A Context-Aware Approach for Handling Concept Drift in Classification

by

Lida Barakat

This thesis is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Department of Management Science

the

Management School

May 2018

Abstract

Adapting classification models to changes is one of the main challenges associated with learning from data in dynamic environments. In particular, the description of the target concept is not static and may change over time under the influence of varying environmental conditions (i.e. varying context). Although many adaptive learning approaches have been proposed in the literature to address such changes, these are limited in terms of the extent to which the contextual aspects are *explicitly* identified and utilised. Instead, existing approaches mostly rely on monitoring the effects of drift (in terms of the degradation of the classifier's performance). Given this, to achieve more effective concept drift management, we propose incorporating *context awareness* when adapting the classification model to changes. Explicit identification and monitoring of the contextual aspects enable capturing the causes of drift, and hence facilitating more proactive adaptation. In particular, we propose an information-theoretic-based approach for systematic context identification, aiming to learn from data the contextual characteristics of the domain of interest by identifying the context variables contributing to concept changes. Such characteristics are then utilised as important clues guiding the adaptation process of the classification model. Specifically, knowledge of contextual variables are exploited to select the most relevant data for retraining the model via a data *weighting* model, and to signal the need for data re-selection via a change *detection* model. The experimental analyses on simulated, benchmark, and real-world datasets, show that such explicit identification and utilisation of contextual information result in a more effective data selection and drift detection strategies, and enable to produce more accurate predictions.

Declaration

I verify that this thesis is my own work, and that it has not been submitted for the award of a higher degree elsewhere. I also verify that the word length of this thesis does not exceed the permitted maximum of 80,000 words.

Lida Barakat

May 2018

Acknowledgments

First and foremost, I would like to express my gratitude to my supervisors Dr. Nicos Pavlidis and Dr. Sven Crone for their support, guidance, and patience throughout my PhD study. I am also very grateful to Professor Robert Fildes for his insightful comments during the annual progress review meetings.

I would also like to acknowledge with much appreciation the crucial role of Damascus University, providing me with the basic knowledge to pursue a PhD degree in the first place, and supporting me financially throughout my studies.

Additionally, I owe special thanks to the members of the Department of Management Science at Lancaster University Management School. I would like to especially thank Gay Bentinck for her assistance throughout my study, and her help in printing and submitting this work.

I would also like to express my sincere gratitude to Fr. Andrew Phillips for his spiritual support and prayers.

Last but not least, my love and appreciation go to my beloved family, my father Wafik, my mother Olga, my sisters Lina and Linda, my nephew and niece, Aleksander and Anna, and my brothers-in-law Samhar and Stewart. This work would not have been possible without your endless love, support, and faith in me.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

1.1 Introduction

In supervised classification problems, available historical labeled examples (input-output pairs) are normally utilised in order to learn a *target concept*, which refers to the underlying function between the variable to be predicted and the respective input data. The continuous flow of information in many domains requires the ability of the classification model to incorporate new data so that it can update its concept description and make more accurate predictions. In a static environment, where the target concept does not change, all available data observations can be incorporated into the classification model to improve its performance. However, in the real world, the target concept is not static and may change over time under the influence of varying conditions, a phenomenon referred to as *concept drift*. In the presence of such a drift, the previously seen data examples may not remain relevant. Hence, to be able to adapt to drift, it is essential for the classification model not only to accommodate new information, but also to have some data selection mechanism to assess which data are relevant to be incorporated when adapting the classifier. This task, however, is far from trivial since

drift can occur at any time, and its type (gradual, abrupt, etc.) cannot be known in advance. This poses a challenge on how to select *relevant* and *sufficient* training data to adapt the model, which is a vital issue since the ability of the classification model to make accurate decisions is highly dependent on the data used for the learning process. In particular, one of the most important challenges that many researchers try to solve when building a classifier in drifting environments is identifying the appropriate training data size (window size). This requires dealing with two conflicting properties: *stability* and *plasticity* [62]. More precisely, a small window will be able to capture changes (plasticity), but will make the model more responsive to randomness and will not contain a sufficient number of examples (data observations) for a stable concept description [21]. A large window, on the other hand, will ensure more stable concept description and will be less responsive to randomness (stability). However, it will decrease the classifier's ability to react to concept drift, especially for abrupt changes [21].

Existing approaches for handling concept drift can be mainly classified into two types of approaches [17], continuous and based on change detection. In the first type, a continuous drift is assumed, constantly applying some criteria to discriminate between data samples, without attempting to identify when the drift actually occurs (e.g. [10, 12, 13, 14, 15]). In the second type, a concept drift is detected, typically based on monitoring the performance of the classifier (e.g. [3, 6, 7, 21, 100]), and relevant data are identified accordingly.

These existing approaches suffer from a number of limitations as detailed in the following section. Most importantly, these approaches ignore important clues when handling concept drift. In particular, it is generally agreed in the literature that contextual changes (i.e. changes in environmental conditions) are usually the main reason behind concept drift [21, 89, 19]. Such contextual changes, for example, may correspond to

changes in economic conditions for financial prediction, the time of the year for weather prediction, season and time of the week for electricity price prediction, etc. However, the majority of existing approaches do not consider such contextual factors despite their importance in providing useful and relevant information for the purpose of adaptation.

The rest of the chapter is organised as follows. Section 1.2 discusses the limitations in existing work that the thesis aims to address. The research aims and contributions are provided in Section 1.3. Finally, an outline of the reminder of the thesis is presented in Section 1.4.

1.2 Limitations in Existing Work

The main limitations of adaptive learning approaches existing in the literature can be summarised with respect to data selection and how the change is detected as presented below.

1.2.1 Data Selection

One of the main challenges associated with learning in the presence of drift is having an appropriate data selection mechanism that will enable the classification model to adapt to changes (choose the right data examples for adjusting its concept description either once a change is detected or on a continuous basis). The main assumption in most approaches that have been developed to address concept drift is that the importance of the examples decreases with age, i.e. recent examples are the most relevant while older examples should be forgotten eventually. This assumption, however, is not necessarily true. This is because, changes in concept usually occur due to contextual changes, which could reappear over the course of time triggering the reappearance of the same concept so that older examples become relevant again.

Moreover, in the case of insufficient number of data examples coming after the change, considering only the most recent data will result in including examples coming before and after the change. Such incorporation of the most recent prior to the change data leads to inaccurate classification decisions. This is because the classification rules will be influenced by past irrelevant data, and thus will not be able to capture sufficiently the current state of the target concept. On the other hand, data in the past could be more relevant compared to the most recent data, especially in the case of recurring concepts. Given this, decisions as to which examples to include should be based not only on time, but other criteria have to be taken into account as we explain in Section 1.3.

1.2.2 Drift Detection

Given that drift detection is one of the main steps towards handling concept drift, several approaches have been proposed for this purpose. However, existing approaches mainly focus on monitoring the performance accuracy of the classification model to identify the point where the drift has occurred, which is a costly process in terms of performance degradation. This is because it usually requires a sufficient number of misclassified instances before drift can be confirmed. Specifically, whenever a misclassification occurs, three different possibilities regarding the source of this error have to be considered: *first*, it is a random noise (unpredictable fluctuations in the concept of interest) and should be ignored; *second*, it signals a concept drift, and hence the training data must be revised to adapt the classifier to the new concept; and *finally*, it is another representative example of the same underlying concept, and thus should be added to the existing knowledge of the classifier (old data is still relevant to the current concept). The latter case is referred to in literature as *virtual* drift [67].

Based on this, there is a need for a better drift detection approach that allows a faster (less costly) and more reliable method to recognise drift.

1.3 Research Aims and Contributions

To address the above issues of concept drift adaptation, we propose incorporating context awareness when adapting the classification model to changes driven by the fact that context changes are usually considered the main reason behind concept drift [9, 19, 21].

In the thesis, we focus on *explicitly* identifying context (i.e. context is defined by a set of variables) for the adaptation process. That is, we assume that knowledge about potential contextual information (derived from the background knowledge about the domain) can be characterised as certain variable(s). Given such information, our goal is to deduce contextual variables that actually affect the concept of interest from the set of candidate ones. Such an approach, however, may not be applicable in some domains, where context cannot be explicitly represented as certain variables due to the absence of background knowledge or inability to quantify context. In such domains, context may be captured *implicitly*, by monitoring its effects (mostly based on the predictive accuracy of the classifier), and trying to group data samples accordingly, with no actual access to the contextual information. Existing approaches for doing so include monitoring the predictive ability of the input variables [32], data clustering [9, 77, 80, 94], and utilising the frequency of attribute values occurrence [78, 79]. A detailed discussion of these approaches is provided in Section 2.4.2.2, page 30. Whilst *implicit* context modelling offers some flexibility in terms of selecting more relevant training examples (rather than simply relying on the training example's age) and dealing with recurrent concepts, the possibilities of its utilisation for the adaptation process remain limited. On the other hand, *explicit* context modelling (the focus of this thesis) offers a more proactive and prompt approach to respond to changes, without the delays associated with monitoring classification performance. In particular, concept changes can be signalled promptly by monitoring changes in context variables, and the classifier

can be promptly adapted with the most relevant data by relying on explicit evidence (explicit similarity among context variables).

It is worth mentioning that perfect contextual knowledge is not assumed in the thesis. Rather, our experimental analysis shows that the ability to identify and utilise even imperfect contextual knowledge contributes to better adaptation to drift (improves the performance of the classifier) in comparison to context independent approaches.

We first propose a context learning model that aims to explicitly identify the variables that actually represent contextual information for the concept of interest. The context variables identified are then utilised in adapting the classification model for data selection (via a context-based data weighting approach) and for concept change detection (via an adaptive context-based multi-model learning algorithm). More details are provided below.

1.3.1 Context Identification Model

Recently, learning with respect to context started to receive increasing attention [9, 32, 94]. Yet, these approaches mostly rely either on monitoring the implicit context, or the contextual variables are assumed to be known a priori, with no identification method being suggested. In response, we propose a systematic approach for explicit context identification by identifying the context variables contributing to concept changes. The contextual properties of a candidate variable is measured based on its ability to discriminate between occurring concepts. For this purpose, the concepts of uncertainty and information content from Shannon's Information Theory [24] are utilised. A computational solution for these measures based on k-nearest neighbour estimator [29] is also presented. Moreover, to account for any possible correlations among context variables, a context selection algorithm based on the proposed context identification criteria is provided, along with a solution for high dimensionality problems.

Part of this work has been published as [1]:

L. Barakat. Context Identification and Exploitation in Dynamic Data Mining: an Application to Classifying Electricity Price Changes. In: Bouchachia A. (eds) Adaptive and Intelligent Systems. Lecture Notes in Computer Science, vol. 8779, pages 80-89, Springer, Heidelberg, 2014.

In this thesis, the identification and selection of contextual information are performed prior to classification, assuming the availability of pre-existing training data from which context can be deduced. This imposes restrictions on the applicability of the proposed approach in domains where training data becomes available only gradually, calling for an incremental learning approach. Whilst incremental context learning is out of the scope of this thesis, the information theoretic measures proposed can be extended to enable such learning [83, 123, 124, 125].

1.3.2 Context Exploitation Model - Example Weighting

As stated earlier, most approaches proposed for addressing concept drift limit the space of the candidate examples for adjusting the classification model to those that fall within the window of the most recent data, thus neglecting many older possibly relevant examples. To overcome this, we propose utilising knowledge of relevant contextual information to facilitate the selection of more relevant training data for the classification model. Specifically, we propose to *weight* the training examples according to how similar their context is to the current context, thus controlling the contribution of each example in estimating the classification model parameters according to its relevance for the current situation.

To measure the contextual similarity, two approaches are distinguished. In the first approach, we present a simple context-based example weighting function, where the

distance between context variable values is measured based on the numerical differences in values. In the second approach, an improved context-based example weighting function is provided, where the similarity between context variable values is determined based on their conceptual similarity (whether the values belong to the same concept or not).

Part of this work has been published as [2]:

L. Barakat. A Context-Driven Data Weighting Approach for Handling Concept Drift in Classification. In Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015, Springer, Wroclaw, pages 383-393, 2015.

1.3.3 Context Exploitation Model - Drift Detection

In this part, we propose an adaptive learning model equipped with a drift detection strategy that relies on monitoring the factors causing the drift (i.e. monitoring changes in relevant contextual variables) rather than monitoring its consequences (as in existing approaches). Such utilisation of contextual information will enable a faster (less costly in terms of performance degradation) and more reliable method to recognise drift. Moreover, we propose to utilise contextual knowledge for exploiting the possibility of concept recurrence during the learning process by reusing a previously learned classification model. The ability to exploit concept recurrence will enable a faster adaptation after a change point (as opposed to re-learning the predictive model from new examples), and consequently improves the predictive accuracy. Two realisations of the adaptive learning model are proposed: a *purely contextual model* and a *hybrid model*. In the former version, drift detection and the recognition of concept recurrence are only based on the utilisation of contextual information. That is, change detection is based on monitoring any deviations in the values of the context variables. Similarly,

the value(s) of the context variable(s) are utilised for recognising the recurrence of a previously observed concept, and the selection of the appropriate classifier (in terms of context similarity). In the hybrid model, on the other hand, drift detection strategy and concept recurrence are combined with additional measures to boost the performance of the purely contextual model in cases where perfect contextual knowledge may not be available.

One might argue that instead of utilising context for data selection and change detection, context variables could alternatively be incorporated as additional input variables into the classification model. This, however, may unnecessarily increase the problem dimensionality, and achieve lower prediction accuracy compared to the external utilisation proposed (as demonstrated by our results), especially that context variables are not relevant for prediction during periods of concept stability (potentially long periods).

1.4 Thesis Outline

This thesis is organised as follows. Chapters 2 and 3 present a detailed review of the related work and the experimental design utilised throughout the thesis, respectively. The proposed context identification model is presented in Chapter 4. Context exploitation model with example weighting is explained in Chapter 5. Chapter 6 details the proposed approach for context-aware adaptation based on drift detection. Finally Chapter 7 concludes the thesis along with outlining future research directions.

Chapter 2

Literature Review

Having set the research goals in Chapter 1, we provide in this chapter a review of adaptive learning approaches that have been proposed in the literature for addressing concept drift in classification. We start with a brief overview of the phenomenon of concept drift in Section 2.1, followed by a review of different concept change types in Section 2.2. The application domains with concept drift problem are presented in Section 2.3. A general review of existing approaches is given in Section 2.4. Finally, Section 2.5 concludes this chapter.

2.1 The Notion of Concept Drift

To build a classification model, which is able to predict the class membership using the values of given input variables, a data sample with known target classes is required for training this model. This data (the training set) consists of a set of input variables (attributes) with the corresponding class labels (the outputs). The rules induced by the classification model depend on the existing relationship between the class labels and attribute values provided in the training set. However, the relation between the

input and output variables discovered in the training data may not match that in the application data (data unseen previously to which the classification model will be applied). This difference can occur due to the dynamic environment under which the data of interest is generated, which leads to the problem of concept drift. Concept drift has received increasing attention from researchers in different fields and has been studied under various names, e.g. population drift, non stationarity, covariate shift, temporal evolution [65, 64, 15]. The phenomenon of concept drift can be illustrated using Bayes' theorem for posterior probability. Let's assume that X represents a vector of input variables, and Y is the target class. The decision of classifying X to Y will depend on the class posterior probability, which is given as follows [63]:

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} \tag{2.1}$$

where $P(Y|X)$ is the probability of Y given input vector X, $P(X|Y)$ is the class conditional distribution of the input variables X, $P(Y)$ is the class prior probability, and $P(X)$ is the probability of observing attribute vector X (this is the same for all classes $y \in Y$, and thus could be omitted).

In the presence of concept drift, the posterior probability $P(Y|X)$ varies over time, i.e. $P_t(Y|X)$ may not be equal to $P_{t+1}(Y|X)$. In other words, drift causes changes in the decision boundaries of the classifier over time. For example, considering a binary classification problem with two dimensional input vector, Figure 2.1 illustrates a change in the distribution of the attributes with respect to the two classes, resulting in a change in the classification rules for the discrimination between these two classes (i.e. a change in decision boundary). Such changes in $P(Y|X)$ can occur because of the changes in the prior probability $P(Y)$ and/or changes in the class conditional distributions of the input variables $P(X|Y)$ (see Equation 2.1).

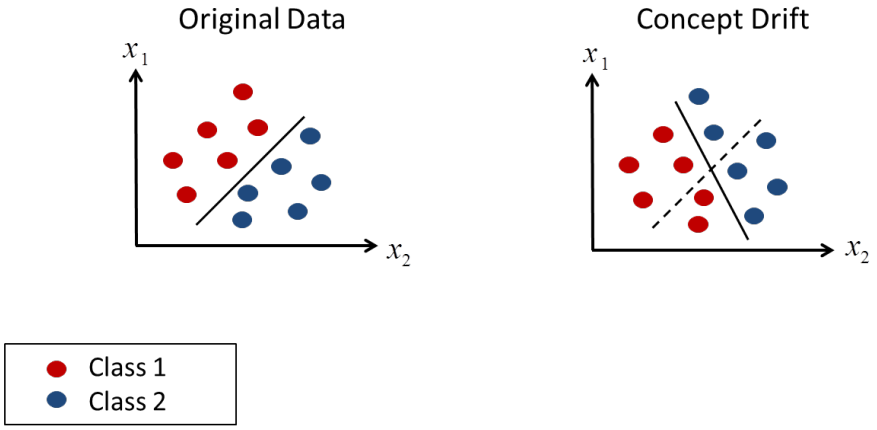


FIGURE 2.1: Graphical representation of concept drift.

2.2 Types of Concept Changes

Two types of concept changes are normally distinguished in the literature [5]: *real* and *virtual* (see Figure 2.2). In real concept drift, changes occur in the class conditional distribution (i.e. changes in $P(Y|X)$). Such changes cause the decision boundery of the classifier to change (see Figure 2.2). In virtual concept drift, on the other hand, changes occur in the distribution of the input variables (i.e. changes in $P(X)$) with no associated changes in $P(Y|X)$. Some authors regard virtual drift as changes that only occur at the classifier level due to imperfect representation of the real world (i.e. there is no change in reality) [67]. In general, both real and virtual changes occur together [5].

In this thesis, we focus on real concept changes, which are not visible form changes in the data distribution of the input variables. Note that, the adaptation necessary to address real drift can also be applied to address virtual drift but not the other way around [5].

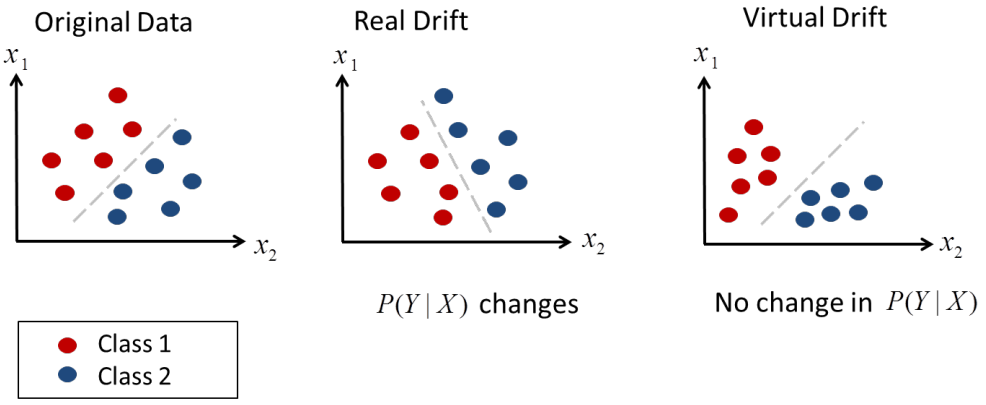


FIGURE 2.2: Types of concept changes: real and virtual.

Concept changes can further be divided in terms of the magnitude of the change into two types [99]: global and local. With global concept drift, changes occur in the whole space of the input variables, while in local drift, only certain regions of the input variable exhibit changes. Local concept changes usually take longer time to be detected due to the scarcity of the number of examples belonging to the new concept [96]. In the thesis, the focus is on addressing global concept drift, and hence the literature review is also tailored to global changes.

Furthermore, taking into consideration the pattern of the concept change, three types are normally distinguished (see Figure 2.3) [5]: *abrupt* (sudden), *gradual* and *recurrent*.

In abrupt changes, the drift occurs immediately with one concept replacing the previous one (see Figure 2.3(a)). For example, assuming that we have a news recommender system, the person sudden interest in house prices is considered as an abrupt change [100]. In this type of drift, changes occur at certain time steps with static distribution between consecutive changes.

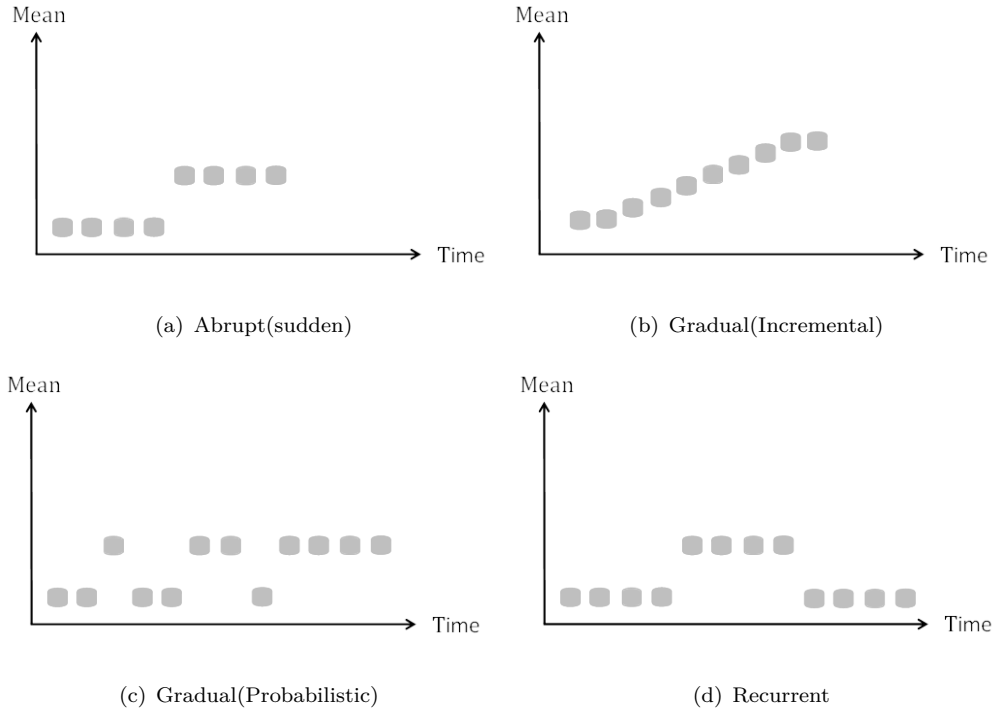
In gradual changes, the drift occurs at each time step at a certain rate (depending on the speed of drift) [15] (see Figure 2.3(b)). This type of drift is usually more difficult to

detect compared to abrupt changes. This is because the changes between time steps are normally very small such that the drift is only observed over a long period of time [100]. Another type of gradual drift that is usually referred to in the literature is probabilistic drift. In this type of drift, more than one concept is active at the same time: the current concept and the new one but with different sampling probabilities. That is, at the beginning of the change, the probability of sampling from the new concept is low, while the probability of sampling from the current concept is high. As time passes, the probability of sampling from the new concept increases (see Figure 2.3(c)). In the example with the news recommender system, a gradual drift is a growing interest in the real estate news as the user increases its interest in buying a flat.

