learning Research - Proceedings Track*, **11**, 44–50. 151

Black, M. & Hickey, R. (1999). Maintaining the performance of a learned classifier under concept drift. *Intelligent Data Analysis*, **3**, 453–474. 46

Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**, 123–140. 23

Bruckhaus, T. (2007). The business impact of predictive analytics. *Knowledge Discovery and Data Mining: Challenges and Realities*, 114. 25

Catley, C., Smith, K., McGregor, C. & Tracy, M. (2009). Extending crisp-dm to incorporate temporal data mining of multidimensional medical data streams: A neonatal intensive care unit case study. In *Computer-Based Medical Systems, 2009. CBMS 2009. 22nd IEEE International Symposium on*, 1–5. 12

Chang, C.C. & Lin, C.J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, **2**, 27:1–27:27, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 78

Chawla, N.V., Bowyer, K.W., Hall, L.O. & Kegelmeyer, W.P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, **16**, 321–357. 179

Church, K. & Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, **16**, 22–29. 29

Cohn, D., Atlas, L. & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, **15**, 201–221. 52

Cooley, R. (1999). Classification of news stories using support vector machines. In *Proc. 16th International Joint Conference on Artificial Intelligence Text Mining Workshop*. 26, 28, 30

Cover, T. & Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, **13**, 21 –27. 19

Cunningham, P. (2009). A taxonomy of similarity mechanisms for case-based reasoning. *Knowledge and Data Engineering, IEEE Transactions on*, **21**, 1532 –1543. 19

Delany, S., Cunningham, P., Tsymbal, A. & Coyle, L. (2005). A case-based technique for tracking concept drift in spam filtering. *Knowledge-Based Systems*, **18**, 187–195. 2, 67

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, **7**, 1–30. 85, 86

Dietterich, T. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems*, vol. 1857 of *Lecture Notes in Computer Science*, 1–15, Springer Berlin / Heidelberg. 22, 23

Dumais, S.T., Platt, J.C., Hecherman, D. & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management*, 148–155, Bethesda, Maryland, USA. 17, 19, 21, 26, 151

EL-Manzalawy, Y. & Honavar, V. (2005). *WLSVM: Integrating LibSVM into Weka Environment*. Software available at `http://www.cs.iastate.edu/~yasser/wlsvm`. 78

Fan, W., Huang, Y., Wang, H. & Yu, P.S. (2004a). Active mining of data streams. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, vol. 117, 457. 3, 55, 56, 115

Fan, W., Huang, Y.a. & Yu, P. (2004b). Decision tree evolution using limited number of labeled data items from drifting data streams. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, 379 – 382. 56

Fawcett, T. & Provost, F.J. (1996). Combining data mining and machine learning for effective user profiling. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 8–13. 2

Fayyad, U.M., Piatetsky-Shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, **17**, 37–54. 12

Fischer, B. & Zigmond, M. (2010). The essential nature of sharing in science. *Science and engineering ethics*, **16**, 783–799. 60

Frank, A. & Asuncion, A. (2010). UCI machine learning repository. 60

Freund, Y. & Schapire, R.E. (1996). Experiments with a new boosting algorithm. In L. Saitta, ed., *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, 148–156, Morgan Kaufmann. 23

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, **11**, 86–92. 85

Gama, J., Medas, P., Castillo, G. & Rodrigues, P. (2004). Learning with drift detection. In A. Bazzan & S. Labidi, eds., *Advances in Artificial Intelligence –SBIA 2004*, vol. 3171 of *Lecture Notes in Computer Science*, 66–112, Springer Berlin / Heidelberg. 3, 33, 42, 53

Gao, J., Fan, W., Han, J. & Yu, P.S. (2007). A general framework for mining concept-drifting data streams with skewed distributions. In *Proc. SDM'07*. 34, 49

Guo, G., Li, S. & Chan, K. (2000). Face recognition by support vector machines. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, 196 –201. 2

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. & Witten, I.H. (2009). The weka data mining software: an update. *SIGKDD Explor. Newsl.*, **11**, 10–18. 78

Helmbold, D.P. & Long, P.M. (1994). Tracking drifting concepts by minimizing disagreements. *Machine Learning*, **14**, 27–45. 36