Finally, recurrent concept drift refers to the change that reappears (either abruptly or gradually) after some time interval (see Figure 2.3(d)). Such type of drift is different from seasonality as it is not necessarily periodic, and it is not known when it would reappear [100].

It is worth mentioning that the above categorisation of change types is not exhaustive. For example, in some surveys [102, 97, 96] concept changes are further categorised according to predictability and some other factors.

Each type of change may require a different adaptation strategy. For example, single models are utilised when abrupt concept changes are expected. This is because, the old model becomes instantly irrelevant for the new concept. On the other hand, in the case of gradual and recurrent changes, the most common adaptive strategy is to keep multiple models [65]. Hence, the proposed approaches for handling concept drift in the literature usually assume certain change types and adapt the learning models towards these changes (these assumptions may not always be stated explicitly). Based on this, the effectiveness of the proposed approaches are normally tested on simulated data or on real dataset with simulated drift for which the adaptive approach is proposed.



(a) Abrupt(sudden) (b) Gradual(Incremental)

(c) Gradual(Probabilistic) (d) Recurrent

FIGURE 2.3: Patterns of concept changes assuming one dimensional dataset (changes occur in the mean of the dataset).

2.3 Applications

Adaptive learning models attempting to tackle concept drift have been applied in a variety of domains, examples of which include flight simulation [9], information filtering [80], vowel recognition [32], changes in electricity price prediction [6], consumer credit classification [66, 15], medical diagnosis [103], etc.

Different change types exist in different application domains. Hence, one solution for handling concept drift that is suitable for all applications is not feasible. Moreover, each application has certain properties that require corresponding characteristics from

the adaptive learning model [104]. More details about these properties are provided below.

Some application properties are related to the data under consideration. This includes the data type (binary, numerical, categorical, etc.), the presence of missing values, the level of dimensionality, the presence of noise, how the data is organised (e.g. sequential, time series), the shape of the incoming data (e.g. stream, batches), and the availability and accessibility of the data (only single access to the data, or the ability to re-access the data). Moreover, there are properties that characterise the environment in which the adaptive model will operate. This includes the assumptions with regard to the type of change (e.g. abrupt, gradual, recurrent), and how predictable the change is (unpredictable, somewhat predictable, predictable). Other important application-specific properties are related to the operational settings of the learning task of interest. With this regard, four factors are normally considered. The first factor is the label availability. In some applications, the label becomes available in the next time step (e.g. in food sales predictions) or on demand (e.g. user preferences). On the other hand, in applications such as credit scoring, the outcome of the decision may become available years after the decision of granting the credit is made. The second factor is the speed of the decision making. In some applications, the decision has to be made immediately (e.g. in fraud detection), while other applications have more flexibility (e.g. in credit scoring). The Third setting is the cost of making false prediction, which is important in choosing the evaluation metrics of the adaptive model. In some applications, only the prediction accuracy of the model is important (e.g. in online mass flow prediction). Other applications, in addition to the prediction accuracy, require timely and accurate change identification (e.g. in food sale prediction). The last factor is related to the ground truth label (it may be defined based on objective rules, it may be defined based on personal opinion, or it may not be defined at all or too costly to define).

In [5], the applications with concept drift problems are grouped into four categories: monitoring and control, personal assistance and information, management and strategic planning and ubiquitous environment applications. Monitoring and control applications focus on monitoring certain automated processes. Two application tasks are distinguished: prevention and detection tasks and monitoring for management tasks. The main source of drift are adversary activities, which usually occur abruptly. Examples of these applications include intrusion detection in computer security [105], fraud detection in finance [106], industrial monitoring [107], and service monitoring [108]. The input data usually come as a stream in large volumes, and the decision needs to be made immediately.

The second application area is personal assistance and information. The tasks here are divided into three categories [104]: individual assistance for personal use (such as news classification [109] and document classification [110]), business assistance (such as customer profiling for marketing [111]) and assistance for specified information (such as recommender system for project management [112]). The main source of drift in these applications is changes in user preferences over time. The cost of false prediction is lower compared to other applications.

The third category of applications is management and strategic planning. The tasks here are predictive analytical such as evaluation of credit worthiness [113], diagnostics in medical research (e.g. [114]), and biometric authentication (e.g. [115]). The source of drift in these applications are usually changes in the environment (e.g. economic conditions,). Here the amount of available data is limited, and the true labels are normally available (labels may be delayed with certain application tasks). The decision speed is lower compared to other application as the prediction accuracy is the main focus in these applications since the cost of false prediction is high.

Finally, ubiquitous environment application tasks include moving and stationary systems, which interacts with changing environments. Examples include robot vehicles [116], smart household appliances [117], and computer games [119]. The complex environment is considered as a source of drift in this application area.

In Figure 2.4, we provide a summary of the main properties of the above application tasks [118].

Properties	**Monitoring and Control**	**Personal Assistance and Information**	**Management and Strategic Planning**	**Ubiquitous Environment**
Data Volume	High	Medium	Medium	High
Incoming Data	Stream	Batches	Stream	Stream
Drift Type	Abrupt	Gradual	Abrupt/Gradual	Abrupt/Gradual
Label Availability	Fixed lag	On demand	Real time	Fixed lag
Decision Speed	High	Medium	Low	High
Cost of False Prediction	Medium	Low	High	High
Accuracy	Approximate	Approximate	High	High
Past Data Access	Yes/No	Yes	Yes	No

FIGURE 2.4: Applications Properties.

In this thesis, we contribute mainly to the management and strategic planning applications, where the source of drift is changes in the environment of the task under consideration. As we will see in Chapter 5, the proposed approach assumes the availability of time for decision making given that computationally expensive models (to achieve high prediction accuracy) are acceptable in this domain [65].

2.4 Classifier Adaptation in Response to Concept Drift

Existing adaptive learning approaches for handling concept drift can be mainly classified (in terms of how adaptation is achieved) into continuous adaptation and change detection based adaptation [17]. The former assumes continuous drift, and updates the

classification model on data arrival without attempting to identify when drift occurs. The latter approach, on the other hand, involves detecting changes in the target concept, and then adapting the classification model in response to it. These approaches are discussed next.

Note that, as indicated previously, the presented work and literature review are focused on addressing real global concept drift, and hence the techniques presented may not be suited for other types of changes such as local drifts. For example, the majority of the approaches apply certain types of windowing, which assume the drift affects the whole input variables space. Such approaches are not suitable for detecting local concept drift, where changes affect only part of the example space. Moreover, as we will see in the following sections, addressing real drift depends on the performance accuracy of the classification model. On the other hand, such feedback about the performance of the classifier is not necessary for addressing virtual drift [5], which is also out of the scope of this thesis.

The main characteristics of the papers presented in this thesis with respect to the change addressed and the application domain are summarised in Figure 2.5. The last column in Figure 2.5 also explains how each paper is validated. It is worth mentioning that evaluating the performance of the learning models under concept drift is still an open issue [49]. One of the main challenges is that models continuously evolve over time [49]. Unlike static environments with fixed sample sizes and stationary distributions, the data in dynamic environments arrives continuously and is generated from non-stationary distributions. Given this, standard evaluation methods such as cross-validation, leave-one-out and bootstrap are not applicable in such settings [49]. Hence, in the presence of concept drift, classification models are normally evaluated using two procedures: holdout an independent test set and prequential analysis [49]. More details about these procedures and the evaluation metrics are explained in Chapter 3 (Section 3.3).

Papers	Implementation Domain	Change Addressed	Recurrency Addressed	Validation
[12]	Personal Assistance and Information	Gradual	No	- Benchmark and real datasets; - Evaluation metric: a time plot of the predictive accuracy computed utilising holdout an independent test set; - The results are averaged over 10 runs.
[15][66]	Management and Strategic Planning	Abrupt and gradual	No	- Artificial and real datasets; - Evaluation metrics: time plots of H-measure and application-specific measure. Both measures are computed utilising holdout an independent test set; - The results are averaged over 100 runs; - The significance of the results are tested utilising piared t-test with 0.01 significance level.
[77]	Personal Assistance and Information	Abrupt	Yes	- Real dataset; - Evaluation metric: a time plot of the predictive accuracy computed utilising prequential analysis with a sliding window; - The significance of the results are tested utilising t-test with 0.01 significance level.
[80]	Personal Assistance and Information	Abrupt and gradual	No	- Real dataset with simulated drift; - Evaluation metrics: a time plot of the predictive accuracy computed utilising prequential analysis with a sliding window. Overall recall and precision measures are alse presented; - The results are averaged over 4 runs.
[21]	No real dataset utilised	Abrupt and gradual	Yes	- Benchmark and artificial datasets; - Evaluation metric: a time plot of the predictive accuracy computed utilising holdout an independent test set; - The results are averaged over 10 runs.
[6][7]	Management and Strategic Planning	Abrupt	No	- Benchmark and real datasets; - Evaluation metric: a time plot of the error rate computed utilising prequential analysis with a sliding window.
[3]	Management and Strategic Planning	Abrupt and gradual	No	- Benchmark and real datasets; - Evaluation metrics: a time plot the predictive accuracy computed utilising prequential analysis with a sliding window. The following overall measures are also presented: recall, precision, false positive rate, false negatve rate, mean delay in change detection and runtime.
[32]	Personal Assistance and Information/Management and Strategic Planning	Abrupt and gradual	Yes	- Benchmark and real datasets; - Evaluation metric: a time plot of the predictive accuracy computed utilising prequential analysis with a sliding window; - The results are averaged over 10 runs.
[68]	Monitoring and Control/Management and Strategic Planning	Abrupt and gradual	No	- Benchmark and real datasets; - Evaluation metrics: a time plot of the predictive accuracy computed utilising holdout an independent test set (with 0.95 confidence intervals). The overall average area under the curve (AUC) measure is alse presented; The results are averaged over 50 runs.
[82]	No real dataset utilised	Abrupt and gradual	No	- Benchmark datasets; - Evaluation metric: a time plot of the predictive accuracy computed utilising holdout an independent test set; - The results are averaged over 500 runs.
[84]	Monitoring and Control / Management and Strategic Planning	Abrupt and gradual	Yes	- Benchmark and real datasets; - Evaluation metrics: time plots of the error rate and Q-statistic[49]. Both measures are computed utilising prequential analysis with a sliding window.
[69]	Management and Strategic Planning	Abrupt	No	- Artificial, benchmark and real datasets; - Evaluation metrics: the average overall distance between the true probability (that generates the data) and the estimation, the overall predictive accuracy and the following change detection specific measures: the rate of false positives, the rate of changes detected and the mean time until change detection.
[70]	Personal Assistance and Information	Abrupt and gradual	No	- Real datasets with simulated drift; - Evaluation metrics: a time plot of the predictive accuracy computed utilising prequential analysis with a sliding window. Overall recall and precision measures are alse presented; - The results are averaged over 10 runs.
[78]	Personal Assistance and Information	Abrupt and gradual	Yes	- Benchmarkk and real datasets ; - Evaluation metrics: a time plot of the predictive accuracy computed utilising prequential analysis with a sliding window.
[83]	Monitoring and Control	Abrupt	No	- Artificial and real datasets; - Evaluation metrics: the overall predictive accuracy and computational complexity.
[17]	Monitoring and Control	Abrupt	No	- Artificial and real datasets; - Evaluation metrics: oveall values for the following change detection specific measures: the rate of false positives, the rate of true positives, recall, presicion, and the number of examples required for change detection.
[67]	No real dataset utilised	Abrupt	Yes	- Benchmark and artificial datasets; - Evaluation metric: a time plot of the predictive accuracy computed utilising holdout an independent test set; - The results are averaged over 10 runs.
[28]	Management and Strategic Planning	Abrupt and gradual	Yes	- Benchmark and real datasets; - Evaluation metric: a time plot of the predictive accuracy computed utilising prequential analysis with a sliding window.
[9]	Ubiquitous Environment	Abrupt and gradual	Yes	- Benchmark and real datasets; - Evaluation metric: a time plot reflecting the number of times each test example was correctly classified; - The results are averaged over 100 runs.
[77]	Personal Assistance and Information	Gradual	Yes	- Artificial dataset; - Evaluation metric: a time plot of the predictive accuracy computed utilising prequential analysis with a sliding window; - The significance of the results are tested utilising t-test with 0.01 significance level.
[94]	Management and Strategic Planning	Gradual	Yes	- Real datasets and real datasets with artifiial context; - Evaluation metrics: an overall predictive accuracy and the computational complexity. - The significance of the results are tested utilising McNemar paired test with 0.01 significance level.
[89]	No real dataset utilised	Abrupt and gradual	No	- Artificial and benchmark datasets; - Evaluation metric: a time plot of the predictive accuracy computed utilising holdout an independent test set; - The results are averaged over 10 runs.
[93]	No real dataset utilised	Abrupt and gradual	No	- Benchmark and real dataset with simulated drift; - Evaluation metric: a time plot of the predictive accuracy computed utilising prequential analysis with a sliding window.

FIGURE 2.5: Summary of reviewed papers.

2.4.1 Continuous Adaptation

The classifier in these approaches is continuously (re-)estimated using either a window of the latest observed examples or a time-based example weighting function [13, 12, 77, 80, 66, 15]. Both approaches are presented below.

In the windowing approach, the simplest implementation is to fix the size of the window, and delete the oldest example whenever a new example arrives (utilising a sliding window of a fixed size) [13]. However, choosing the size of the window requires knowledge of temporal occurrence of drift (the point in time when drift occurs) and its type (abrupt or gradual), both of which are difficult to determine in advance. For instance, when data exhibit abrupt changes, a small window is required to follow these changes [21], while with gradual changes, a larger window could be more appropriate since smaller number of past data become irrelevant [100]. Existing approaches with adaptive window size are based on change detection and are discussed in Section 2.4.2.

The second proposed approach in continuous adaptation is giving weights to each example in the training data set so that the oldest example gets the lowest weight, and the most recent example gets the highest weight. In other words, the impact of past data is continuously removed according to the time when each example was observed. To achieve this, a time varying parameter is incorporated into the classification model to control the contribution of each example in adjusting the model parameters. Examples of this approach can be found in [12, 77, 80, 66, 15].

In continuous adaptation approaches, concept change is assumed to be present at any time step, and hence no change detection mechanism is required. Consequently, there is no period of performance degradation as with the change detection based approaches (which are discussed in the next section). Despite this advantage, the assumption of continuous changes could result in forgetting data continuously even in the absence

of drift. Moreover, the relevance of training data is assumed to decrease with time, neglecting older possibly relevant examples, especially in the case of recurring concepts.

2.4.2 Change Detection based Adaptation

Various change detection methods for handling concept drift are proposed in the literature. Unlike continuous adaptation, where the main proposed adaptive strategies are context independent (a formal definition of context is provided in Section 2.4.2.2), these approaches can be classified into: context independent learning and context dependent learning. More details are presented next.

2.4.2.1 Context Independent Learning

Of the most well-known algorithms to handle concept drift is FLORA2 by Widmer and Kubat [21]. The classification rules in this algorithm are built with respect to a fixed-size window of the latest examples. These rules are updated on the arrival of new examples and on drift detection. The drift is detected by monitoring the prediction accuracy of the classifier, and the rate at which new classification rules are formulated. Whenever the drift is detected, the size of the current window is decreased by deleting the oldest 20% examples. To address concept recurrency, an extended version of FLORA2 is presented that considers the possibility of reusing existing classification rules. The basic idea is to store stable classification rules for future use. Hence, whenever a change is suspected and the window is decreased in size, all stored classification rules are retrieved and compared. Previous classification rules are reused again if they appear to match labelled examples from the current window more accurately than that of the current classification rules.

Other drift detection approaches proposed in the literature mainly depend on monitoring the performance of the classification model and on detecting changes in the data distribution. Examples of these approaches are presented below.

The drift detection algorithm (DDM) proposed by Gama et al. [6] is one of the most widely used methods for change detection. The main idea behind this approach is that if the distribution of the incoming examples is stationary, the probability of making an error will decrease or at least stabilize as more examples become available (assuming the classifier is constantly updated with these new examples). A significant increase in this probability indicates that the distribution generating the examples has changed, signalling a drift. In particular, it is assumed that the error of each incoming example represents a random variable from Bernoulli trials, and that the number of misclassified examples, denoted as E, follows a Binomial distribution. Given this, the probability of making an error, denoted as p_i, and the associated standard deviation, denoted as s_i, are computed incrementally for each incoming example according to the following formulas: $p_i = \frac{E}{i}, s_i = \sqrt{\frac{p_i(1-p_i)}{i}}$. The learner's performance at example i corresponds to $p_i + s_i$, which is compared against two registers: a warning threshold computed as $p_{min} + 2 * s_{min}$, and a drift threshold computed as $p_{min} + 3 * s_{min}$. Here, p_{min} and s_{min} are the minimum error probability and minimum standard divination, respectively, among the examples observed so far, obtained in the sequence of calculating p_i and s_i for the incoming examples. Based on this, if $p_i + s_i < p_{min} + 2 * s_{min}$ the classifier is in the *normal* state, while if $p_i + s_i \geq p_{min} + 3 * s_{min}$ the classifier is in the *drift* state. If $p_i + s_i$ is in between the above levels, a *warning* state is reached that requires more examples to either confirm the drift (the error rate increases to the drift level) or return to the normal state (the error rate decreases to the normal level). The ability of DDM approach to signal warning and drift states during the model operation is an important property for data selection when adapting the classification model to changes. This is because once the drift is confirmed, all the examples between warning and drift states

are used for adapting the model, facilitating data selection task. DDM is utilised in our analysis for the proposed hybrid learner as will be explained in Chapter 6.

A similar but gradual drift oriented detection approach is the early drift detection method (EDDM) presented by Garcia et al. [7]. Instead of monitoring the error rate of the classifier to detect drift, EDDM depends on monitoring the distance between two classification errors (i.e. the number of correctly classified examples between two consecutive classification errors). This is based on the idea that if the distribution is stationary, the predictions of the learning algorithm will improve during learning, and the distance between subsequent errors will increase. The average distance between two consecutive classification errors ($\acute{p}_i$) and its standard deviation ($\acute{s}_i$) are computed for each misclassified example and compared to the maximum distance and standard deviations ($\acute{p}_{max}$ and $\acute{s}_{max}$) registered so far during the learning process. The warning and drift states are defined (when at least 30 errors have occurred) in terms of the following equations: $\frac{\acute{p}_i+2\acute{s}_i}{\acute{p}_{max}+2\acute{s}_{max}} < \alpha$ for the warning state, and $\frac{\acute{p}_i+2\acute{s}_i}{\acute{p}_{max}+2\acute{s}_{max}} < \beta$ for the drift state. Both α and β are predefined thresholds.

Another accuracy-based approach is presented in [3], which depends on monitoring the number of misclassified examples for the currently active classifier. For this purpose, a binary variable is introduced with values 0 and 1, where 0 represents a correctly classified example, while 1 represents an incorrect classification. The values of this variable is monitored over time with respect to a sliding window by applying Shannon's entropy [24]. The entropy measure is computed incrementally and compared to its maximum theoretical possible value (in this case 1). Whenever it reaches this value, the drift is signalled.

Other approaches for change detection depend on comparing the accuracy of two classifiers with respect to two different windows: a sliding window with the most recent

examples and a reference window with older examples. For instance, Bach and Maloof [68] propose a learning algorithm that combines two classifiers: one classifier is continuously updated on data arrival since the last drift detection point (referred to as a stable learner), and the second classifier (referred to as a reactive learner) is trained using a sliding window of a fixed size. The former classifier is used to perform the classification task, while the latter to signal a change in the target concept. On arrival of each new example, both classifiers are used to predict its class and then compared in terms of the frequency at when the stable learner misclassifies the examples correctly classified by the reactive learner. If this frequency exceeds a predefined threshold, the drift is detected and the stable learner is replaced by the reactive learner for performing classification. Other examples utilising the accuracy of two classifiers to detect changes can be found in [82, 100].

Moreover, rather than monitoring the classifier accuracy, some detection methods rely on identifying changes in probability distributions. To detect such changes, various approaches are utilised. For example, in [100], the change point is detected by examining each time step for a possible change using Hotelling multivariate T^2-test. This test compares the mean of each class sample before and after the potential change point (the examined time step), and the returned p-value of the test (which corresponds to the probability that no change have occurred) is used to compute the probability of a change at that point, assuming that the mean of each target class changes independently. Another commonly used test for concept change detection is the Page-Hinkley test (PHT) [81]. According to this test, a cumulative variable, denoted as m_T, is computed at each time step by taking the difference between the observed values and their mean. In particular, at time step T, m_T is computed as follows:

$$m_T = \sum_{t=1}^{T} (x_t - \bar{x}_T - \sigma)$$

where $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$, and σ is the magnitude of allowed changes. The value of the test is computed as $PH_T = m_T - M_T$, where $M_T = min(m_t, t = 1...T)$. If this difference exceeds a predefined threshold, a change in the distribution is detected.

In the adaptive windowing algorithm (ADWIN) proposed in [69], changes in the distribution are detected based on comparing the difference in means of the classification errors between all possible sub-windows of a fixed size sliding window with recent examples. Whenever two large enough sub-windows have distinct enough means (determined according to Hoeffding bound), the change is detected, and the older sub-window is deleted from the window. A similar idea to ADWIN, but more expensive in terms of time and memory, is introduced in [70], when data is assumed to arrive in batches of examples. Their window resizing approach depends on training the classifier on all possible window sizes, starting from the most recent batch of data and increasing the window size by one batch, so that the largest possible window contains all available historical data. The optimal window size is chosen by estimating the generalization error (classifier specific leave-one-out estimator of the error rate) of each window on the last batch of data, so that the window size that achieves the minimum error is used to classify the next batch of incoming examples.

In some detection approaches, changes in the distributions are measured with respect to two different windows: a window with recent data examples and a window with older examples. To measure the difference, a number of methods are utilised such as Kullback-Leibler divergence [17] and entropy [83]. If the values of these measures exceeds a predefined threshold, the drift is detected.

The main advantage of the above proposed approaches is eliminating the need for unnecessary reassessment of data relevance (e.g continuously re-weighting data examples even if a drift did not actually occur). Moreover, identifying the change point, even approximately, provides a better indication of the most relevant data for adjusting the

classifier. For example, if an abrupt change occurs, the only data that should be considered is that which come after the change point, since all previous data (before the change) will become not relevant. However, the proposed approaches mainly depend on measuring the performance accuracy of the current classifier to detect changes, and hence suffer from performance degradation. Moreover, as with continuous adaptation approaches, newer data is favoured in adapting to changes, while older data is eventually forgotten without accounting for recurrency in concepts. The contextual conditions under which the data observations were collected are also not considered, which could vary the importance of the example to the current concept. The current efforts towards incorporating such contextual information during the adaptation process are presented next.

2.4.2.2 Context Dependent Learning

A relatively new trend that has emerged in the literature to address concept drift is learning with respect to context changes. Generally, there are many definitions of the term context. A commonly used definition is the one provided by Dey [26] that defines context as "any information that can be used to characterize the situation of an entity". Here, by context we refer to the situation and the environment of the target concept [25]. Usually, the context is domain dependent. For instance, the time of the year is the context for weather prediction rules, while location and day of the week constitute context for identifying user preferences. From the view point of concept learning, context can be seen as the information that explains changes in the target concept. For example, what is meant by a "nice weather" changes with different seasons [67]. The main attempts that exist in the literature to use contextual information in addressing the problem of concept drift can be categorized into explicit context dependent learning (i.e. context is determined by a set of variables) and hidden

(missing) context dependent learning. A summary of these approaches is presented below.

Explicit Context Dependent Learning

A formal definition of context and context dependent variables are given by Turney in [31], who was among the first to recognise the problem of context in supervised learning, motivated by the possible difference in context condition between training and application data [32]. According to this definition, a variable C is considered contextual if it is not in itself predictive, but affects the discrimination ability of some input variable(s) X, i.e. $P(Y|C) = P(Y)$ and $P(Y|C, X) \neq P(Y|X)$.

In [75], Turney provided a review of different strategies for managing explicit context information to improve the accuracy of classification models. Four heuristic strategies are identified: contextual normalisation, contextual expansion, contextual weighting, and contextual classifier selection. The contextual normalisation approach depends on scaling input variable values in the training and testing sets to reduce the variations in variables that are due to different context conditions (i.e. reduce the sensitivity to context). This is done by subtracting the mean and dividing by the standard deviation which are computed from observations with the most similar context (i.e. observations with similar values in contextual variables) [74]. In contextual expansion, contextual variables are included as additional input variables to the classification model [72, 73]. In contextual weighting, each input variable is given a weight based on its relevance to the classification task (the relevance is identified by context similarity) [72, 73]. In contextual classifier selection, contextual variables are used to train a classifier to select a relevant classifier from previously trained classifiers.

Although Turney's strategies present an attempt towards explicitly taking contextual information into consideration, the utilisation of these strategies is still limited and

not focused on adapting the classifier to concept drift. For example, the data preprocessing strategies of contextual weighting and contextual normalization in [72, 73, 74] are effective for some classification algorithms (e.g. for instance-based learning), and ineffective for others (e.g. discriminant analysis) [73]. In addition, the weighting strategy in [72, 73] does not consider varying weights with respect to the current context even though changes in context values may affect the discriminant ability of the input variables [76], and consequently their weights.

Explicit utilisation of contextual information in adapting the classifier to concept drift is exploited in the adaptive learning model Mining Recurring Concepts (MReC) proposed by Gomes et al. [28]. In this model, context is utilised for classifier selection from a pool of previously trained classifiers (a model repository) in the case of recurrent concepts. In particular, the proposed learning model consists of a base-level learner that is used for performing the classification task, and a meta-level learner that is used for context utilisation by learning the relation between trained classifiers and context (the context is represented by one variable or a vector of variables). Concept drift is detected based on DDM approach. Given this, a new classifier is trained continuously whenever a warning level is reached. If the drift is confirmed (by reaching the drift level), each previously trained classifier in the model repository is compared to the new classifier in terms of similarity of their classification decisions on the data examples arrived between the recorded warning and drift levels. Such similarity is specified by taking the average of the number of times both classifiers agreed on their decisions. If the similarity measure for any of the stored classifiers is above a predefined threshold, the recurring concept is confirmed, and the new classifier is discarded. On the other hand, if no recurring concept is detected (the similarity below the threshold for all existing classifiers), the new classifier is stored and used for performing the classification task. If a recurrent concept is confirmed, each stored classifier is assigned a score that is partially determined based on its context similarity with the most recent context,

and the classifier with the highest score is selected as a base learner. In particular, the context similarity is specified by a weighting factor that is proportional to the probability that the classifier of interest represents the current concept given particular context. To compute this probability, the meta-level classifier, utilising Naive Bayes, is trained using records consisting of context information as input variable(s) with the corresponding classifier in use as a target variable (utilising an appropriate identifier for the classifier). The context of each classifier is defined as the most frequently observed values for context during classifier's training period and utilisation. Despite the fact that MReC algorithm explicitely utilises context in the adaptation process, the contextual variables are assumed to be known a priori, with no identification method being suggested. Moreover, contextual exploitation in this approach is limited to classifier selection, without utilising context in concept change detection or in concept recurrency detection as will be presented in this thesis.

Implicit Context Dependent Learning

The main approaches that have been developed when information about context is not available can be summarised as follows: monitoring the predictive ability of the input variables [32], data clustering [9, 77, 80, 94], and utilising the frequency of attribute values occurrence [78, 79]. A summary of these approaches is presented below.

In the approach proposed in [32], it is assumed that context is hidden in the dataset, and can be defined by identifying some indicator variables. In particular, the proposed classification model consists of a base-level learner that is used for the classification task, and a meta-level learner that is used to focus the base learner on the relevant examples from the training dataset that match the current context. The context is recognised on the meta-learning level by trying to identify those attributes which characterise particular contexts when taking certain values (contextual clues). Identifying these clues

depends on detecting a significant correlation between co-occurrence of some attribute values and the predictive ability of others. Predictive attributes are determined using the chi-square test of independence with respect to a sliding window of a fixed size which is used by the base learner to perform the classification task. Similarly, contextual attributes are identified using the chi-square test of independence but with respect to the entire dataset (since the beginning of the learning process). Two algorithms with different base learners are proposed: METAL (B) with a Bayesian classifier as a base learner, and METAL (IB) with an instance-based learning algorithm (single nearest neighbour model). In METAL (B), when a new example arrives, and the meta-learner detects contextual attribute(s), a selective strategy is applied to the current window by selecting examples having the same values for contextual attribute(s) as of the new example. If no contextual attribute(s) could be detected, all data in the window is used for classification. On the other hand, in METAL (IB) algorithm, two weighting strategies are proposed. The first strategy is giving weights to the examples according to the degree of similarity (using Euclidean distance) between the new example and the examples in the current window with respect to contextual attributes. In the second strategy, the weights are given to the attributes that are predictive during the current context (attribute weighting). In the feature weighting strategy, the meta-learner uses Bayes' rule to predict the probability for each attribute being predictive given a particular context (the values of contextual attributes in the incoming example), and uses these probabilities to assign weights.

In the second attempt for dealing with a hidden context, Harries et al. [9] propose an algorithm to partition the historical training data into data intervals that share similar contexts. Context similarity is determined by the degree to which the same classification rule induced from one data interval can be applied accurately into other intervals. The data intervals are iteratively modified through a process called contextual clustering. In contextual clustering, similar intervals (from adjacent or disjoint data intervals)

are combined into one contextual cluster that corresponds to one stable concept. In particular, the training data is ordered with respect to time, and divided into initial intervals that correspond to some probable change in context. The division is done either randomly, or based on prior knowledge about the domain. These intervals are then used for building temporary classification rules (interim concept) from each interval. Specifically, each learned concept is used in classifying all available examples in the training dataset, and a score (the number of correct classifications) for each concept-example combination is computed with respect to a fixed size window surrounding each example. New contextual clusters are formulated by allocating each example to the contextual cluster that gets the highest score with its interim concept for that example, and new classification rules (concepts) are learned from the new clusters. The process iterates either for a fixed number of repetitions or when the same concepts result after each iteration. The resulting stable concepts are used in classifying new examples using a voting method, so that the concept that achieves the highest accuracy in classifying the last several examples is used for making next classification decision.

Utilising classification accuracy and applying it to group data under similar hidden context is also exploited in [77]. In this approach, a meta-level learning classifier is trained on a fixed size sliding window of the most recent data (which is assumed to represent the latest context). This classifier is used to select examples from past observations that are relevant to the current context. The relevance is determined by the degree to which this classifier correctly classifies previous examples in the training set. The accuracy for classifying each example is specified by the average accuracy the classifier achieves for a fixed number of its surrounding examples. Hence, all examples that have prediction accuracy above a particular threshold are included in the training data set of the base learner. A similar approach, but for data arriving in a batch form, is introduced in [80], which also uses a classifier trained on the most recent batch of data for weighting and selecting the most relevant (well classified) batches. However, the main

criteria for selecting the training data (the relevant batches) is based on minimising the estimated performance error of the classifier with regard to the most recent batch of data. The estimated error is determined using a special form of classifier specific leave-one-out error estimate. More precisely, the error rate of the classifier on every batch is compared to its error on the most recent batch. This error is used to exclude batches that get a significantly higher estimation error than the most recent batch, or alternatively, assign weights with respect to this error (i.e., batches with the highest error get the lowest weight and batches with the lowest error get the highest weights). In [84], a two layer model is proposed: a base model for performing classifications, and a meta-model that is used to select the most relevant model for the current concept. To detect drift, the DDM approach is utilised. Detecting implicit context (and concept recurrency) is conducted with the help of the meta-classifier that is trained on the input variables similarly to the base classifier. However, the target label for this classifier is either true or false. If an example is correctly classified by the associated base classifier, the label is true. If the example is incorrectly classified by the base classifier, the label is false. Each previously trained classifier is stored with its corresponding meta-classifier. The relevance of each classifier is determined based on the meta learner that predicts the error rate of its corresponding classifier (with respect to a sliding window), such that the classifier with the lowest estimated error is selected.

In addition, different class independent clustering methods for the implicit context are provided in [94]. The main idea behind these methods is that data in the same cluster tend to have the same context, so that clustering data enables training a certain classifier for each cluster.

The last approach for capturing the effect of a hidden context utilises the order in which input variables take certain values. This idea is presented by Mandl et al. [78] and in a subsequent work in [79]. It is assumed that the order in which examples are presented

to the learning system is not random, but in some domain order [79], and therefore could be used to reveal the hidden context. Three windows are used for each example in the training set to identify potential change points in context as follows. The first window contains the sequence of training examples since the last discovered context change. The second window contains a fixed number of the most recent examples. The third window contains the latest, non-overlapping with the second window, examples that are used to test classifiers trained on both windows. If the classifier that is trained on the smaller window achieves more accurate results, a new context is detected.

Overall, the proposed approaches in learning with unobserved context provide more flexibility in selecting the training data based on its relevance to the approximated context and less on its age, especially in the presence of recurring concepts. However, these approaches provide only approximate ways in identifying context, mainly dependent on the predictive accuracy of the classifier for determining context similarity (for example in [9, 77, 80, 78]). Moreover, most approaches use a window of a fixed size to define the most recent context, which poses the same problems mentioned earlier (fixing the window size requires assuming certain type and speed of drift, and dealing with the stability-plasticity problem). For example, in [32] the window should be large enough to increase the accuracy of the classification system, and small enough to exclude instances from different concepts in order to enable the meta-leaner to identify predictive attributes and consequently contextual clues.

Note that, in the above presented approaches for handling concept drift, only a single classifier is utilised at a certain time step to classify incoming examples. This is the main focus of this thesis. On the other hand, in some approaches, multiple classifiers are combined at the same time to classify new examples (usually by voting) [49]. This approach, which is normally referred to as ensemble learning, depends on keeping multiple classifiers trained on similar concepts. The main adaptation strategy with this

regard is by assigning weights to the output of individual classifiers to achieve the final classification decision (either combined or selected), and determining the number of classifiers within the ensemble. The weights usually depend on the performance accuracy of each classifier within the ensemble on the most recent data. Examples of these approaches can be found in [71, 20, 85, 70]

Moreover, in some approaches instead of using a window of consecutive examples, a criterion for training data selection inside the window is applied to measure the usefulness of the example for the current concept. This is usually referred to as instance-based approach. The advantage of such approach in comparison to consecutive windowing is the ability to exclude irrelevant observations inside the window such as noisy examples and examples from different concepts (e.g. the examples that arrive during the transient period between the old and new concepts in the case of gradual concept changes). The criteria utilised for measuring the usefulness of each examples mainly depend on time (i.e. the most recent examples are considered the most relevant) and on space (i.e. the position of the example in the examples space) [95, 89, 93, 101, 94]. For example, in [94], the proposed data selection approach utilises both the recency of the example and its distance with respect to the example to be classified to identify the relevance of the example for the current concept. That is, the training set consists of the examples with the smallest distances, which allows addressing recurrency in concepts. Similar approaches are proposed in [89] and [93]. However, in these approaches the distance metric is also utilised to eliminate redundant examples (examples that are positioned close to each other in the neighbourhood are considered as redundant). In addition, any example in the window with a label different from the labels in its neighbourhood is removed. Hence, in both approaches recurrency in concepts is not considered. Although utilising such approaches for data selection provides more flexibility compared to the standard windowing approaches, these approaches are still limited to selecting from the most recent examples, with no utilisation of context in the analysis.

2.5 Conclusion

Adapting classification models to concept changes is one of the main challenges associated with learning in dynamic environments, where the definition of the target concept may change over time under the influence of various context. The state-of-the-art adaptive learning approaches that have been proposed in the literature to address concept drift can be generally categorised into context independent learning and context dependent learning. In the former approach, time is considered as a measure for training example relevance, with recent examples being considered the most relevant, while older ones being forgotten eventually. Moreover, these approaches mostly rely on monitoring the effects of drift to recognise changes (in terms of the degradation of the classifier's performance over time and changes in the data distributions), without taking into consideration the contextual factors under which training examples are observed. As a result, they neglect important evidence for detecting the occurrence of a change (essential to ensure that only relevant data is captured in the learning process), and ignore situations where old observations may become relevant again (i.e. when a previously encountered concept reappears). On the other hand, the approaches to context-dependent learning, which have started to receive increasing attention, facilitate more effective adaptation. The main research efforts with this regard are tailored to keeping multiple classifiers associated with certain contexts. However, these approaches mostly rely on monitoring approximated implicit context, and remain limited in terms of the extent to which the contextual aspects are explicitly identified and utilised. In this thesis, we propose enriching classification under concept drift with explicit contextual information, and exploiting such information during the learning process to facilitate drift detection and selection of more relevant training data as introduced in the following chapters.

Chapter 3

Experimental Design

In this chapter, we present the datasets utilised throughout the thesis, along with a general framework for context simulation. The evaluation metrics and classification models used for the experiments are also presented.

3.1 Datasets

Generating data under concept drift is not a straightforward task. This is because changes can be simulated in an infinite number of ways, where it is difficult to determine the most useful and realistic settings [52].

Throughout the thesis, a number of artificial datasets are utilised with different types of changing environments. Moreover, two publicly available real-world datasets are also utilised, which have been widely used for evaluating concept drift handling systems (both exhibit concept drift and contextual characteristics, which make them suitable for the purpose of our evaluation). A summary of the main characteristics of these datasets are provided in Tables 3.1 and 3.2. The visualisation of some of these datasets

is presented in Figure 3.1. Note that, for illustration purposes, only the most predictive variable (chosen experimentally) that best illustrates the pattern present in the dataset is depicted. More details about the datasets are provided in the following sections.

TABLE 3.1: Characteristics of the real and benchmark datasets utilised.

Name	Dimension	Size	Balance	Drift Type	Severity	New Concepts	Recurrent Concepts	Noise
Email	99	1500	0.53:0.47	Abrupt	Global	2	Yes	Noise-Free
Electricity	4	2265	0.57:0.43	Gradual	Local/Global	Not known	Yes	Not known
Sine1	2	2000	0.49:0.51	Abrupt	Global	2	Yes	Noise-Free
Gauss	2	2000	0.51:0.49	Abrupt	Global	2	Yes	Noisy Examples
Stagger	3	Changing	0.45:0.55	Abrupt	Global	3	Yes	Added Noise
Hyperplane	3	1200	0.5:0.5	Gradual	Local	3	Yes	Noisy Examples
Circles	2	2000	0.81:0.19	Gradual	Local/Global	4	-	Noise-Free

TABLE 3.2: Characteristics of the artificial datasets utilised.

Name	Dimension	Size	Balance	Drift Type	New Concepts	Recurrent Concepts
Dataset1	5	Changing	0.5:0.5	Abrupt	4	No
Dataset2	5	2000	0.5:0.5	Gradual	4	No
Dataset3	Changing	2000	0.5:0.5	Abrupt	4	No
Dataset4	5	3000	0.5:0.5	Abrupt	5	Yes
Static	5	2000	0.5:0.5	No drift	-	-

3.1.1 Artificial Datasets

3.1.1.1 Simulated Framework

In this section, we introduce artificial datasets generated according to the framework proposed by Narasimhamurthy and Kuncheva [52]. The idea is to assume that data is generated according to a number of distributions. Each distribution is represented in terms of class prior probabilities $P(Y)$ and class conditional probabilities $P(X|Y)$. Here, Y is the target class label, and X is a vector of input variables. In particular, for each data generating distribution, we assume equal prior probabilities (i.e. $P(Y = y) = 0.5$, for $y \in \{0, 1\}$), while setting the class conditional probabilities to

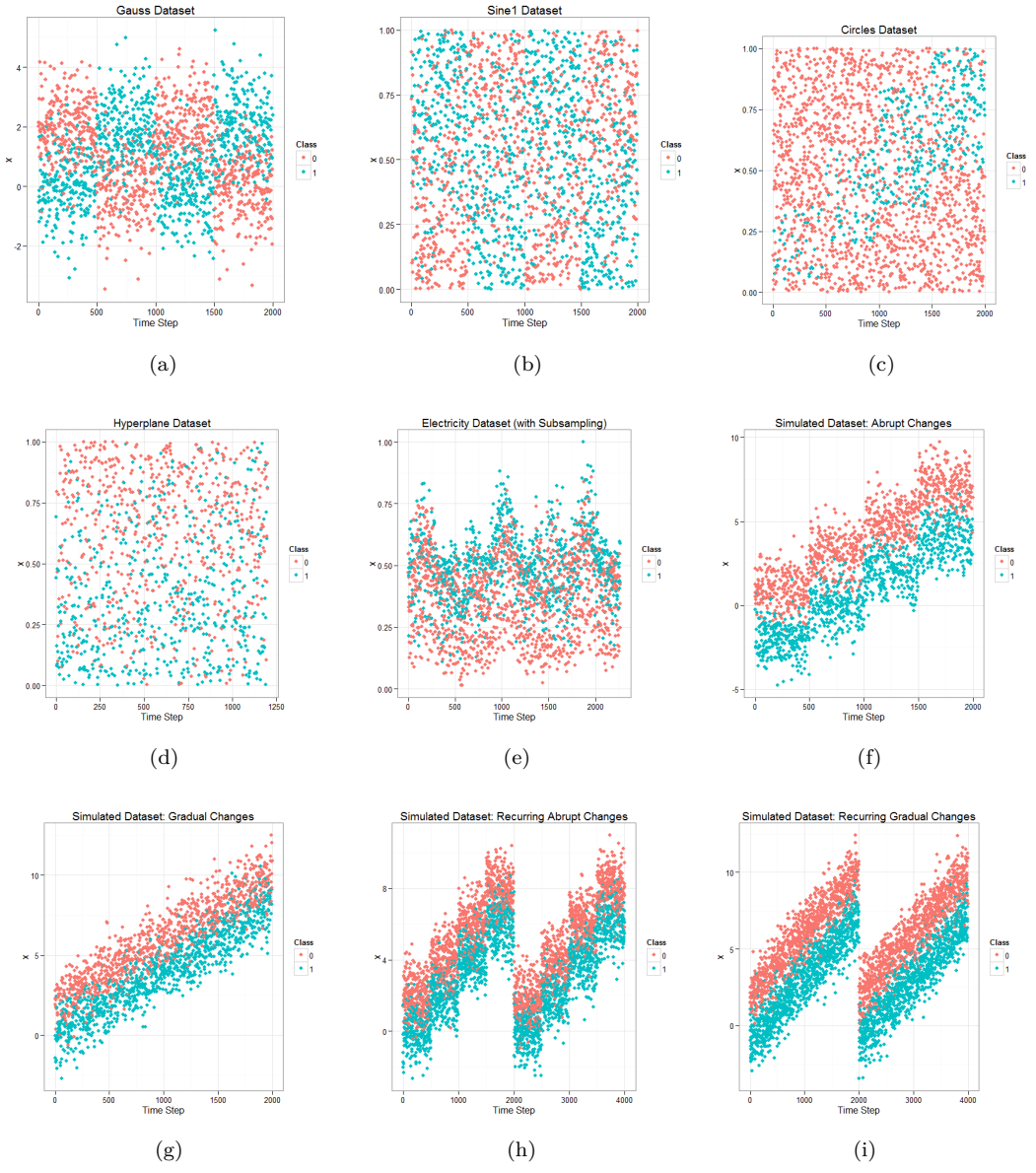


FIGURE 3.1: Visualisation of the used datasets.

be normally distributed: $X|Y \sim \mathcal{N}(\mu_y, \Sigma_y)$, where μ_y is the mean vector for the input variables given each class, and Σ_y is the associated covariance matrix. Assuming that the covariance matrices Σ_y remain the same, changes between concepts are generated by shifting the mean vectors μ_y of the input variables for each class such that: $\mu_y^t = \mu_y^{(t-1)} + \omega$. Here, ω is a deterministic quantity by which the mean vectors are changing. The values of these parameters are dependent on the type of change assumed between concepts. Specifically, according to these settings, we define the following datasets:

Static dataset. Here, the data generating distribution does not change over time, i.e. $\mu_y^t = \mu_y^{(t-1)}$ at each time step. Specifically, one data generating distribution is assumed, where the values of the mean vectors for each class are initialised as follows: $\mu_1 \in [-2, 0]^m$, and $\mu_0 \in [0, 2]^m$, where m is the input dimensionality that would be varied in our experiments.