Ho, T.K., Hull, J. & Srihari, S. (1994). Decision combination in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **16**, 66 –75. 24

Hsu, C.W., Chang, C.C. & Lin, C.J. (2003). A practical guide to support vector classification. Tech. rep., Department of Computer Science, National Taiwan University. 21

Hu, R., Lindstrom, P., Delany, S.J. & Mac Namee, B. (2010). Exploring the frontier of uncertainty space. In *AISTATS 2010 Workshop on Active Learning and Experimental Design*. 181

Huang, S. & Dong, Y. (2007). An active learning system for mining time-changing data streams. *Intelligent Data Analysis*, **11**, 401 – 419. 56

Hwang, C.L. & Yoon, K. (1981). *Multiple attribute decision making: methods and applications : a state-of-the-art survey*. Springer-Verlag. 83

Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. In *Machine Learning: ECML-98, 10th European Conference on Machine Learning, Proceedings*, 137–142, Chemnitz, Germany. 21, 26, 30, 77, 151

Katakis, I., Tsoumakas, G. & Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, **22**, 371–391. 45

Kelly, M.G., Hand, D.J. & Adams, N.M. (1999). The impact of changing populations on classifier performance. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 367–371, ACM, San Diego, California, United States. 35

Kennedy, K., Delany, S. & Mac Namee, B. (2011). A Framework for Generating Data to Simulate Application Scoring. In *Credit Scoring and Credit Control XII, Conference Proceedings, Credit Research Centre, Business School, University of Edinburgh*, CRC. 60

Kifer, D., Ben-David, S. & Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, vol. 30, 180–191, VLDB Endowment. 3, 54, 115

Klinkenberg, R. (1999). Learning drifting concepts with partial user feedback. *Beiträge zum Treffen der GI-Fachgruppe*, **1**, 44–53. 51, 66, 76, 90

Klinkenberg, R. (2001). Using labeled and unlabeled data to learn drifting concepts. In *Workshop notes of the IJCAI-01 Workshop on Learning from Temporal and Spatial Data*, 16–24. 48, 66, 76

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, **8**, 281–300. 45, 66, 76

Klinkenberg, R. & Joachims, T. (2000). Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, 11. 3, 43

Klinkenberg, R. & Renz, I. (1998). Adaptive information filtering: Learning in the presence of concept drifts. In *Workshop Notes of the ICML/AAAI-98 Workshop Learning for Text Categorization*, 33–40, AAAI Press. 3, 38, 42, 51, 66, 76, 77, 90

Kolter, J. & Maloof, M. (2003). Dynamic weighted majority: a new ensemble method for tracking concept drift. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, 123–130. 44, 62

Kubat, M. (1989). Floating approximation in time-varying knowledge bases. *Pattern recognition letters*, **10**, 223–227. 3, 41

Kuncheva, L. (2002). A theoretical study on six classifier fusion strategies. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **24**, 281 –286. 23

Kuncheva, L. (2008). Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In *Proceedings of the 2nd Workshop SUEMA*, 5–10, Patras, Greece. 36, 37

Kuncheva, L.I. (2009). Using control charts for detecting concept change in streaming data. Tech. Rep. BCS-TR-001-2009, School of Computer Science, Bangor University, UK. 3, 43, 63, 114, 123, 133, 148

Lanquillon, C. (1999). Information filtering in changing domains. In *Proceedings of the 16th international joint conference on artificial intelligence*, 41–48. 3, 17, 26, 55, 67, 76, 77, 115, 117, 123, 147, 149, 152, 153

Lewis, D.D. & Gale, W.A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 3–12, Springer-Verlag New York, Inc. 51, 94

Lindstrom, P., Delany, S.J. & Mac Namee, B. (2008). Autopilot: simulating changing concepts in real data. In *Proceedings of the 19th. Irish Conference on Artificial Intelligence and Cognitive Science*, 21. 61

Lindstrom, P., Delany, S.J. & Mac Namee, B. (2010a). Handling concept drift in a text data stream constrained by high labelling cost. In H.W. Guesgen & R.C. Murray, eds., *Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference*, AAAI Press. 6, 48, 49, 90