Abrupt-change dataset. Here, μ_y is made to change by an abrupt quantity Δ every p time steps. Datasets with different speeds of drift (concept duration) are generated by varying the value of parameter p as we will see in the next chapters.

Gradual-change dataset. Here, changes in μ_y occur continuously at each time step, i.e. $p = 1$, but with smaller (more gradual) magnitude of changes δ. That is, $\delta = \frac{\Delta}{D}$, where D represents the duration between two abrupt concept changes, and Δ is the magnitude of an abrupt change. Datasets with different speeds of drift are generated here by changing the parameter δ.

Note that, the incremental drift generated in our experiments assumes continuous drift at each time step. That is, there is no period of concept stability between the old and new concepts as the concept is continuously changing. This means that there is no transient period (the time steps between the old and new concepts) that needs to

be eliminated. Another scenario of incremental concept changes is when the new concept appears after a transient period. One of the main challenges in such scenario is to eliminate the examples that belong to the transient period once the new concept starts. This requires determining the time when the new concept has started following the transient period. It is worth noting that in our context-aware approach, the examples arriving during the transient period will not be taken into account as they will have different contextual conditions. Such examples might only be considered at the beginning of the change when the concept is new and there is no sufficient number of examples.

3.1.1.2 Benchmark Datasets

In this section, we introduce a number of artificial benchmark datasets that are commonly used in the literature for testing adaptive learning models.

STAGGER Concepts [18]. This dataset exhibits abrupt concept drift, and is defined by three categorical input variables: size, colour, and shape. Each variable has three possible values: $size \in \{\text{small}, \text{medium}, \text{large}\}$, $\text{colour} \in \{\text{red}, \text{green}, \text{blue}\}$, and $\text{shape} \in \{\text{square}, \text{circular}, \text{triangular}\}$. The target concept switches between a sequence of the following three descriptions: Concept 1 ($\text{size} = \text{small} \wedge \text{colour} = \text{red}$); Concept 2 ($\text{colour} = \text{green} \wedge \text{shape} = \text{circular}$); and Concept 3 ($\text{size} = \text{medium} \vee \text{large}$).

Sine1 dataset [88]. This dataset is defined in terms of two uniformly distributed input variables in the range $[0, 1]$, and a binary class label. Positive and negative examples are defined by equation $y = \sin(x)$. Specifically, examples lying below the curve $\sin(x)$ are labeled as positive, and as negative otherwise. The drift is generated by reversing the labeling of the examples during each concept.

Gauss [88]. This dataset is defined in terms of two normally distributed variables, where the mean for positive examples is equal to $[0, 0]$ with standard deviation of 1, and the mean for negative examples is equal to $[2, 0]$ with standard deviation of 4. The drift is generated by reversing the labelling of the examples during each concept.

Rotating Hyperplane [20]. This dataset exhibits gradual concept drift, and is defined by equation: $\sum_{k=1}^{m} a_k x_k = a_0$, where a_k and x_k are the weight and value of input variable X_k, respectively, m is the input vector dimensionality, and a_0 is a threshold value distinguishing two classes for a given concept. The value of a_0 is determined such that $a_0 = \frac{1}{2}\sum_{k=1}^{m} a_k$. That is, the hyperplane divides the space of the examples into two parts, where examples satisfying $\sum_{k=1}^{m} a_k x_k \leq a_0$ could be labelled as positive, and as negative otherwise. The drift is generated by continuously changing the weights a_k of the input variables during each concept. Specifically, at each time step, a_k is updated by quantity $\frac{\Delta_a}{D}$, where Δ_a is the magnitude of change for a_k during each concept, and D is the duration of the concept. In our experiments, we generate 3 uniformly distributed input variables over the range $x_k \in [0, 1]$, $k = 1 : m$. The weights, a_k, are randomly initialised between 0 and 1, i.e. $a_k \in [0, 1]$, and then made to change during each concept by $\Delta_a = 1.5$.

Circles [88]. Similarly to *Sine1*, this dataset is defined in terms of two independent and uniformly distributed input variables, $x_k \in [0, 1]$, $k = 1 : 2$, and a binary class label. Positive and negative examples are determined according to their location with respect to a particular circle. Specifically, examples lying outside the circle are labeled as positive, and as negative otherwise. Four different concepts are defined corresponding to the following four circles:

Center	$[0.2, 0.5]$	$[0.4, 0.5]$	$[0.6, 0.5]$	$[0.8, 0.5]$
Radius	0.15	0.2	0.25	0.3

3.1.2 Real-World Datasets

Electricity Market Dataset. Electricity Market dataset (or Elec) [8] consists of 45312 records concerning electricity price changes, obtained from the Australian New South Wales Electricity Market, between May 1996 and December 1998. Each record represents a period of 30 minutes, with a binary class label and four categorical inputs variables. The class label refers to the price change (up/down) with respect to a moving average of the last 24 hours. This dataset exhibits drifts due to changes in consumption habits corresponding to various candidate contextual characteristics such as season and time of the week [28, 8]. Furthermore, due to the dependencies among the labels in this dataset [23] (which cause even the simplest classifier predicting the next label based only on the previously seen label to achieve very high accuracy), we subsample the data to break such dependencies [100]. Specifically, we take every 20th observation at regular intervals, resulting in a total of 2265 records utilised for classifier evaluation.

Emailing List Dataset. The emailing list dataset (or Elist) [11] contains 1500 examples on how to filter email messages into either interesting or junk according to user preferences. Each record consists of 913 Boolean input variables representing words frequently appearing in the body of the message, with a binary class label. Two recurring concepts are exhibited: Concept 1 (the user is only interested in medical messages), and Concept 2 (the user changes their interest to space and baseball). Such interest change is associated with the context variable Location. Changes in concept are made to occur every 300 examples.

3.2 Contextual Data

3.2.1 Context for Artificial Datasets

To facilitate a comprehensive set of experiments, a general framework for generating contextual data is provided. It allows varying the dimensionality of context data and introducing contextual attributes of various importance.

We consider both discrete and continuous context variables. To introduce *relevant* context variables, we follow the intuitive notion for context, and make these variables to have certain values (in the case of discrete variables) or certain distributions (in the case of continuos variables) whenever a particular concept is in effect, such that a change in these values will indicate a change in the underlying concept. More details on the framework are provided below.

3.2.1.1 Perfect Context Variables

We generate a perfect context variable as follows. In the case of a continuous context variable, its values are made to follow a specific normal distribution for each concept. Specifically, the mean of the context variable's distribution, μ_c, is made to shift over time, either abruptly by a significant quantity Δ_c, every p time steps in the case of abrupt changes, or gradually by a slight quantity δ_c, each time step in the case of a gradual change. That is,

$$\mu_c^{(t+p)} = \mu_c^t + \Delta_c \quad \text{(for abrupt changes)}$$
$$\mu_c^{(t+1)} = \mu_c^t + \delta_c \quad \text{(for gradual changes)}$$

where $\delta_c = \frac{\Delta_c}{D}$, and D is the duration between two abrupt concept changes.

The case of a discrete context variable is more suitable for abrupt changes. For this case, given n concepts, we introduce a context variable C_j with values $v_1, v_2, ..., v_n$, such that $C_j = v_1$ under Concept 1, $C_j = v_2$ under Concept 2,..., $C_j = v_n$ under Concept n.

3.2.1.2 Context Variables of Varying Importance

A possible way to generate a context variable of a particular importance is to make the importance of the variable (on a scale between 0 and 1) corresponds to the number of concepts the variable is able to distinguish. That is, a contextual importance of 0.75 means that the value of the context variable (or the value of its distribution mean in the case of continuous context) changes only in 75% of concept change points. For example, given 5 concepts, a 75% importance indicates that only in 3 out of 4 concept change points the context variable changes its value (or the value of its distribution mean). Similarly, an importances of 0.5 indicates that only in 2 out of 4 concept change points the context variable changes its value (or the value of its mean), etc.

3.2.1.3 Correlation-based Context Variables

Another possible way to generate context variables of varying importance is to make these variables have varying *correlation degrees* with a perfect context variable, corresponding to the importance of the variables. This is achieved by controlling the covariance matrix Σ according to which the variables are generated. Specifically, we generate a normally distributed perfectly contextual variable as described above, denoted Z. Now, to generate a context variable C_j with a specific correlation degree with variable Z, we first sample variable C_j from a normal distribution (with a zero-mean and unit-variance), define a covariance matrix Σ for Z and C_j with the required correlation degree, and then *transform* the data utilising Cholesky decomposition B of

the required covariance matrix Σ, where $BB^T = \Sigma$ (with B^T denoting the transpose of matrix B). The latter transformation step is achieved by multiplying the Cholesky decomposition B by the data matrix (of variables Z and C_j), resulting in a transformed data matrix with the specified correlation from which the context variable is extracted.

Generally, we divide n candidate context variables into n_1 relevant variables (of varying importance), n_2 redundant variables, and n_3 irrelevant variables ($n_1 + n_2 + n_3 = n$). Each relevant variable is made to be correlated with the perfect variable with correlation degrees ranging from 1 to 0.1 with a step size of 0.1. In generating redundant context variables, each variable is set to be perfectly correlated with another relevant context variable. Finally, the irrelevant context variables are generated to have zero correlation with the perfect variable.

3.2.2 Context for Real-World Datasets

The contextual properties for the considered real-world datasets are defined as follows. In the electricity dataset, we regard the following variables as candidate context variables: day of the week (previously used by other researchers as a contextual variable [8]; [28]) and season (this variable is derived with the help of dates provided for each data instance). The value domain for each of these variables is given in Table 3.3. In the email dataset, interest change is associated with the context attribute Location [11], which takes two possible values: Location$\in \{work, home\}$.

Table 3.3: Candidate context variables for electricity dataset.

Variable	**Value Domain**
Day of the Week	$\in \{Mon, Tue, Wed, Thu, Fri, Sat, Sun\}$
Season	$\in \{Winter, Spring, Summer, Autumn\}$

3.3 Evaluation Metrics

In the evaluation of the proposed adaptive learning models, we assume an incremental (stream) learning scenario where the classification model is (re-)estimated on the arrival of each new example (consisting of the input value vector, the target class label, and the associated context). In particular, whenever a new example arrives, the classification model makes a prediction, after which the real class label of this example becomes available. The example is then used to update the model.

Assessing the performance of learning models requires determining the records that are used for training the model, and the records that are used for testing it. In the considered stream learning scenario, the temporal order of data is important since the model is continuously updated during operation, and the incoming data may belong to different concepts. Two approaches are normally utilised in the literature for assessing the performance of learning algorithms in such settings [49]: holdout an independent test set and prequential analysis. The former depends on regular application of the classification model of interest to an independent test set, while the latter utilises each data example for testing the model and then for training it. In our analysis, we utilise the prequential approach as with this approach, all the available data is used for training the model, without the need to keep an independent test set. Moreover, applying the holdout estimate in the presence of concept drift may be problematic given the uncertainty regarding the examples that belong to the currently active concept.

In the presence of concept drift, monitoring the predictive accuracy of the adaptive learning model over time is a major goal. This is because it allows monitoring the behaviour of the learning model (the classifier may perform well only on some parts of the data, and deteriorates in performance on other parts) and its ability to adapt to concept changes. In our experiments, an average accuracy (a ratio between the

number of correctly classified examples and the total number of classified examples) is computed on the arrival of each new example with respect to a moving window of the most recent data. Utilising a moving window is one of the most frequently used forgetting strategies in prequential analysis [49].

In addition to the predictive accuracy, we consider the following evaluation metrics that are widely used in the literature.

Kappa Statistic (KS) [50]. This measure compares the performance of the classifier of interest with a random classifier, computed according to the following formula: $\frac{a-a_r}{1-a_r}$, where a is the accuracy rate of the classifier of interest, and a_r is the accuracy rate of a random classifier, which assigns the same number of examples for each class as the classifier of interest. The values of KS range between -1 and 1, where 0 indicates that the performance of the classifier matches that of the random classifier (i.e. the resulting accuracy is random). On the other hand, the higher the value of KS, the better the performance of the classifier.

The Area Under the ROC Curve (AUC). This measure is one of the most widely used measures for comparing the performance of multiple classifiers. Its values range between 0.5 and 1, where a value of 1 indicates a classifier with a perfect discrimination ability [59]. This measure, however, uses different misclassification costs for each classifier, which corresponds to utilising different performance evaluation metric for each classifier [61]. To overcome this, we use another metric, namely the H-measure, that is described below.

H-measure [61]. This is an alternative measure to AUC. However, unlike AUC, the H-measure assigns an equal misclassification cost distributions to all classifiers, which is an important property when comparing models that evolve over time. As with the above measures, higher values of the H-measure indicate better performance.

Moreover, to evaluate the statistical significance of the difference in performance between the proposed adaptive learning models against other approaches under comparison, we utilise the non-parametric McNemar's test that is normally utilised in the literature for stream learning settings [5]. The advantage of this test is that with algorithms that can be executed only once (the case of stream learning), McNemar's test achieves the smallest Type I error (the probability of incorrectly detecting a difference when no difference exists), with a value less than 5% [51].

3.4 Classification Models

The main classification model adopted in our experiments is the Naive Bayes classifier. This is due to its computational efficiency, ability to handle discrete and continuous variables, robustness to noise, and ability to produce accurate results compared to more complex models [86]. However, since the contributions presented in this thesis are not specific to any particular classification model, experimental results with other two widely used classification models, namely logistic regression and perceptron, are also presented.

More details on the above classification models are presented next.

3.4.1 Naive Bayes Classifier

Bayes classifier is a probabilistic classifier that uses Bayes theorem (see Equation 3.1) to predict the most probable classification outcome.

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} \tag{3.1}$$

Let's assume that $x = (x_1, x_2, ..., x_m)$ represents a value vector of input variables, and $y \in \{0, 1\}$ is the target class label. The decision of assigning label y to vector x is performed as follows.

Compute the probability of each class given vector x:

$$P(Y = 1|x) = \frac{P(Y = 1)P(x|Y = 1)}{P(Y = 1)P(x|Y = 1) + P(Y = 0)P(x|Y = 0)}$$

$$P(Y = 0|x) = \frac{P(Y = 0)P(x|Y = 0)}{P(Y = 1)P(x|Y = 1) + P(Y = 0)P(x|Y = 0)}$$

Assign x to the class with the largest probability. That is, $\hat{y} = 1$ if $P(Y = 1|x) > P(Y = 0|x)$, and $\hat{y} = 0$ otherwise.

Given a limited sample size, it is difficult to achieve meaningful estimates for $P(x|Y = y)$. Therefore, these estimates are normally simplified by assuming class conditional independence among input variables. Thus, the term $P(x|Y = y)$ is computed as follows:

$$P(x|Y = y) = \prod_{j=1}^{m} P(x_j|Y = y) \tag{3.2}$$

Based on the above, Naive Bayes makes its predictions according to the following equation:

$$\hat{y} = argmax_y[P(Y = y) \prod_{j=1}^{m} P(x_j|Y = y)] \tag{3.3}$$

Note that $P(X)$ is omitted form Equation 3.3 since it is constant among the classes, and hence its computation is not required for class prediction. Given the set $\{u(i)\}_{i=1}^{t-1}$ of the observed labelled examples up to time t, where $u(i) = (x(i), y(i))$ is the example observed at time step i, all probabilities are estimated based on this set using the relative frequencies.

In the case of continuous input data, it is usually assumed that input variables in each class are normally distributed [87], and $P(x|Y = y)$ in Equation 3.3 is computed as:

$$P(x_j|Y = y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x_j - \mu)^2}{2\sigma^2}) \tag{3.4}$$

where μ and σ are the mean and standard deviation, respectively, of the input variable x_j in the training data under $Y = y$.

Despite the simplicity of the NB assumption with regard to the independence among input variables, it is still able to achieve good results even with data where this assumption does not hold. This is because although the independence assumption may result in inaccurate posterior probability estimates, the classification remain correct as the maximum probability is often assigned to the correct class [86].

3.4.2 Logistic Regression

In logistic regression classifier, given value vector of input variables $x = (x_1, x_2, ..., x_m)$, and the target class label Y, the prediction is generated according to the following equation:

$$P(Y = 1|x) = g(\beta^T z) \tag{3.5}$$

where, $g(.)$ is the logistic function such that Equation 3.5 is equivalent to:

$$P(Y = 1|x) = \frac{e^{\beta^T z}}{1 + e^{\beta^T z}} \tag{3.6}$$

where $z = (x_0, x_1, x_2, ..., x_m)$ denotes the vector of input variables, and $\beta = (\beta_0, \beta_1, ..., \beta_m)$ represents the vector of regression parameters that need to be estimated. Note that, β_0 represents a bias term that is independent of input data, with $x_0 = 1$.

Given the set $\{u(i)\}_{i=1}^{t-1}$ of the observed labelled examples up to time t, parameters β can be estimated by maximising the following log-likelihood function.

$$LogLik = \sum_{i=1}^{t-1} \varphi(\beta|u(i)) \tag{3.7}$$

where, $\varphi(\beta|u(i))$ is the log-likelihood of the ith observation, and is given as follows:

$$\varphi(\beta|u(i)) = y(i)log(\frac{e^{\beta^T z(i)}}{1+e^{\beta^T z(i)}}) + (1-y(i))log(1-\frac{e^{\beta^T z(i)}}{1+e^{\beta^T z(i)}}) \tag{3.8}$$

The logistic regression classifier can be applicable for both discrete and continuous variables, can handle cases with high dimensionality, and support learning from weighted training data as we will see in Chapter 5.

3.4.3 Perceptron

In this classification method, the predicted class label, $\hat{y}$, is generated based on computing a weighted sum of the input variables, and then checking the sign of the outcome. That is:

$$\hat{y} = sign(x_0 w_0 + x_1 w_1 + x_2 w_2 + ... + x_m w_m) \tag{3.9}$$

where $x = (x_1, x_2, ..., x_m)$ is the set of input variables, $w = (w_1, w_2, ..., w_m)$ represents the parameters (also referred to as weights) associated to each input variable, and w_0 is the bias term with $x_0 = 1$. The *sign* function outputs either a value of 1 or 0. That is, $\hat{y} = 1$ if the weighted sum is positive, and $\hat{y} = 0$ if the weighted sum is negative.

At each time step, the parameters w are updated according to the following rule, with λ representing the perceptron learning rate:

$$w_t = w_{t-1} + \lambda(y - \hat{y})x \tag{3.10}$$

That is, perceptron learns in an online mode (i.e. it does not have to iterate over the whole data at each time step). Moreover, the parameters of input variables are corrected only if the current classification is incorrect. These properties allows efficient utilisation of perceptron for incremental learning settings. On the other hand, perceptron will only work properly if the data is linearly separable.

Chapter 4

Context Identification Model

4.1 Introduction

The first step in context-aware adaptation involves answering the following question: *How to identify context*? In other words, how to recognise the variables that represent contextual information for the underlying concept.

In general, context variables can be derived from the background knowledge about the domain. For example, when dealing with users purchasing habits, time and location can be easily identified as contextual variables, since both usually have an effect on the user preferences: a user has different preferences when at work or at home, during weekdays or weekends. However, in many cases, domain knowledge alone might not be sufficient to characterise the context for the situation at hand, especially in the presence of a large number of potential contextual variables, but which might not all affect the concept of interest. Hence, in such domains with many candidates for context, or where it is difficult to distinguish contextual variables from non-contextual ones, we propose to utilise historical data to deduce contextual variables that actually affect the concept

of interest from a set of candidate variables. Such explicit identification of context, as we will illustrate in Chapters 5 and 6, enables capturing the causes of drift, and hence facilitating more effective adaptation.

The remaining of this chapter is organised as follows. Section 4.2 introduces the problem of context identification. Section 4.3 presents the proposed context learning solution. Experimental setup and results are presented in Section 4.4. Finally, Section 4.5 concludes this chapter.

4.2 Problem Formulation

The context knowledge for an application domain of interest can be represented as a tuple (C, rl), where:

- $C = \{C_1, C_2, ..., C_p\}$ is the initial set of candidate context variables that could possibly affect the concept of interest, with $c_j(t)$ being the value of candidate context variable C_j at time step t. For example, in the domain of electricity price prediction, a potential value for candidate context variable *Day of the Week* at time step t could be the value $Monday$.
- Function $rl(C_j) \in [0, 1]$ returns the relevance degree of candidate context variable C_j. This relevance corresponds to the importance of the context variable in distinguishing concept changes.

Since the set of candidate context variables C can be usually derived from the background knowledge about the domain, and assuming the ability to access their values, our problem of context identification reduces to that of assessing relevance rl. We propose to *learn* such relevance from the historical data available on the concept of interest

based on information theoretic measures, namely Conditional Mutual Information and Conditional Entropy (as detailed next). Whilst these are not novel measures in themselves, their utilisation for the purpose of identifying relevant context variables is a novel application.

4.3 Context Learning

The identification of *relevant* context variables requires testing the existence of the dependency between a candidate context variable and the ability to predict the target variable. Before proceeding with context variables identification, we need to distinguish between three types of variables [31]: input variables, context variables, and irrelevant variables. Input variables are relevant for predicting the target variable of interest, either when are taken individually or in combination with other variables. Context variables, on the other hand, are relevant for predicting the target variable only when are taken in combination with other variables. Finally, irrelevant variables are not predictive neither when are taken individually nor in combination with other variables.

Our approach for the actual context recognition is inspired by the definition of context variables proposed by Turney [31]. According to this definition (as introduced in Chapter 2), the dependency between the candidate context variable and the target variable cannot be tested independently, but only in the presence of the input variables. Specifically, an actual contextual variable C_j affects the discrimination ability of the input variables, i.e. $P(Y|X, C_j) \neq P(Y|X)$. We extend this definition to allow measuring the influence degree that C_j has on X (and consequently facilitating the calculation of relevance rl), as follows: the degree of influence of a candidate context variable $C_j \in C$ on the discrimination ability of the input variables X, is proportional to the difference between the conditional probabilities $P(Y|X, C_j)$ and $P(Y|X)$.

Given the above definition, measuring the dependency between C_j and Y can be interpreted as measuring the additional information that C_j has about Y, and that is not already provided by X. In other words, we are interested in measuring the degree to which knowledge of the candidate context variable reduces the uncertainty in the target variable, given the input variables. Shannon's Information Theory [30], which measures *uncertainty* and *information content* (the quantities of interest), thus provides a good formalisation for this purpose. It is introduced next, followed by the respective context identification approach, and a corresponding computational realisation.

4.3.1 Information Theoretic Measures

The measures of Information Theory of interest for our model, as we will see in Section 4.3.2, are *Conditional Mutual Information* and *Conditional Entropy*. In the following, we provide formal definitions for these measures, along with a number of related key information theoretic measures utilised in our analysis, and summarise some of their properties that form the basis of our work presented next.

4.3.1.1 Entropy

The concept of *entropy* relevant for our model building is the entropy of a discrete random variable. It is defined as a measure of the degree of uncertainty in a given random variable [24]. The uncertainty quantified by entropy depends on the probabilities of the possible values the variable of interest can take. Specifically, high values of entropy indicate that each state (outcome) of a variable has similar probability of occurrence, i.e. the variable is highly random. For example, when tossing a fair coin, the entropy is equal to its maximum value [24]. On the other hand, low values of entropy mean the states of a variable have different probabilities of occurrence, i.e. the variable is less random.

Given a discrete random variable Y, with a probability mass function $p(y)$, the entropy associated with this variable, $H(Y)$, is given by [24].

$$H(Y) = -\sum_{y} p(y) \log p(y) \tag{4.1}$$

with the convention that $0 \log 0 = 0$. Here, the base of the logarithm determines the units in which the entropy is measured. Specifically, when the base of the logarithm is 2, entropy is measured in bits, and when the base of the logarithm is e, entropy is measured in nats. In all experiments, we will use the natural logarithm in entropy calculations.

In the case of two random variables, Y and X, the entropy between them is measured by the *joint* entropy, $H(Y, X)$, which is given as follows [24]

$$H(Y, X) = -\sum_{y} \sum_{x} p(y, x) \log p(y, x) \tag{4.2}$$

where $p(x, y)$ is the joint probability distribution.

Joint entropy can also be expressed in terms of the entropy of one of the variables and the *conditional* entropy of the other variable as given below (this is usually referred to as chain rule property):

$$H(Y, X) = H(X) + H(Y|X) \tag{4.3}$$

$$H(Y, X) = H(Y) + H(X|Y) \tag{4.4}$$

where $H(.|.)$ refer to the conditional entropy, which allows quantifying the amount of uncertainty regarding one random variable given knowledge of another random variable(s), computed as follows:

$$H(Y|X) = -\sum_{y} \sum_{x} p(y, x) \log p(y|x) \tag{4.5}$$

According to Equations 4.3 and 4.4, conditional entropy can also be computed in terms of entropy and joint entropy.

Note that, the value of entropy for discrete variables is always non-negative ($H(Y) \geq 0$) as opposed to the entropy of continuous variables that can take negative values, but this is out of the scope of this thesis.

Based on the above, the goal of any prediction task can be formalised as minimising the entropy (uncertainly) of the variable we are trying to predict (i.e. minimising the entropy of the class label $H(Y)$). Hence, any candidate variable minimising this uncertainty is regarded as a predictive variable. Similarly, in the presence of concept changes, this goal can be expressed as minimising the conditional entropy of the target variable given knowledge of input variable(s). Hence, this measure is utilised in our analysis to assess the relevance of each candidate context variable as we will see in Section 4.3.2.

4.3.1.2 Mutual Information

Mutual information has been widely used in pattern recognition and data mining applications. The main advantage of mutual information is the ability to identify the existence of the dependencies between variables regardless of the functional relationships between these variables [45]. A diminishing value of the mutual information does indicate that the variables under consideration are independent, and therefore it is considered as a general measure of the statistical dependency between variables. This is in contrast to the commonly used methods such as Pearson correlation, which only captures linear relations among variables, and hence cannot be used to indicate independence between variables.

Formally, mutual information is defined as a measure of the amount of information that one random variable provides about another random variable [24]. In other words, it quantifies the reduction in uncertainty about one random variable with respect to another random variable. To accommodate our definition of context variables, however, the mutual information concept should be extended to *conditional* mutual information. Similarly to mutual information, conditional mutual information facilitates measuring the reduction in the uncertainty in one random variable given the knowledge of another random variable, but in the presence of a third random variable [24]. With reference to our problem, it facilitates measuring the reduction in uncertainty in the target variable with respect to the context variable, while taking into consideration the knowledge already provided by the input variables. Formal definitions of both measures are provided below.

Mutual information between two random variables X and Y with a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$ is computed as follows [24].

$$I(Y, X) = \sum_{x} \sum_{y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \tag{4.6}$$

with the conventions that $0 \log \frac{0}{0} = 0$, $p \log \frac{p}{0} = \infty$, and $0 \log \frac{0}{q} = 0$. For continuous random variables, the summation in Equation 4.6 is replaced by the integral, and the mutual information becomes

$$I(Y, X) = \int \int p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dxdy \tag{4.7}$$

Mutual information can also be defined in terms of the reduction in the uncertainty in one random variable given the knowledge of another random variable, i.e. the difference between the entropy of one random variable and its conditional entropy (remaining

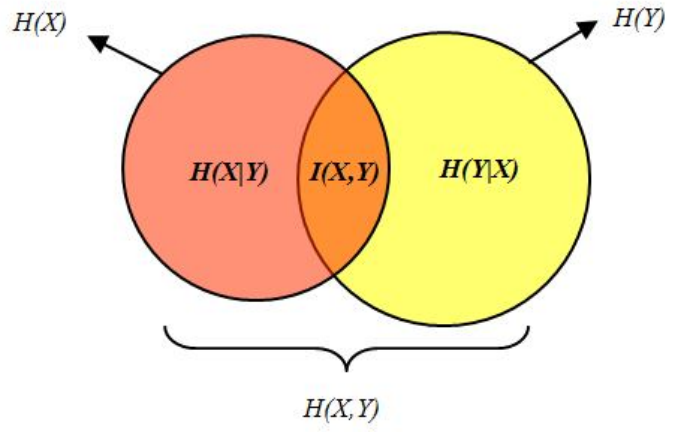


FIGURE 4.1: Venn diagram illustrating the relationship between entropy and mutual information [24].

uncertainty) in the presence of another random variable. Specifically, given two random variables X, Y, the mutual information between these variables can be expressed in terms of their entropies as follows [24].

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{4.8}$$

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{4.9}$$

The relation between mutual information and entropy given in Equations 4.8 and 4.9 is illustrated in Figure 4.1 using Venn diagram.

Conditional mutual information between two random variables C_j and Y given a third random variable X, $I(C_j, Y|X)$, is computed as follows [24].

$$I(C_j, Y|X) = \sum_{x} \sum_{c_j} \sum_{y} p(c_j, y, x) \log \frac{p(c_j, y|x)}{p(c_j|x)p(y|x)} \tag{4.10}$$

In the case of continuous random variables, the conditional mutual information is defined by:

$$I(C_j, Y|X) = \int\int\int p(c_j, y, x) \log \frac{p(c_j, y|x)}{p(c_j|x)p(y|x)} dxdcdy \tag{4.11}$$

Similarly to mutual information, conditional mutual information can be defined in terms of entropies as follows [24].

$$I(C_j, Y|X) = H(C_j|X) - H(C_j|Y, X) \tag{4.12}$$

$$I(C_j, Y|X) = H(C_j, X) + H(Y, X) - H(X) - H(C_j, Y, X) \tag{4.13}$$

Conditional mutual information (or shortly CMI), satisfies three properties [24]: *non-negativity*, i.e. $I(C_j, Y|X) \geq 0$; *measure of independence*, where CMI equals zero *if and only if* the two variables are independent, i.e. $I(C_j, Y|X) = 0 \iff P(Y|C_j, X) = P(Y|X)$; and *boundedness*, where the maximum value of CMI is bounded by the minimum of the conditional entropies of the variables of interest, i.e. $0 \leq I(C_j, Y|X) \leq \min\{H(C_j|X), H(Y|X)\}$. In other words, the maximum value of the conditional mutual information is achieved when one variable can perfectly predict the value of the other variable.

Having defined the measures of information theory of interest, we next present how these measures are utilised in the proposed context identification approach.

4.3.2 Context Relevance Function

In the proposed context identification model, relevance function rl focuses on associating each contextual variable with a weighting factor reflecting its importance (its

contextual knowledge). For example, when $rl(C_j) = 0$ it indicates that candidate context variable C_j has no effect regarding concept drift (i.e. it is not a contextual variable for the concept of interest).

Utilising the information theoretic measures introduced in the previous section (Section 4.3.1), a candidate context variable $C_j \in C$ is considered to be actually contextual if there is a dependency between that variable and the target variable Y, i.e. $I(C_j, Y|X) \neq 0$, indicating that C_j reduces uncertainty about Y. The relevance of variable C_j is calculated according to its corresponding degree of uncertainty reduction, as follows:

$$rl(C_j) = \begin{cases} \frac{I(C_j,Y|X)}{H(Y|X)} & \textbf{if } I(C_j, Y|X) \neq 0 \\ 0 & \textbf{Otherwise} \end{cases} \tag{4.14}$$

Based on the above, assessing the relevance among candidate context variables involves considering three different variables: context variable, target variable, and input variables. Both context and input variables can belong to different spaces (e.g. discrete or continuous), which poses a challenge on the estimation of conditional mutual information as well as the respective conditional entropy. Hence, in the following, we discuss how to estimate these measures, considering discrete and continuous settings.

4.3.3 Estimation of Conditional Mutual Information and Conditional Entropy

In discrete systems, the probability mass functions for estimating (conditional) mutual information can be computed using the relative frequencies from the observed examples (see Equations 4.6 and 4.10). However, computing (conditional) mutual information for continuous variables requires estimating probability density functions (pdf) from the available data samples (see Equations 4.7 and 4.11), which is not a straightforward task, and in fact, considered as one of the main problems associated with applying mutual

information measure. With this regard, different possible non-parametric approaches exist in the literature for estimating mutual information, the most common of which are histogram-based approaches (e.g. [36]), kernel-based density estimations (e.g. [41]), and a recent entropy based k-nearest neighbour estimator (e.g. [44]; [46]). Given that the estimator of the mutual information for context variables must be able to work well with higher dimensions, histogram and kernel density estimators are not appropriate since both suffer from the curse of dimensionality, i.e. the number of examples required for estimating the pdf increases exponentially with the number of variables [44]. Unlike many other approaches, the k-nearest neighbour estimator proposed by Kraskov et al. [29] allows accommodating high variable dimensionality, and produces more accurate results when compared to other approaches ([39]; [42]).

Based on the above, for the purpose of our analysis, we utilise Kraskov's nearest neighbour approach in estimating conditional mutual information. More details on this approach are presented below.

4.3.3.1 k Nearest Neighbour Estimator

The utilisation of k-nearest neighbour statistics has been widely used for entropy estimation, but the extension of this estimator to mutual information was only proposed by Kraskov et al. [29]. In the following, we present Kraskov's method for estimating mutual information between two variables, and show its corresponding derivation for the conditional case ([37]; [48]).

The estimator of the mutual information proposed by Kraskov is based on the expression of mutual information in terms of entropy given in Equation 4.8. The entropies in this equation are estimated using k-nearest neighbour distances as detailed below.

First, given two random variables C, Y of dimensions d_c, d_y, respectively, the joint entropy $H(C, Y)$ is calculated using Kozachenko and Leonenko [40] based estimator given by:

$$\hat{H}(C, Y) = -\psi(k) + \psi(N) + \log c_{d_c} c_{d_y} + \frac{d_c + d_y}{N} \sum_{i=1}^{N} \log \epsilon(i) \tag{4.15}$$

where:

$\psi(k)$ is the digamma function, which is given for an integer k by:
$\psi(k) = \frac{d}{d_k} \ln(k-1)!$; N is the number of examples; c_d is the volume of d-dimensional unit ball (which is equal to 1 in this analysis); $\epsilon(i)$ is twice the distance from ith sample point in the joint space (C, Y) to its kth nearest neighbour.

The distance measure utilised in the above equation is maximum norm, i.e. given two vector variables $Z = (C, Y)$, $Z' = (C', Y')$, the distance between them is computed as [29]:

$$||z - z'|| = max\{||c - c'||, ||y - y'||\} \tag{4.16}$$

where to compute the norms $||c - c'||$ and $||y - y'||$ any distance metric can be utilised.

In the second step, the projection from the joint space of (C, Y) is made to the spaces of variables C and Y, so that the entropies $H(C)$, $H(Y)$ are computed as follows:

$$\hat{H}(C) = -\frac{1}{N} \sum_{i=1}^{N} \psi[n_c(i) + 1] + \psi(N) + \log c_{d_c} + \frac{d_c}{N} \sum_{i=1}^{N} \log \epsilon(i) \tag{4.17}$$

$$\hat{H}(Y) = -\frac{1}{N} \sum_{i=1}^{N} \psi[n_y(i) + 1] + \psi(N) + \log c_{d_y} + \frac{d_y}{N} \sum_{i=1}^{N} \log \epsilon(i) \tag{4.18}$$

where $n_c(i)$, $n_y(i)$ denote the number of data points whose distances from the ith point of C, and Y, respectively, are less than $\frac{\epsilon(i)}{2}$. By substituting Equations 4.15, 4.17

and 4.18 in Equation 4.8, we get the estimate of mutual information, $\hat{I}(C, Y)$, given in Equation 4.19 [29]:

$$\hat{I}(C, Y) = \psi(k) + \psi(N) - \langle \psi[n_c(i) + 1] + \psi[n_y(i) + 1] \rangle \tag{4.19}$$

where C, Y can be of any dimension, and $\langle ... \rangle$ refers to the average over all points i.

Now, in a similar way, the estimate of conditional mutual information is given in Equation 4.20 [48].

$$\hat{I}(C, Y|X) = \psi(k) - \langle \psi[n_{cx}(i)] + \psi[n_{yx}(i)] - \psi[n_x(i)] \rangle \tag{4.20}$$

where, $n_{cx}(i)$, $n_{yx}(i)$, denote the number of data points whose distances from the ith point of the joint spaces (C, X) and (Y, X), respectively, are less than the distances to the kth nearest neighbour from the corresponding ith point in the joint space (C, Y, X).

Kraskov's estimate assumes that all variables are continuous, including the target variable. However, in classification problems, with discrete target variables (the class labels of the examples), the general efficiency of the estimator will decrease [38]. This is because the accuracy of the nearest neighbour search, upon which the estimation is based, in the (joint) space involving a discrete variable will be influenced by having data points that share the same values for this variable. Specifically, when, for example, Y is discrete, and has a larger distance to its kth neighbour than that of C and X (we use maximum norm as a distance metric), the data points that share the same value of Y with the kth nearest neighbour will not be considered in calculating $n_{yx}(i)$, which can reduce the accuracy of the estimator [46]. To overcome this problem, an updated version of Kraskov's estimator to estimate mutual information for classification problems is proposed by Gomez-Verdejo et al. [38], where the probability mass function of the

class label is calculated separately using relative frequencies from the available data samples as detailed in the following section.

4.3.3.2 Classification-Oriented Estimation

In this section, we summarise the extension of Kraskov's mutual information estimator for classification problems as proposed by Gomez-Verdejo et al., followed by its derivation for conditional mutual information proposed for our analysis.

In Gomez's approach, the proposed estimator of mutual information is derived from Equation 4.9. In particular, let us assume that C, Y, are two random variables, where Y represents the categorical class label. Given that Y is discrete, Equation 4.9 becomes:

$$I(C,Y) = H(C) - \sum_{y} p(Y=y)H(C|Y=y) \tag{4.21}$$

Similarly to Kraskov's estimator, the entropies $H(C)$, $H(C|Y)$ in Equation 4.21 are estimated using the following formula:

$$\hat{H}(C) = -\psi(k) + \psi(N) + \log c_{d_c} + \frac{d}{N}\sum_{i\in N}\log\epsilon(i) \tag{4.22}$$

where $\epsilon(i)$ is twice the distance from ith sample point in the space of C to its kth nearest neighbour.

By substituting Equation 4.22 in 4.21 we get the following estimator [38]:

$$\hat{I}(C,Y) = \psi(N) - \frac{1}{N}\sum_{y} N_y\psi(N_y) + \frac{d}{N}[\sum_{i\in N}\log\epsilon(i) - \sum_{y}\sum_{i\in N}\epsilon(i_y)] \tag{4.23}$$

where N_y denotes the number of data samples belonging to class y, and $\epsilon(i_y)$ is twice the distance from ith sample point in the space of C to its kth nearest neighbour obtained

from the data samples belonging to class y. This is in contrast to $\epsilon(i)$ which is obtained from all available data samples.

Based on the above, we now present our estimator for conditional mutual information similar to that given in Equation 4.23 as detailed below.

In order to get an estimator for conditional mutual information, we use the following chain rule [24]:

$$I(C, Y|X) = I(Y, \{C, X\}) - I(Y, X) \tag{4.24}$$

By substituting Equation 4.9 in 4.24 we get:

$$I(C, Y|X) = H(C, X) - H(C, X|Y) - H(X) + H(X|Y) \tag{4.25}$$

Now, to compute $I(C, Y|X)$ according to Equation 4.25, we first compute the joint entropy $H(C, X)$ using formula similar to that of Equation 4.15, i.e.

$$\hat{H}(C, X) = -\psi(k) + \psi(N) + \log c_{d_c} c_{d_x} + \frac{d_c + d_x}{N} \sum_{i \in N} \log \epsilon(i) \tag{4.26}$$

where $\epsilon(i)$ is twice the distance from ith sample point in the joint space of (C, X) to its kth nearest neighbour considering all examples available. The conditional entropy $H(C, X|Y)$ is computed similarly but here only the data samples belonging to a certain class are considered when searching for neighbours as illustrated in Equation 4.27.

$$\hat{H}(C, X|Y) = \sum_{y} p(y)[-\psi(k) + \psi(N_y) + \log c_{d_c} c_{d_x} + \frac{d_c + d_x}{N_y} \sum_{i_y \in N_y} \log \epsilon(i_y) \tag{4.27}$$

Here, $\epsilon(i_y)$ is twice the distance from ith sample point in the joint space of (C, X) to its kth nearest neighbour obtained from the data samples belonging to class y.

Now, the entropy $H(X)$ is not estimated independently (i.e. according to Equation 4.22) but by projecting from the space of (C, X) into the space of X using the following formula [29]:

$$\hat{H}(X) = -\frac{1}{N}\sum_{i\in N}\psi[n_x(i)+1] + \psi(N) + \log c_{d_x} + \frac{d_x}{N}\sum_{i\in N}\log\epsilon(i) \tag{4.28}$$

where $n_x(i)$ denotes the number of data points whose distances from the ith point of X are less than $\frac{\epsilon(i)}{2}$, where $\epsilon(i)$ is obtained according to Equation 4.26 using all available data samples.

In a similar way, $H(X|Y)$ is computed by

$$\hat{H}(X|Y) = \sum_{y} p(y)[-\frac{1}{N_y}\sum_{i\in N_y}\psi[n_x(i_y)+1]+\psi(N_y)+\log c_{d_x}+\frac{d_x}{N_y}\sum_{i_y\in N_y}\log\epsilon(i_y)] \tag{4.29}$$

where $n_x(i_y)$ denotes the number of data points in the examples belonging to class y, whose distances from the ith point of X are less than $\frac{\epsilon(i_y)}{2}$, where $\epsilon(i_y)$ is obtained according to Equation 4.27.

Now, having estimated conditional mutual information for continuous context variables, we now show how these estimates can be extended to the case with discrete context variables. First, in the case where X is continuous, $I(C_j, Y|X)$ can be computed as follows:

$$I(C_j, Y|X) = I(C_j, \{Y, X\}) - I(C_j, X) \tag{4.30}$$

which is equivalent to:

$$I(C_j, Y|X) = H(C_j, X) + H(Y, X) - H(X) - H(C_j, Y, X) \tag{4.31}$$

By substituting Equation 4.4 in Equation 4.31, we get the following expression:

$$I(C_j, Y|X) = H(C_j) + H(X|C_j) + H(Y) + H(X|Y) - H(X) - H(C_j, Y) - H(X|C_j, Y) \tag{4.32}$$

This allows estimating $I(C_j, Y|X)$ following the same steps explained for Equation 4.25.