Lindstrom, P., Hu, R., Delany, S.J. & Mac Namee, B. (2010b). SVM based active learning with exploration. In *AISTATS 2010 Workshop on Active Learning and Experimental Design.* 179

Lindstrom, P., Mac Namee, B. & Delany, S.J. (2011). Drift detection using uncertainty distribution divergence. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, 604–608, IEEE Computer Society. 6, 114

Lindstrom, P., Mac Namee, B. & Delany, S.J. (2013). Drift detection using uncertainty distribution divergence. *Evolving Systems*, **4**, 13–25. 114

Manning, T. & Walsh, P. (2013). Improving the performance of CGPANN for breast cancer diagnosis using crossover and radial basis functions. In L. Vanneschi, W. Bush & M. Giacobini, eds., *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, vol. 7833 of *Lecture Notes in Computer Science*, 165–176, Springer Berlin Heidelberg. 2

Masud, M., Gao, J., Khan, L., Han, J. & Thuraisingham, B. (2008). A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, 929 –934. 48, 53, 91

McCarthy, J., Minsky, M.L., Rochester, N. & Shannon, C.E. (1955). A proposal for the dartmouth summer research project on artificial intelligence. Tech. rep., Dartmouth College. 1

Mitchell, T.M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1st edn. 1

Montgomery, D.C. (2004). *Introduction to Statistical Quality Control*. Wiley, 5th edn. 119

Mooney, R.J. & Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, 195–204, ACM, New York, NY, USA. 2

Narasimhamurthy, A. & Kuncheva, L.I. (2007). A framework for generating data to simulate changing environments. In *Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications table of contents*, 384–389. 33, 60

Nemenyi, P. (1963). *Distribution-free multiple comparisons*. Ph.D. thesis, Princeton. 86

Nishida, K. & Yamauchi, K. (2007). Detecting concept drift using statistical testing. In V. Corruble, M. Takeda & E. Suzuki, eds., *Discovery Science*, vol. 4755 of *Lecture Notes in Computer Science*, 264–269, Springer Berlin / Heidelberg. 3, 43, 48, 114

Opitz, D.W. & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research (JAIR)*, **11**, 169–198. 22, 23

Opitz, D.W. & Shavlik, J.W. (1995). Generating accurate and diverse members of a neural-network ensemble. In *Advances in Neural Information Processing Systems 8, NIPS*, 535–541, MIT Press, Denver, CO, November 27-30, 1995. 23

Pang, B. & Lee, L. (2004). A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Association for Computational Linguistics, Stroudsburg, PA, USA. 2

Peng, Y., Kou, G., Wang, G. & Shi, Y. (2011). FAMCDM: a fusion approach of MCDM methods to rank multiclass classification algorithms. *Omega*, **39**, 677 – 689. 82

Porter, M. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, **14**, 130–137. 29, 71

Quinlan, J. (1986). Induction of decision trees. *Machine learning*, **1**, 81–106. 16, 29

Rokach, L. (2010). *Pattern Classification Using Ensemble Methods*. World Scientific Publishing Company. 22, 23

Roy, N. & Mccallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *In Proc. 18th International Conf. on Machine Learning*. 95

Ruta, D. & Gabrys, B. (2000). An overview of classifier fusion methods. *Computing and Information systems*, **7**, 1–10. 23

Salton, G., Wong, A. & Yang, C.S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, **18**, 613–620. 26

Schlimmer, J.C. & Granger, R.H. (1986). Incremental learning from noisy data. *Machine Learning*, **1**, 317–354. 61

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, **34**, 1–47. 26

Sebastião, R. & Gama, J. (2007). Change detection in learning histograms from data streams. In J. Neves, M. Santos & J. Machado, eds., *Progress in Artificial Intelligence*, vol. 4874 of *Lecture Notes in Computer Science*, 112–123, Springer Berlin / Heidelberg. 3, 55, 115, 119

Shearer, C. (2000). The crisp-dm model: the new blueprint for data mining. *Journal of Data Warehousing*, **5**, 13–22. 12

Silva, C. & Ribeiro, B. (2003). The importance of stop word removal on recall values in text categorization. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 3, 1661 – 1666 vol.3. 26, 28, 29