In the case where X is discrete, $I(C_j, Y|X)$ can be estimated according to Equation 4.10. It is worth mentioning, however, that although the case of estimating conditional mutual information with discrete X does not pose a challenge unlike the case with continuous X (as indicated previously), such estimations require a relatively large sample size, depending on the dimensionality of X and the number of possible states for each variable in X. Otherwise, the joint conditional probabilities, and consequently conditional mutual information (see Equation 4.10), will not be estimated correctly.

Before proceeding with the evaluation of the proposed criterion for the identification of relevant context variables, we first show how it can be utilised to address any possible redundancy among context variables. More details are outlined next.

4.3.4 Addressing Correlation Among Context Variables

Mutual information has been widely used as a powerful selection criterion for assessing both relevance and redundancy between variables in the input variable selection literature (e.g. [34]; [43]; [35]). In the following, we provide a brief overview on its utilisation in variable selection, illustrate the required modifications for its implementation in selecting context variables, and present the corresponding solution.

Formally, the problem of variable selection under the framework of mutual information can be formalised as follows [34]: given an initial set of variables, $C = \{C_1, C_2, ..., C_p\}$, we need to select the subset $S \subseteq C$ that jointly has the largest mutual information

with the target variable, Y. Specifically, we need to find the subset S that maximises the following term (also referred to as the max-dependency criterion) [43]:

$$argmax_{S \subseteq C} I(S, Y) \tag{4.33}$$

For the purpose of our analysis, we modify the above formulation for context variable selection as follows: given an initial set of variables, we need to select a subset of variables that will best characterise the current context (contain maximum information what concerns the ability to discriminate between concepts). That is, we need to select the subset $S \subseteq C$ that jointly has the largest mutual information with the target variable, Y, in the context of the input variables, X. Specifically, we need to find the subset S that maximises the following term:

$$argmax_{S \subseteq C} I(S, Y|X) \tag{4.34}$$

Since the number of possible combinations for achieving the optimal subset is exponential, the time to solve the variable selection problem is also exponential. For example, given $p = 10$, the number of possible combinations is $2^p = 1024$. Consequently, reducing the search space is essential to reduce the computational cost of selection. To achieve this, sequential-based search strategies are usually utilised to select the candidate sets, where one feature is selected (or removed) at each time step.The most common search strategies for variable selection are sequential forward selection and sequential backward elimination [47]. The forward search strategy starts with an empty set, and at each time step, the variable that maximizes the joint mutual information is added to the set. On the other hand, the backward search strategy starts with the set S that contains all variables, and then at each time step, the variable that maximizes the joint mutual information by its removal is excluded from the set. Because backward

search strategy is more computationally expensive, most search strategies are based on forward selection [47]. For a similar reason, we base the proposed selection algorithm on forward selection, and detail its stages below along with the challenges involved.

It is worth noting that forward selection suffers from two limitations [47]. In particular, unlike backward selection strategy, it limits measuring the relevance of each candidate variable only to those variables that have already been selected. In addition, once the variable is selected it cannot be removed (this limitation applies to the backward selection as well). Some attempts exist to overcome these two limitations [122], but this is out of the scope of this thesis.

4.3.4.1 Context Selection Algorithm

Given an initial set of variables $C = \{C_1, C_2, ..., C_p\}$, target variable Y, and input variables X, we start with an empty set S, where the first variable $C_j \in C$ to be selected is the one that has the highest value for the conditional mutual information, i.e.

$$argmax_{C_j \in C} I(C_j, Y|X) \tag{4.35}$$

For selecting the following variables, given already selected variables S, the next variable to be selected, $C_j \in C \setminus S$, should maximise:

$$argmax_{C_j \in C \setminus S} I(\{S, C_j\}, Y|X) \tag{4.36}$$

Given that the term $I(\{S, C_j\}, Y|X)$ satisfies the following chain rule (see Equation 4.24):

$$I(\{S, C_j\}, Y|X) = I(Y, S|X) + I(Y, C_j|\{S, X\}) \tag{4.37}$$

and since $I(Y, S|X)$ is constant for a given feature subset S, so maximising Equation 4.36 is equivalent to maximising the term $I(Y, C_j|\{S, X\})$.

The utilisation of $I(Y, C_j|\{S, X\})$ as a selection criterion addresses both relevance and redundancy between variables. This is because $I(Y, C_j|\{S, X\})$ can be expressed as:

$$I(Y, C_j|\{S, X\}) = I(C_j, \{Y, S, X\}) - I(C_j, S) \tag{4.38}$$

Consequently, in order for the variable to be selected, it has to provide the largest information with regard to the target variable, i.e. addressing relevance measured by $I(C_j, \{Y, S, X\})$, that is not already given by variables in S, i.e. addressing redundancy between variables measured by $I(C_j, S)$. The detailed steps of the corresponding proposed selection algorithm are provided in Algorithm 1.

Algorithm 1 CMI-based Context Selection Algorithm

1: $S \leftarrow \emptyset$, $C = \{C_1, C_2, ..., C_p\}$
2: Compute $I(Y, C_j|X)$ for each variable $C_j \in C$
3: Select the variable C^* with the largest $I(Y, C_j|X)$:
   $C^* \leftarrow argmax_{C_j \in C}\{I(Y, C_j|X)\}$
4: Add the selected variable to S: $S \leftarrow \{C^*\}$
5: Exclude the selected variable from C: $C \leftarrow C \setminus C^*$
6: **while** $C \neq \emptyset$ **do**
7: For all $C_j \in C$ compute $I(Y, C_j|\{X, S\})$
8: Select the variable C^* that maximises the following criterion:
   $C^* \leftarrow argmax_{C_j \in C}\{I(Y, C_j|\{X, S\})\}$
9: $S \cup \{C^*\}$
10: $C \leftarrow C \setminus C^*$

Note that, the variables with the highest values for the CMI are included in the analysis. As a termination criterion for the context selection algorithm, we specified the maximum number of variables that can be selected. Alternatively, it is possible to introduce a threshold such that any variable with a CMI less than a certain value is excluded from the set of contextual variables.

Despite the advantages of applying $I(Y, C_j|\{S, X\})$ as a selection criterion, the main challenge associated with it is that it requires estimating conditional mutual information between high-dimensional variables as it involves conditioning on the input variables. To overcome this, it is possible to utilise one of the approximation criteria that have been proposed in the literature on input variable selection as detailed below.

4.3.4.2 Context Selection Algorithm: Heuristic-based Computation

Various approximation methods in the input variable selection literature are proposed for computing the selection criterion in Equation 4.33. The most popular approximation method is based on the heuristic criterion proposed by Battiti [34] in mutual information-based feature selection algorithm (MIFS). Specifically, the term $I(Y, S)$ is approximated by taking the difference between the relevance of the individual candidate variable C_j with the target variable approximated by $I(Y, C_j)$, and its redundancy. The latter is approximated by the weighted sum of the mutual information between the candidate variable and formerly selected variables in S. That is, Equation 4.33 is approximated as:

$$I(Y, C_j|S) \cong I(Y, C_j) - \beta \sum_{C_i \in S} I(C_i, C_j) \tag{4.39}$$

where $\beta \in [0, 1]$ is a redundancy weighting parameter.

Given the difficulty to specify the value of parameter β, the most popular implementation of Battiti's approach is the min-redundancy max-relevance (mRMR) criterion

proposed by Peng et al. [43], where the parameter β in Equation 4.39 is replaced by the average of $I(C_i, C_j)$, i.e.

$$I(Y, C_j|S) \cong I(Y, C_j) - \frac{1}{|S|} \sum_{C_i \in S} I(C_i, C_j) \tag{4.40}$$

where $|S|$ is the cardinality of S.

In the case of context variables, the selection criterion $I(Y, C_j|\{S, X\})$ with mRMR algorithm becomes as follow:

$$I(Y, C_j|\{S, X\}) \cong I(Y, C_j|X) - \frac{1}{|S|} \sum_{C_i \in S} I(C_i, C_j) \tag{4.41}$$

The detailed steps of MIFS and mRMR approaches are presented in Algorithms 2 and 3, respectively, after appropriate adaptation to match our problem of context variable selection. Both algorithms will be tested in our experimental analysis.

Algorithm 2 MIFS for Context Variables Selection

1: $S \leftarrow \emptyset$, $C = \{C_1, C_2, ..., C_p\}$

2: Compute $I(Y, C_j|X)$ for each variable $C_j \in C$

3: Select the variable C^* with the largest $I(Y, C_j|X)$:

$C^* \leftarrow argmax_{C_j \in C}\{I(C_j, Y|X)\}$

4: $S \leftarrow \{C^*\}$

5: $C \leftarrow C \setminus C^*$

6: **while** $|S| < r$ **do**

7: Select the variable C^* that maximises the following criterion:

$C^* \leftarrow argmax_{C_j \in C}\{I(Y, C_j|X) - \beta \sum_{C_i \in S} I(C_i, C_j)\}$

8: $S \cup \{C^*\}$

9: $C \leftarrow C \setminus C^*$

Algorithm 3 mRMR for Context Variables Selection

1: $S \leftarrow \emptyset$, $C = \{C_1, C_2, ..., C_p\}$

2: Compute $I(Y, C_j|X)$ for each variable $C_j \in C$

3: Select the variable C^* with the largest $I(Y, C_j|X)$:

$C^* \leftarrow argmax_{C_j \in C}\{I(C_j, Y|X)\}$

4: $S \leftarrow \{C^*\}$

5: $C \leftarrow C \setminus C^*$

6: **while** $|S| < r$ **do**

7: Select the variable C^* that maximises the following criterion:

$C^* \leftarrow argmax_{C_j \in C}\{I(Y, C_j|X) - \frac{1}{|S|}\sum_{C_i \in S} I(C_i, C_j)\}$

8: $S \cup \{C^*\}$

9: $C \leftarrow C \setminus C^*$

Other two popular approximation algorithms existing in the literature are normalised mutual information based feature selection (NMIFS) [36], and conditional mutual information based feature selection (CMIFS) [35]. Both algorithms are also based on Battiti's approximation criteria. In NMIFS, $I(Y, C_j|S)$ is approximated as:

$$I(Y, C_j|S) \cong I(Y, C_j) - \frac{1}{|S|}\sum_{C_i \in S} \frac{I(C_i, C_j)}{\min\{H(C_i), H(C_j)\}} \tag{4.42}$$

which is modified as follows for the case of context variable

$$I(Y, C_j|\{S, X\}) \cong I(Y, C_j|X) - \frac{1}{|S|}\sum_{C_i \in S} \frac{I(C_i, C_j)}{\min\{H(C_i), H(C_j)\}} \tag{4.43}$$

In CMIFS, the relevance and redundancy are approximated by utilising two previously selected variables through the following expression:

$$I(Y, C_j|S) \cong I(Y, C_j|C_1) - I(C_j, C_l|C_1) + I(C_j, C_l|Y) \tag{4.44}$$

where C_1, C_l are the first and last selected variables, respectively. In the case of context variables, this becomes:

$$I(Y, C_j|\{S, X\}) \cong I(Y, C_j|\{C_1, X\}) - I(C_j, C_l|C_1) + I(C_j, C_l|Y) \tag{4.45}$$

The detailed steps of NMIFS and CMIFS algorithms (after accounting for the case of context variables) are detailed in Algorithms 4 and 5, respectively.

Algorithm 4 NMIFS for Context Variables Selection

1: $S \leftarrow \emptyset$, $C = \{C_1, C_2, ..., C_p\}$
2: Compute $I(Y, C_j|X)$ for each variable $C_j \in C$
3: Select the variable C^* with the largest $I(C_j, Y|X)$:
   $C^* \leftarrow argmax_{C_j \in C}\{I(C_j, Y|X)\}$
4: $S \leftarrow \{C^*\}$
5: $C \leftarrow C \setminus C^*$
6: **while** $|S| < r$ **do**
7: Select the variable C^* that maximises the following criterion:
   $C^* \leftarrow argmax_{C_j \in C}\{I(Y, C_j|X) - \frac{1}{|S|}\sum_{C_i \in S} \frac{I(C_i, C_j)}{\min\{H(C_i), H(C_j)\}}\}$
8: $S \cup \{C^*\}$
9: $C \leftarrow C \setminus C^*$

All presented algorithms order the variables according to their importance, and terminate whenever the number of selected variables reaches the required number of variables to be selected (determined by a user-defined parameter r).

Algorithm 5 CMIFS for Context Variables Selection

$S \leftarrow \emptyset$, $C = \{C_1, C_2, ..., C_p\}$

Compute $I(C_j, Y|X)$ for each feature $C_j \in C$

Select the variable C^* with the largest $I(C_j, Y|X)$:

$C^* \leftarrow argmax_{C_j \in C}\{I(C_j, Y|X)\}$

$S \leftarrow \{C^*\}$

$C \leftarrow C \setminus C^*$

$C_1 \leftarrow C^*$ and $C_l \leftarrow C^*$

Select the variable C^* with the largest $I(C_j, Y|C_1)$:

$C^* \leftarrow argmax_{C_j \in C}\{I(C_j, Y|C_1)\}$

$S \cup \{C^*\}$

$C \leftarrow C \setminus C^*$

$C_l \leftarrow C^*$

while $C \neq \emptyset$ **do**

  for $C_j \in C$ **do**

    If $I(C_j, Y|X) > 0$ and $\frac{I(C_j, Y|\{C_l, X\})}{I(C_j, Y|X)} \leq \theta$, where $\theta \in [0, 1]$

    Remove C_j from the candidate set of context variables: $C \leftarrow C \setminus C_j$

  Select the variable C^* that maximises the following criterion:

  $C^* \leftarrow argmax_{C_j \in C}\{I(Y, C_j|\{C_1, X\}) - I(C_j, C_l|C_1) + I(C_j, C_l|Y)\}$

  $S \cup \{C^*\}$

  $C \leftarrow C \setminus C^*$

  $C_l \leftarrow C^*$

4.4 Evaluation

In this section, we conduct an empirical evaluation of the proposed context identification model, and study the influence of various factors affecting its behaviour. We also

illustrate the advantage of the proposed approach over other approaches existing in the literature. Further details are presented next.

4.4.1 Objectives

The main objectives of our empirical analysis can be summarised as follows.

Objective 1. Test the ability of the proposed approach to correctly identify the actual contextual characteristics of the domain, and to rank these according to their importance.

Objective 2. Test the behaviour of the proposed approach under varying factors, including type of concept change (abrupt or gradual), speed of change, presence of noise and problem dimensionality (for both input and context variables).

Objective 3. Test the behaviour of the proposed approach in a static environment (i.e. in settings with no concept drift).

Objective 4. Compare the performance of the proposed approach against other approaches that could be utilised in recognising the relevance of context variables.

Objective 5. Test the ability of the proposed approach to address redundancy among context variables.

The corresponding experimental setup and results are discussed in what follows. All presented experimental results are averaged over multiple runs.

4.4.2 Experimental Design

Datasets. To allow us to control the actual contextual properties of the domain and to evaluate in different settings (as indicated in the above objectives), we base our

evaluation on artificial datasets generated according to the data generation framework introduced in Chapter 3. As context, we introduce 10 candidate context variables with decreasing order of importance, ranging the correlation levels from 1 to 0.1 with a step size of 0.1 (the description of this context simulation framework is provided in Chapter 3). Other datasets utilised in the analysis include Sine1, Gauss, and Circles. Context for these datasets is simulated as indicated above. Moreover, we utilise the electricity market *Elec* and email *Elist* datasets as test cases with real candidate contextual characteristics: day of the week and season for *Elec* dataset and location for *Elist* dataset (see Chapter 3 for the description of these datasets). In all experiments, the value of the parameter k (for k nearest neighbour estimator utilised in estimating conditional mutual information) is set to 6 (see Appendix A for the evaluation of different k values).

Performance Measures. For the artificial datasets, we compare the weights identified for candidate context variables against their ranking preferences being according to the correlation degrees stated above (the weights for the contextual variables range between 0 and 1, where 1 indicates a perfectly contextual variable, while 0 indicates a non-contextual variable). For the real datasets (electricity and email datasets), since the actual contextual characteristics are not known in advance, the significance of the obtained results is tested based on a non-parametric permutation test [27]. The idea of this test is as follows: randomly shuffle (reorder) the data to generate independent series and then compare the estimated value of the test statistic (here CMI) from the original data with the distribution of this value from the permuted data. We perform 1000 such permutations, and report the corresponding p-value, obtained by counting the number of times the value of CMI in the permuted data is at least as extreme as that of the observed value in the actual data, divided by the number of permutations. Note that, the effectiveness of the approach is also evaluated in terms

of improving classification performance when the identified contextual information is exploited during classification as will be illustrated in Chapter 5.

Related Approaches. Looking at the input variable selection literature, existing approaches are usually grouped into two main categories [53]: filters and wrappers. Wrapper methods usually assess the worth of input variables by selecting the combination that achieves the best performance of the target classification model. On the other hand, filter methods assess the worth of input variables independently of the classification model utilising criteria based on general characteristics of the data. Given that we evaluate and utilise context variables at a meta-level independently of the classification model, filter methods are more relevant and applicable for our case. Besides the presented and commonly used mutual information, there are other filter measures in the literature that could be utilised for the purpose of context identification. This includes classical statistical approaches such as t-Statistics [55], Chi-square test, and symmetrical Uncertainty [54]. These approaches are compared against our approach in Section 4.4.7.

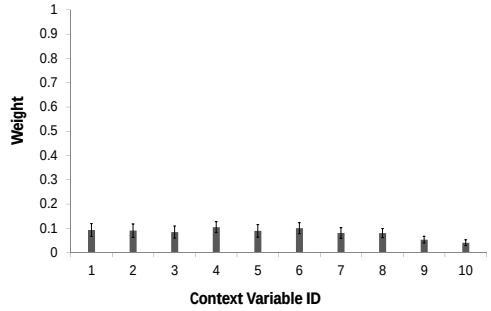
4.4.3 Effect of Speed of Changes

In this section, we test how the proposed context identification approach is affected by different types of concept changes (abrupt or gradual), considering varying speed of changes. For this purpose, utilising the above mentioned data generation framework, we generate 4 concepts with 5 normally distributed input variables. In the case of abrupt changes, we vary the value of parameter p as follows: 10, 50, 100, 200, 500 and 1000. In the case of gradual changes, we fix the duration of each concept to 500 and vary the parameter ω with values: $\frac{1\alpha}{500}$, $\frac{5\alpha}{500}$, $\frac{10\alpha}{500}$, $\frac{20\alpha}{500}$. For example, with a step size of 5, μ_y increases by $\frac{5\alpha}{500}$ after 5 examples are generated. In a similar way, the perfectly contextual variable, based on which correlated context variables are generated (see

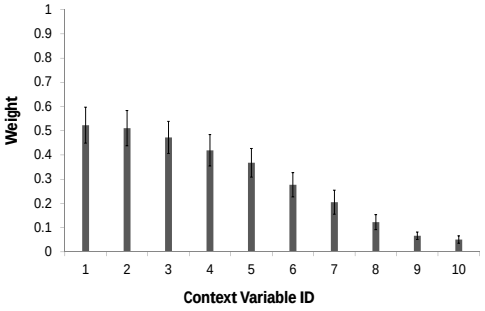
Chapter 3), is also made to change gradually at different time steps as indicated above. Figures 4.2 and 4.3 report the corresponding results of the considered speed of changes for the two cases of abrupt and gradual concept changes.

In Figure 4.2, it can be seen that starting from 50 examples per concept change, the proposed approach is able to recognise the relative importance among candidate context variables. Specifically, contextual variables with the highest importance (largest correlations in this case) get the largest weights, followed by the less important variables that get smaller weights. We can also observe that reducing the speed of change (i.e. increasing the sample size per concept) results in a better recognition of the actual importance of context variables. For example, variable C_{10}, which has no importance in distinguishing between concepts (with true correlation of 0.1), gets a weight around 0.10 with concept duration of 50, while it gets a weight around 0 when the concept duration is 1000. Similarly, variable C_1 (the perfectly contextual one with true correlation of 1), gets a weight around 0.7 when the concept duration is 1000, while it reaches only 0.5 with a concept duration of 50.

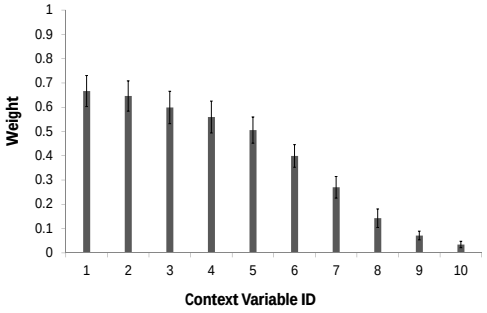
Now in the case of gradual concept changes, the results in Figure 4.3 show that the relative importance between variables is also recognised for all step sizes (the variables are correctly ranked according to their importance). On the other hand, we can see that the learned weights are less sensitive to gradually drifting concepts as they behave similarly across different speeds of changes (i.e. across all step sizes). The reason for that is the non-incremental learning scenario assumed in the experimental analysis.



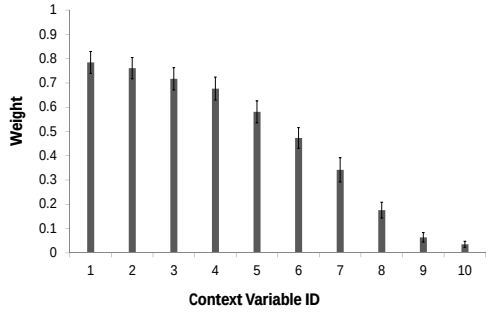
(a) 10 examples per concept



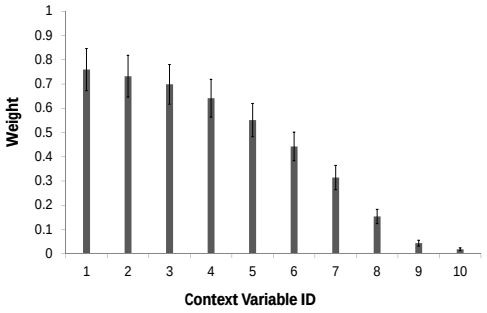
(b) 50 examples per concept



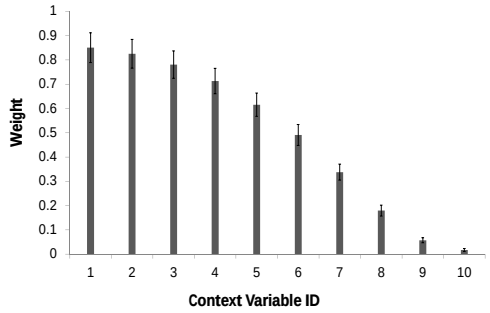
(c) 100 examples per concept



(d) 200 examples per concept

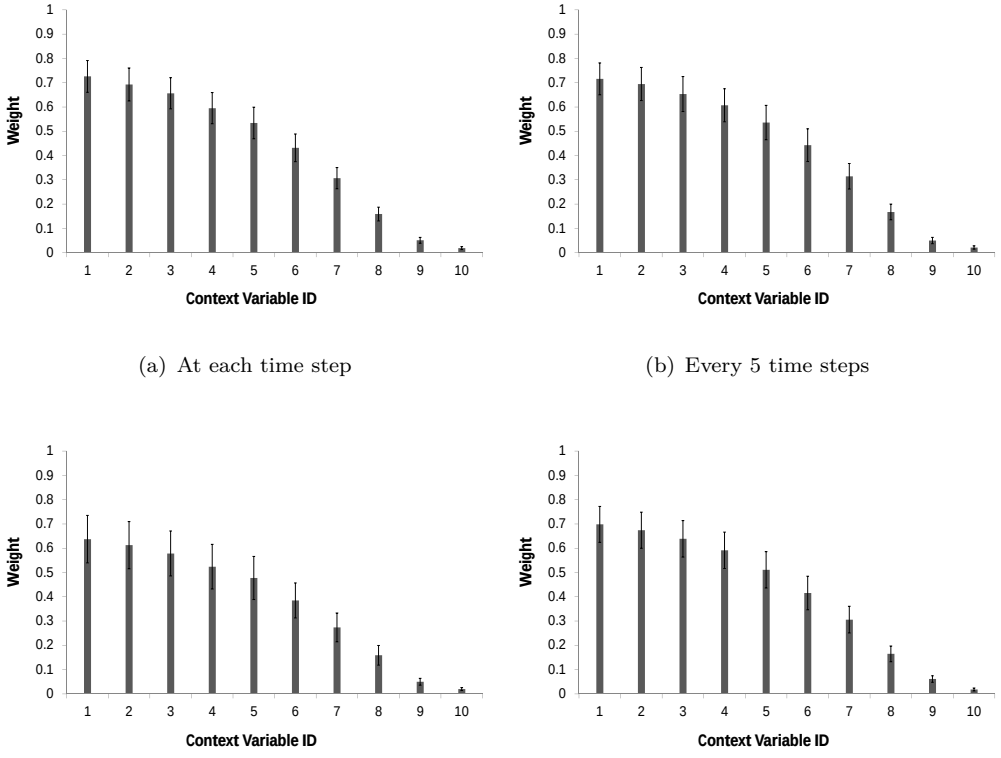


(e) 500 examples per concept



(f) 1000 examples per concept

FIGURE 4.2: Evaluation of context identification strategy given various speed of changes: abrupt concept changes. The error bars denote a 95% confidence intervals.



(a) At each time step

(b) Every 5 time steps

(c) Every 10 time steps

(d) Every 20 time steps

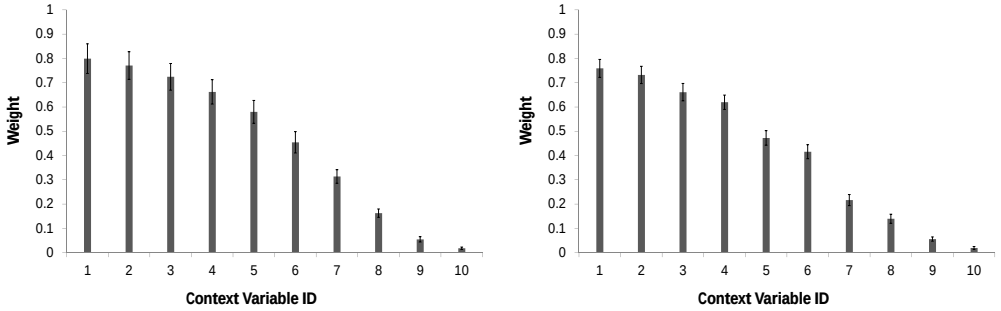
FIGURE 4.3: Evaluation of context identification strategy given various speed of changes: gradual concept changes. The error bars denote a 95% confidence intervals.

Based on the above, although the approach performs reasonably well with various abrupt concept durations, the experiments show that reducing the sample size worsens the accuracy of the learned relevance. Moreover, in the case of gradual changes, the approach is less sensitive to the speed of changes.

4.4.4 Effect of Dimensionality

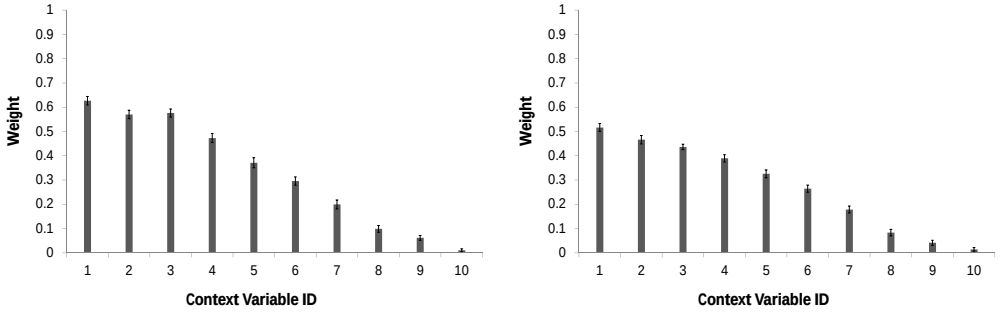
Our aim in this section is twofold. First, to study the effect of the input dimensionality of the dataset on the accuracy of the proposed approach. Second, to study the behaviour of the proposed approach under varying context dimensionality. For this purpose, we utilise the same experimental setup described in Section 4.4.3 with regard to concept change settings. However, here we fix the duration of each concept to 500, and vary the dimensionality of the dataset as described below.

First, we vary the dimension of X from 5 to 30 as illustrated in Figure 4.4. As can be seen, increasing the dimensionality of the dataset affects the accuracy of the approach to identify the actual importance of each variable. However, the relative importance among variables is preserved in all cases. In the second experiment, we fix the dimension of X to 5 and increase the context dimensionality from 10 to 30. To do this, we generate 10 relevant variables (1-10), 10 redundant variables (11-20), and 10 irrelevant variables (21-30). The results presented in Figure 4.5 shows the robustness of the approach to the increased dimension. This is because, both relevant and redundant variables achieve similar results, while the weights of the irrelevant variables fluctuates around zero. Note that, in this experiment, the redundancy among variables is not addressed, and hence relevant and redundant variables get similar weights. Handling redundancy will be tested in Section 4.4.10.



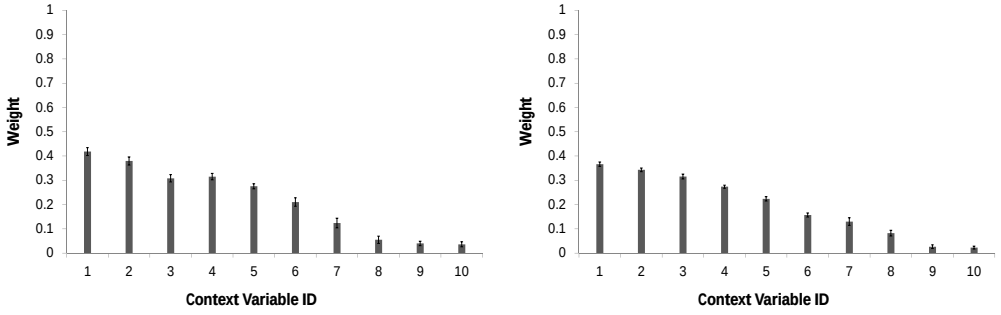
(a) 5 input variables

(b) 10 input variables



(c) 15 input variables

(d) 20 input variables



(e) 25 input variables

(f) 30 input variables

FIGURE 4.4: Evaluation of context identification strategy given various input dimensionality.

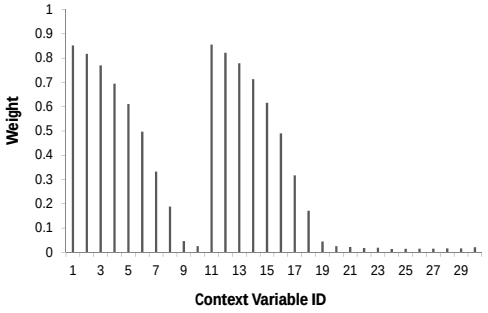


FIGURE 4.5: Evaluation of context identification strategy with higher context dimension: 1-10 relevant variables, 11-20 redundant variables, 21-30 irrelevant variables.

4.4.5 Effect of Noise

In this section, we study the influence the noise level has on the context identification accuracy. Two types of noise are distinguished in the analysis: label noise and context noise. In the former case, we study the influence of label noise (noise in the target variable). In the latter case, we study the influence of context noise (various levels of overlapping distributions in context variable values among concepts). We utilise the same experimental setup described in the previous section, but this time fixing the input dimension and context dimension to 5 and 10, respectively.

First, to test the influence of the label noise, we vary the level of noise introduced to each concept from 0 to 5%. For example, a 5% noise means that 5% of the examples have incorrect class labels. The results are presented in Figure 4.6.

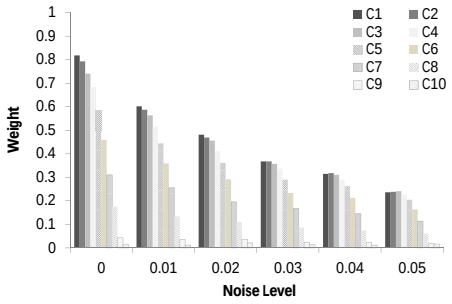
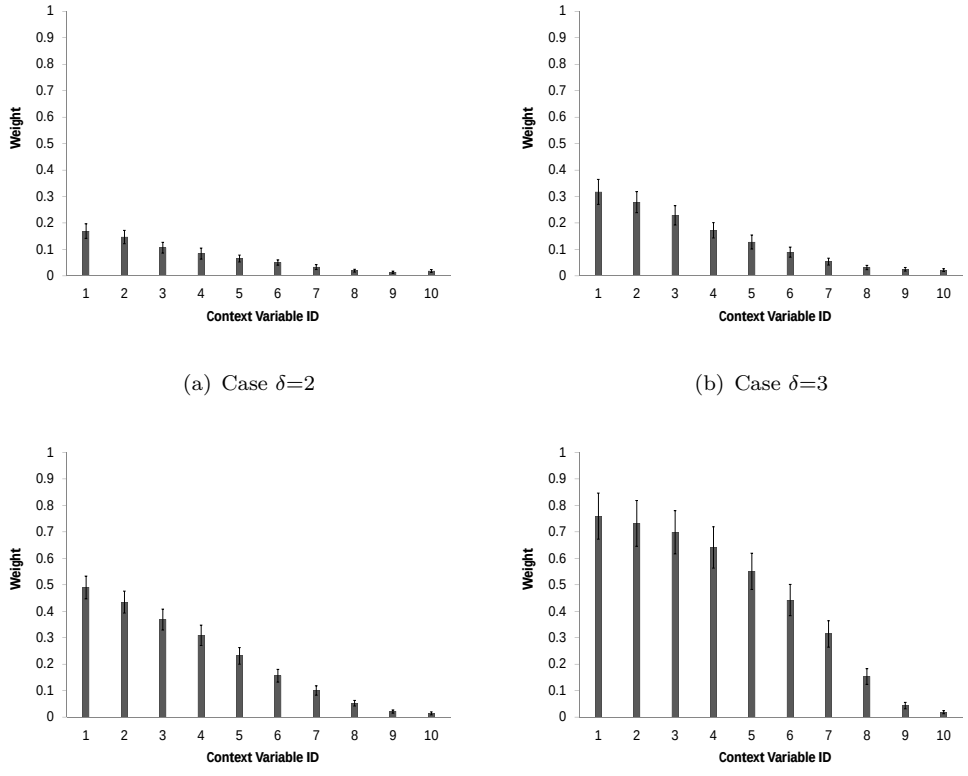


FIGURE 4.6: Evaluation of context identification strategy with varying label noise levels.

As can be seen, although the relative importance between variables is preserved across all noise levels, increasing the noise level decreases the learned contextual importance of the context variables. It is worth mentioning, however, that the dataset utilised in the analysis is not a noise free dataset (given the experimental setups), and hence the introduced label noise add to the already existing noise in the dataset.

Now, we test how the proposed context identification method will perform in settings with noisy context variables. To model such settings, we introduce various levels of an overlap in the distributions of the context variable values. Specifically, the mean of the context variable values is made to change by the following values: 2 (a large overlap between value distributions), 3, 4 and 8 (no overlap between value distributions). The results are presented in Figure 4.7. As expected, when the overlap in the distribution of the context variable values increases, the importance of the context variables decreases. This is because the higher the overlap, the higher the probability that the variable will take similar values across different concepts (i.e. not perfectly contextual variable). As a result, the weight assigned to that variable will decrease.



(a) Case δ=2

(b) Case δ=3

(c) Case δ=4

(d) Case δ=8

FIGURE 4.7: Evaluation of context identification strategy with an overlap in context distributions among concepts.

4.4.6 Static Dataset

In this section, unlike previous experiments, we study how the proposed context identification method will perform in settings with no concept drift. As mentioned previously, in a static environment (or within the same concept), context variables have no predictive ability, and hence should be identified as irrelevant. For this purpose, we generate 2000 examples fixing the data generating distribution to be the same among all four concepts, i.e. $\mu_y^t = \mu_y^{(t-1)}$ at each time step. Note that this dataset exhibits noise

due to the overlap of the class conditional distributions. The results in Figure 4.8 demonstrate the effectiveness of the proposed approach as no variable is identified as contextual (all variables get weights close to zero). Moreover, as stated earlier, the robustness of the approach to correctly identify context in the presence of noise is also illustrated.

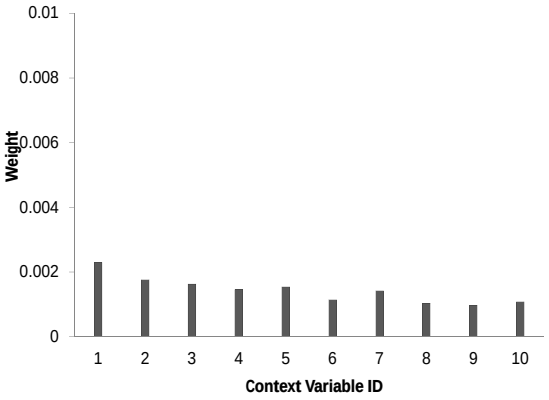


FIGURE 4.8: Evaluation of context identification strategy in a static environment.

4.4.7 Comparative Analysis

In this section, we are comparing the measure we adopted (conditional mutual information) for assessing the importance score of context variables (and hence for their selection), against other popular measures that could alternatively be utilised for this purpose. We generate 2000 examples utilising the same experimental setup introduced in the previous sections for both abrupt and gradual concept changes. In utilising the other variable selection methods, the importance score is computed for each variable based on these measures. In particular, the score for t-statistics is computed according to the following formula [55]: $\frac{|\mu_j^1 - \mu_j^0|}{\sqrt{\frac{(\sigma_j^1)^2}{n_1} + \frac{(\sigma_j^0)^2}{n_0}}}$, where μ_j^1, σ_j^1 are the mean and standard deviation, respectively, of variable C_j in examples with class 1, μ_j^0, σ_j^0 are the mean and standard deviation, respectively, of variable C_j in examples with class 0, n_1 and n_0 are

the number of examples in each class. The score for symmetrical uncertainty is defined as [54]: $2 * \frac{H(C_j)+H(Y)-H(C_j,Y)}{H(C_j)+H(Y)}$, where C_j is the variable of interest, and Y is the target class label. The chi-squared statistic is defined by: $\Sigma_y \Sigma_{c_j} \frac{(A_{yc_j}-E_{yc_j})^2}{E_{yc_j}}$, where A_{yc_j} is the number of examples with class $Y = y$ and $C_j = c_j$, E_{yc_j} is given as $\frac{R_{c_j} * V_y}{n}$ with n is the number of examples available, R_{c_j} is the number of examples with $C_j = c_j$, V_y is the number of examples with $Y = y$. The results are depicted in Figures 4.9 and 4.10 for abrupt and gradual changes, respectively.

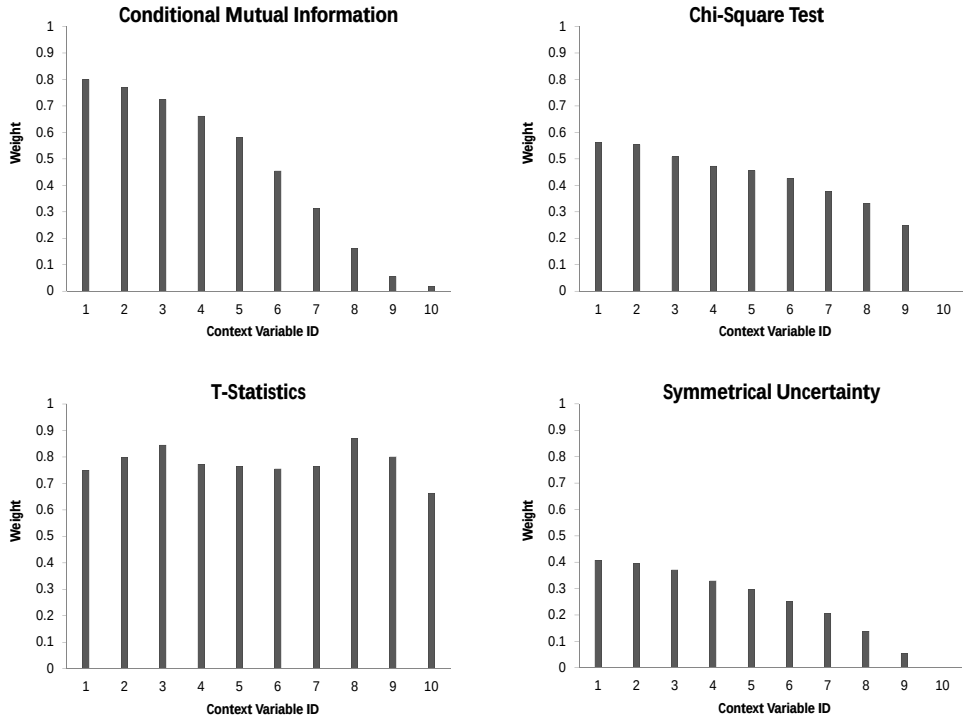


FIGURE 4.9: Comparison of context identification strategies: abrupt changes.

We can see that the conditional mutual information measure outperforms the other approaches in all cases. This is because, as mentioned previously, identifying the actual importance of the context variables requires taking into consideration the input

variables, rather than measuring the direct influence it has on the target variable (the case of the other measures).

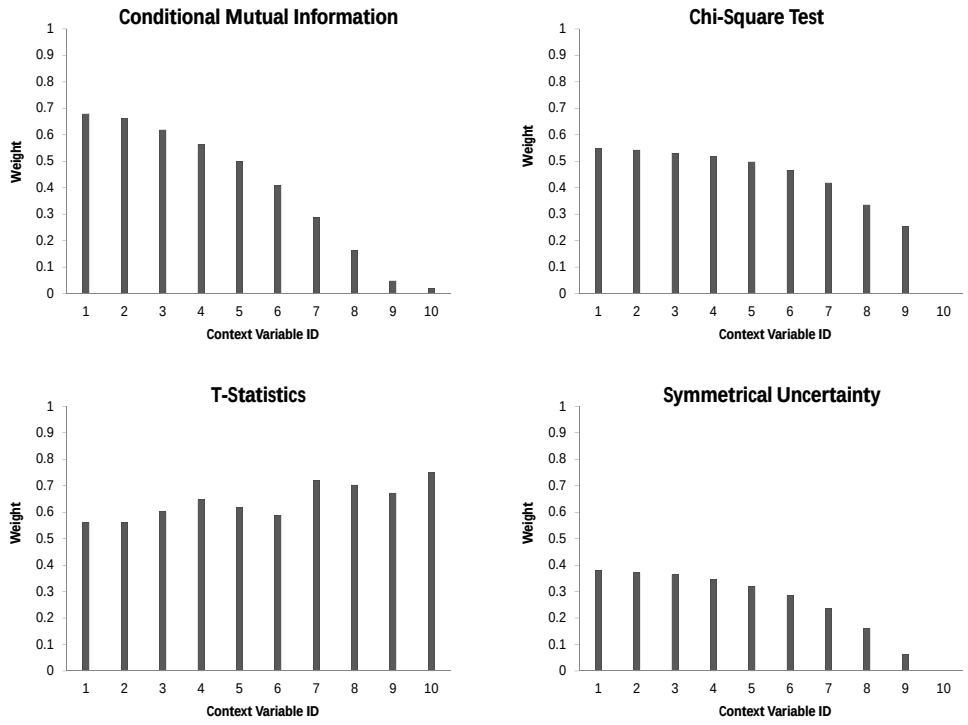


FIGURE 4.10: Comparison of context identification strategies: gradual changes.

4.4.8 Additional Benchmark Datasets

In this section, we apply the proposed context identification method with other datasets that are commonly used in the literature. As with the above experiments, we generate 10 candidate context variables of various importance. The results are reported in Figures 4.11, 4.12 and 4.13.

As can be noticed, the relative importance amongst variables is recognised in all datasets. On the other hand, in line with previously conducted experiments, we can see that context identification performs best in datasets with abrupt changes, and in the

presence of noise, the performance of the approach is negatively affected in recognising the importance of context variables compared to noise-free settings. For example, variable C_1 gets a weight of more than 0.8 with Sine1 dataset (abrupt changes-noise free settings), while it gets only 0.5 weight with Gauss dataset (abrupt changes-high level of noise), and a weight under 0.7 for Circles dataset (gradual changes).