Stanley, K.O. (2003). Learning concept drift with a committee of decision trees. Tech. Rep. AI03-302, Department of Computer Sciences, The University of Texas at Austin. 36, 44

Tong, S. & Koller, D. (2002). Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, **2**, 45–66. 51, 99

Triantaphyllou, E. & Baig, K. (2005). The impact of aggregating benefit and cost criteria in four MCDA methods. *Engineering Management, IEEE Transactions on*, **52**, 213 – 226. 82

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. Tech. Rep. TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin. 35, 36, 38

Tsymbal, A., Pechenizkiy, M., Cunningham, P. & Puuronen, S. (2006). Handling local concept drift with dynamic integration of classifiers. In *19th IEEE International Symposium on CBMS*, 679–684. 60

Tsymbal, A., Pechenizkiy, M., Cunningham, P. & Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. *Information Fusion*, **9**, 56–68. 45

Vapnik, V. (1999). *The Nature of Statistical Learning Theory*. Springer, 2nd edn. 20

Vorburger, P. & Bernstein, A. (2006). Entropy-based concept shift detection. *Data Mining, IEEE International Conference on*, **0**, 1113–1118. 43, 54

Wang, A.H. (2010). Don't follow me: Spam detection in twitter. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, 1 –10. 60

Wang, H., Fan, W., Yu, P.S. & 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, 226–235, ACM, New York, NY, USA. 45

Widmer & Kubat (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, **23**, 69–101. 3, 41

Widmer, G. & Kubat, M. (1992). Learning flexible concepts from streams of examples: FLORA 2. In *European Conference on Artificial Intelligence (ECAI-92)*, 463–467, Vienna, Austria. 41

Widmer, G. & Kubat, M. (1998). Special issue on context sensitivity and concept drift. *Machine Learning*, **32**, 83–201. 35, 37

Woods, K., Kegelmeyer, J., W.P. & Bowyer, K. (1997). Combination of multiple classifiers using local accuracy estimates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **19**, 405–410. 24

Woolam, C., Masud, M. & Khan, L. (2009). Lacking labels in the stream: Classifying evolving stream data with few labels. In J. Rauch, Z. Ras, P. Berka & T. Elomaa, eds., *Foundations of Intelligent Systems*, vol. 5722 of *Lecture Notes in Computer Science*, 552–562, Springer Berlin / Heidelberg. 53

Xu, L., Krzyzak, A. & Suen, C. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, **22**, 418 –435. 2

Xu, Z., Yu, K., Tresp, V., Xu, X. & Wang, J. (2003). Representative sampling for text classification using support vector machines. In *Advances in Information Retrieval*, vol. 2633 of *LNCS*. 99

Yang, Y. & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, 42–49, ACM, New York, NY, USA. 21, 22, 77, 151

Yang, Y. & Pedersen, J.O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, 412–420, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 29, 30

Yang, Y. & Wilbur, J. (1996). Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science*, **47**, 357–369. 29

Zhu, X., Zhang, P., Lin, X. & Shi, Y. (2007). Active learning from data streams. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, 757–762. 3, 49, 52, 91

Zliobaite, I. (2009). Learning under concept drift: an overview. Tech. rep., Vilnius University. 33, 34, 35, 38

Zliobaite, I. (2010). Change with delayed labeling: When is it detectable? In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, 843–850, IEEE Computer Society, Washington, DC, USA. 3, 49, 55, 115

Zliobaite, I. (2011). Combining similarity in time and space for training set formation under concept drift. *Intelligent Data Analysis*, **15**, 589–611. 180

Zliobaite, I., Bifet, A., Pfahringer, B. & Holmes, G. (2011). Active learning with evolving streaming data. In D. Gunopulos, M. Vazirgiannis, D. Malerba & T. Hofmann, eds., *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part III*, ECML PKDD'11, 597–612, Springer-Verlag, Berlin, Heidelberg. 48, 49, 52, 147, 148

Zliobaite, I., Bifet, A., Gaber, M., Gabrys, B., Gama, J., Minku, L. & Musial, K. (2012). Next challenges for adaptive learning systems. *SIGKDD Explorations Newsletter*, **14**, 48–55. 5