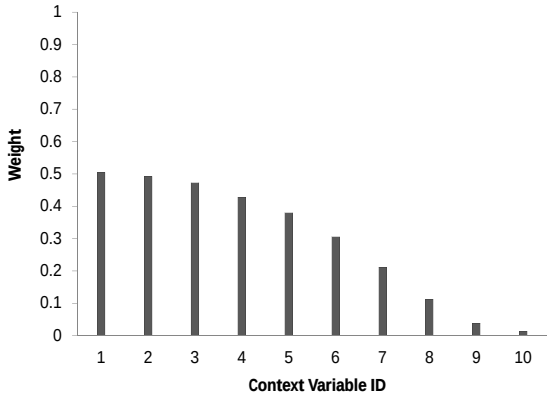


FIGURE 4.11: Evaluation of context identification strategy: Gauss dataset.

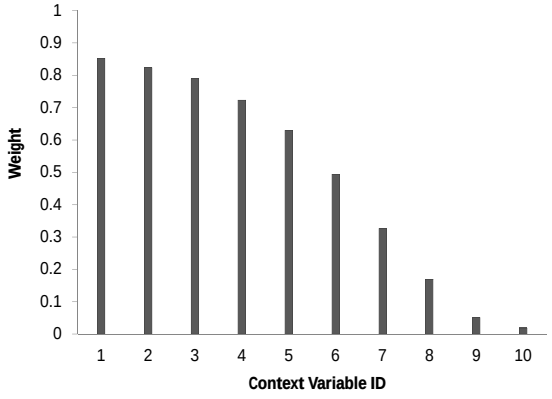


FIGURE 4.12: Evaluation of context identification strategy: Sine1 dataset.

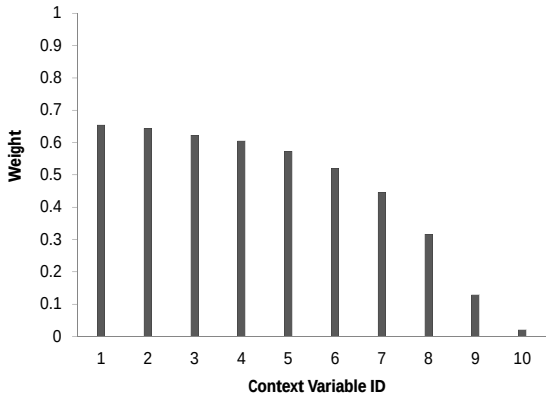


FIGURE 4.13: Evaluation of context identification strategy: Circles dataset.

4.4.9 Real-World Datasets

Now, we apply the proposed context identification approach to deduce the actual contextual variables for the electricity and email datasets. In the electricity dataset (Figure 4.14), the results indicate that both day of the week and season are relevant for characterising context (both get CMI values greater than zero), while the noisy variables are irrelevant (all get values close to zero). Similarly, in the email dataset (Figure 4.15), only location is recognised as relevant context variable. We can also see that the permutation test in both datasets confirms the significance of the obtained CMI values for the identified contextual variables (day of the week and season for the electricity dataset, and location for the email dataset), and the insignificance for the noisy variables.

Variable	CMI	Permutation Test (p-value)
Day of the Week	0.1615	0
Season	0.1124	0
N1	0.0299	1
N2	0.030	1
N3	0.0288	1

FIGURE 4.14: Conditional mutual information estimates for the candidate context variables of the Electricity dataset.

Variable	CMI	Permutation Test (p-value)
Location	0.4518	0
N1	0.0755	1
N2	0.0698	1
N3	0.0768	1

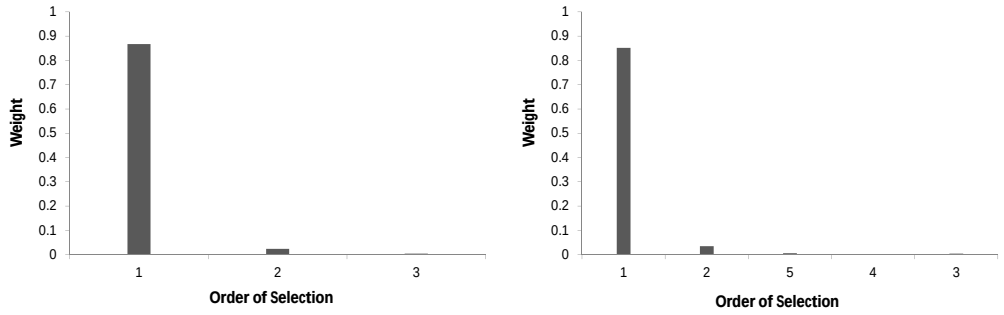
FIGURE 4.15: Conditional mutual information estimates for the candidate context variables of the Email dataset.

4.4.10 Correlation Handling

In this section, we examine the ability of the proposed context identification approach to address correlation among context variables. In particular, the goal is to test the effectiveness of the proposed context identification approach to identify redundant context variables (i.e. select only relevant variables and eliminate redundant variables).Unlike previous experiments, the importance of each candidate context variable is evaluated here taking into consideration other variables previously identified as contextual.

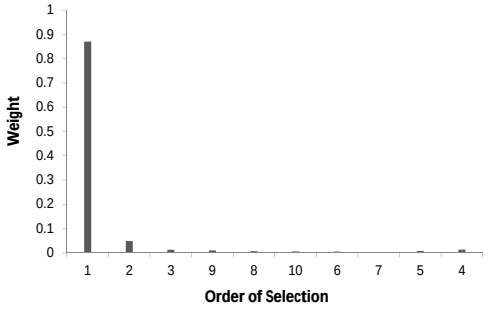
First, we test the behaviour of the proposed context identification method given limited context dimensionality. In the second part, due to computational problems as mentioned in Section 4.3.4, we incorporate existing approximation approaches for handling correlations in the case of high variable dimensionality.

In the first part of the analysis, we consider 3 cases for context generation with various dimensionality. In the first case, we only generate 3 variables, with their importance being defined as follows: variable C_1 is relevant, while variables C_2 and C_3 are redundant (with correlations of 1, 0.9, 0.8, respectively). In the second case, we increase the dimensionality to 5 variables, where only variable C_1 is relevant, while the other variables are redundant (with correlations varying from 1 to 0.5 with a step size of 0.1). In the third case, we generate 10 variables, where only the first variable is relevant, while the other variables are redundant (with correlations varying from 1 to 0.1 with a step size of 0.1). Here, increasing the dimensionality beyond 10 would increase the computational problems of the context selection algorithm, which requires utilising one of the approximation methods. These methods are tested in the next experiment. The results are presented in Figures 4.16 and 4.17, given abrupt and gradual changes, respectively. The effectiveness of the approach is illustrated in all cases, since only the first variable is identified as contextual, while other variables, despite their individual contextual relevance (in this case the correlation degree), get values close to zero. This is because in the presence of one perfectly contextual variable, as in the considered experiments, all the other variables become redundant (provide equivalent contextual information), which is successfully captured by the utilised selection criterion $I(I(Y, C_j|\{X, S\}))$.



(a) 3 candidate context variables

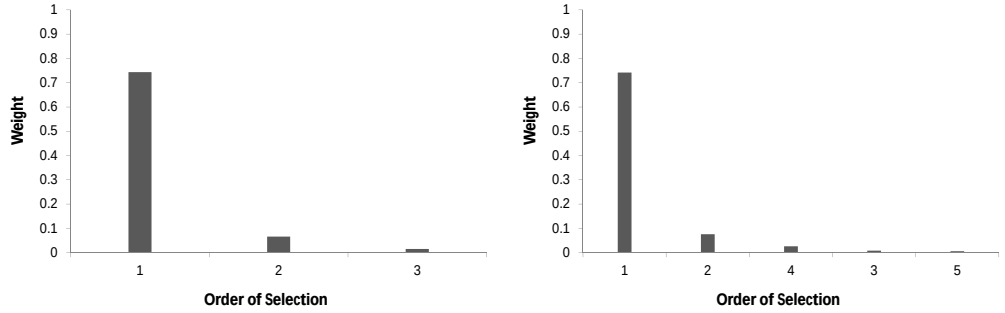
(b) 5 candidate context variables



(c) 10 candidate context variables

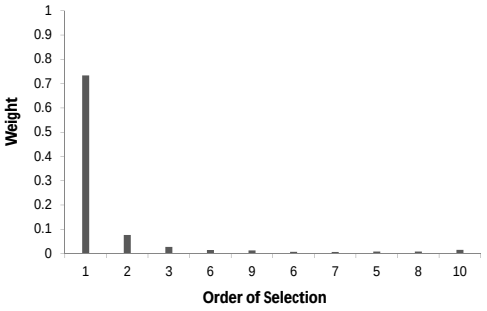
FIGURE 4.16: Evaluation of context identification strategy for correlation handling: abrupt changes

Despite the efficiency of the proposed approach in addressing redundancy illustrated in the above experiments, increasing context dimensionality (given limited sample size) will affect the behaviour of the approach and require incorporating some approximation methods. To illustrate this, we increase context dimensionality by generating 15 candidate context variables, with their importance being defined as follows: 1-5 the variables are relevant (with correlation values varying from 1 to 0.6), 6-10 the variable are redundant (perfectly correlated with variables 1-5), and 11-15 the variables are irrelevant (with zero correlations).



(a) 3 candidate context variables

(b) 5 candidate context variables



(c) 10 candidate context variables

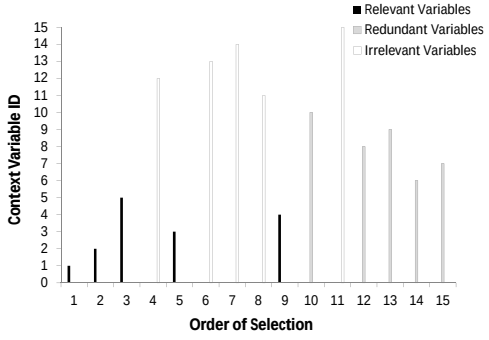
FIGURE 4.17: Evaluation of context identification strategy for correlation handling: gradual changes

Here, in addition to the proposed context selection approach, we compare how the existing approximation algorithms mentioned in Section 4.3.4 perform in the case of our problem. The aim is to select first the relevant context variables ordered in ascending order (the first selected variable is the more relevant), followed by the redundant and irrelevant variables. In all approximation approaches, CMI is estimated with the help of the k-nearest neighbour estimator. The values of parameters β and θ in MIFS and CMIFS approaches are set to 0.5 and 0.2, respectively. Figures 4.18 and 4.19 report the corresponding results for abrupt and gradual changes, respectively.

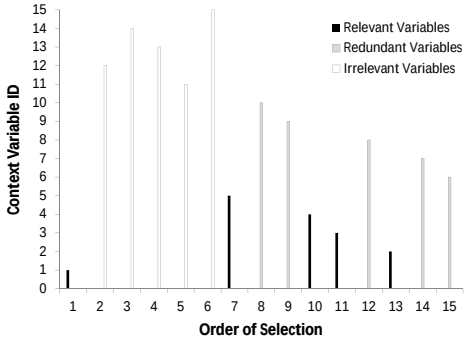
As can be seen, the proposed approach is still performing reasonably well compared to

some of the considered approximation methods (such as mRMR and MIFS approaches). However, as expected, increasing context dimensionality affects the behaviour of the approach, resulting in some ordering inaccuracies due to computational problems. Specifically, we can see that 4 out of 5 irrelevant variables with our approach are selected prior to other relevant variables in the case of abrupt changes, and 5 out of 5 in the case of gradual changes. For example, with abrupt changes (Figure 4.18(a)), irrelevant variable C_{12} is selected prior to the relevant variable C_3. Similarly, in the case of gradual changes (Figure 4.19 (a)), many irrelevant variables are selected prior to the relevant variable C_3.

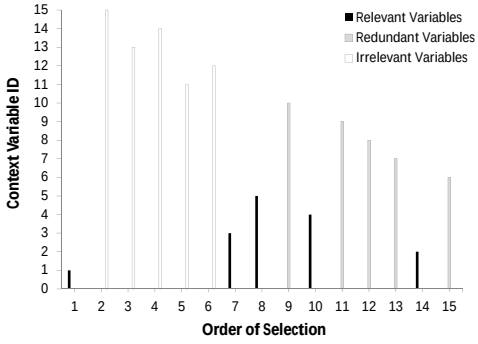
On the other hand, in the approximation approaches, the best performing algorithms are NMIFS and CMIFS. Although these approaches suffer from some inaccuracies with regard to addressing redundancy among variables. For example, in NMIFS approach (see Figure 4.18(d)), variables C_6 and C_1 are selected subsequently despite being perfectly correlated with each other. Similarly, with CMIFS approach (see Figure 4.18(e)), variables C_6 and C_7 are selected along with their correlated counterparts C_1 and C_2. Similar behaviour can also be observed in Figure 4.19 with gradual changes. Despite such inaccuracy with handling redundancy, in both NMIFS and CMIFS approaches, all relevant variables are selected prior to the irrelevant ones unlike the other mRMR and MIFS approaches. Hence, in high dimensional problems, either NMIFS or CMIFS can be utilised for the purpose of our analysis.



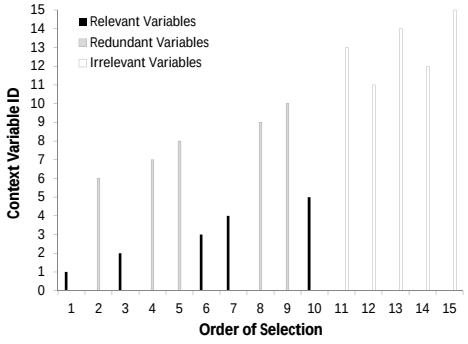
(a) The proposed CMI Algorithm



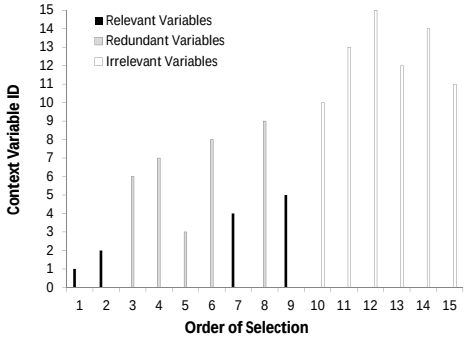
(b) MIFS Algorithm



(c) mRMR Algorithm

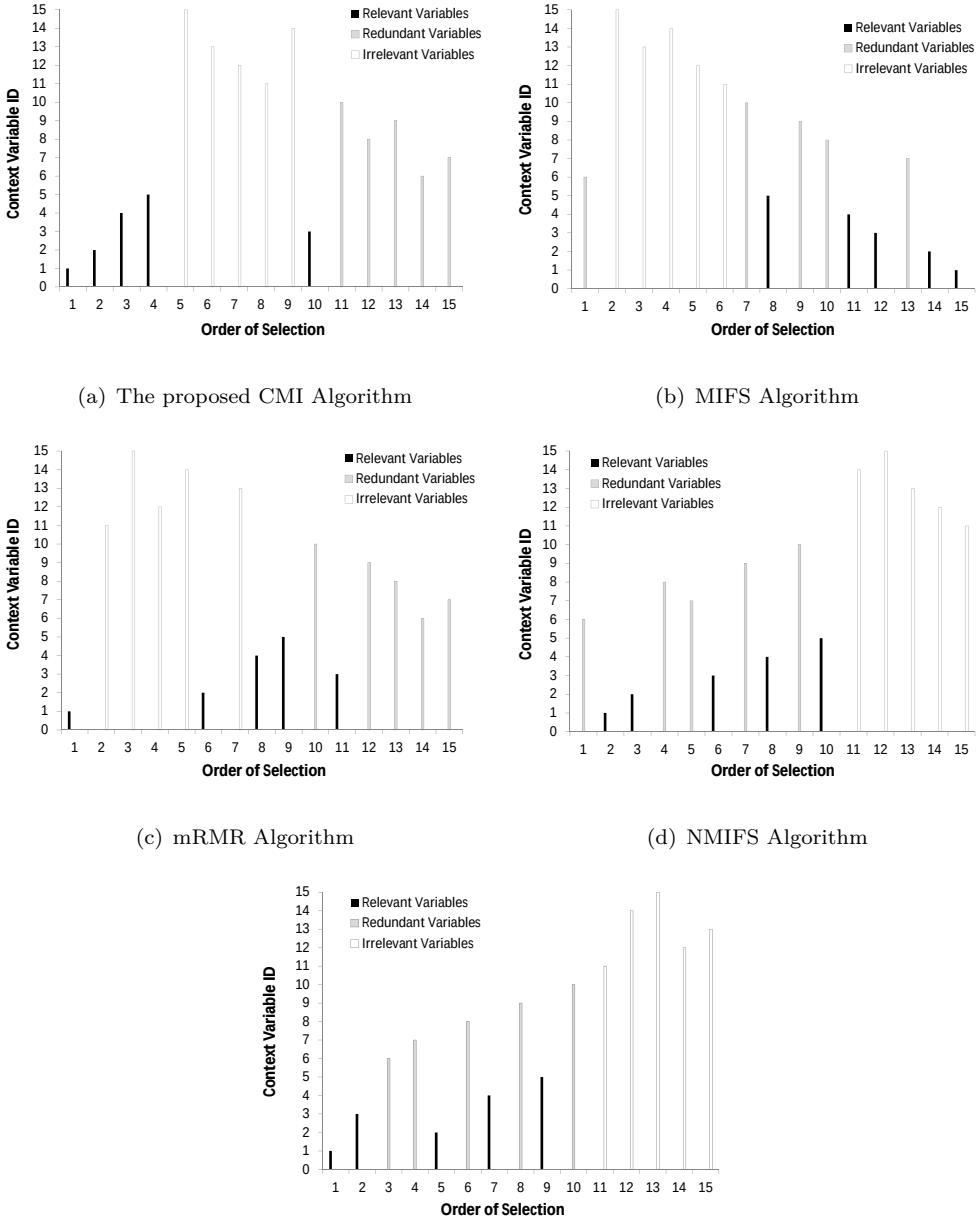


(d) NMIFS Algorithm



(e) CMIFS Algorithm

FIGURE 4.18: Comparison of different approximation algorithms for context variables selection: abrupt changes.



(a) The proposed CMI Algorithm

(b) MIFS Algorithm

(c) mRMR Algorithm

(d) NMIFS Algorithm

(e) CMIFS Algorithm

FIGURE 4.19: Comparison of different approximation algorithms for context variables selection: gradual changes.

4.5 Conclusion

Exploiting context information in adapting the classification model to concept changes enables capturing the indicators of drift, and hence facilitating more effective adaptation. Such exploitation, however, requires identifying relevant context variables that actually affect the concept of interest. In this chapter, we have presented a systematic information-theoretic-based approach (utilising entropy and conditional mutual information measures) for relevant context identification from available historical data. The computation models required for its implementation are also illustrated.

Experimental results demonstrated the ability of the proposed approach to recognise the differential importance among candidate context variables, and to rank these in accordance to this importance. The feasibility and robustness of the approach are tested with respect to a number of factors, including type of change (i.e. whether changes between concepts happen abruptly or only gradually), speed of change, presence of noise, and erroneous context recognition. In addition, a solution for correlation handling and addressing high problem dimensionality are presented.

A main assumption in the experiments presented is the availability of sufficient and representative training data of the domain when identifying relevant context variables. Evaluating the behaviour of the approach in the case of gradual availability of the training data represent an interesting extension of our approach, but is out of the scope of this thesis.

In the next chapter, we demonstrate how the identified contextual knowledge improves the prediction accuracy of the classification model via a data weighting model, which facilitates weighting of training examples according to their relevance.

Chapter 5

Context Exploitation Model - Example Weighting

5.1 Introduction

In classification problems, the predictive model at time step t is normally derived based on the previously observed labeled examples $\{u(i)\}_{i=1}^{t-1}$. As we discussed earlier, in the presence of concept drift, not all such past examples necessarily remain relevant, and hence a selection mechanism is needed to determine which examples should have a (higher) impact on the learning process at hand (i.e. on learning the predictive model at the current time step t). Existing approaches use *recency* as a measure for selecting relevant training examples, with recent examples being considered the most relevant, with older ones being forgotten eventually. In particular, the classifier is (re-)estimated using either a window of the latest observed examples (e.g. [6, 7]), or a time-based example weighting function (e.g. [12, 15]). However, we argue that recency is not necessarily an accurate indicator of example relevance. This is because, recent observations

may have been affected by circumstances that have now changed, and thus recency will weight such examples higher than older but potentially more relevant ones. Moreover, old examples may remain good predictors when their circumstances are comparable to current circumstances (the case of reappearing concepts). Therefore, we argue that the contextual circumstances of past examples should be recorded and incorporated as additional clues when assessing example relevance.

In response, this chapter builds on the contextual knowledge identified in the previous chapter (Chapter 4), and utilises such knowledge to control the contribution of each training example in estimating the classification model parameters according to similarity to current contextual circumstances. In particular, we propose to *weight* each training example based on the degree of similarity between the context under which the training example is collected and the current context. That is, data examples collected under context conditions more similar to the current context (the context of the example to be classified) should receive higher weights, and the weights should decrease correspondingly as this similarity decreases.

The rest of the chapter is organised as follows. An abstraction of existing classification models, upon which our context-aware extension is proposed, is presented in Section 5.2, followed by the proposed context-driven extension in Section 5.3. Evaluation setup and results are reported in Section 5.4, while Section 5.5 concludes the chapter.

5.2 Base Classification Model

From existing classification approaches, we can derive a generic abstraction of a classification model, which will serve as a *base* for our context-aware extension. This abstraction, which we henceforth in this chapter refer to as the *base classification model* (and denote it with the subscript b), can be summarised as a tuple,

$$\langle O^t, W_b^t(u(i)), P_b^t(Y|X) \rangle \tag{5.1}$$

where:

- O^t is the set of available past observations (i.e. labelled training examples) of the system at hand up to the current time step t. That is,

$$O^t = \{u(i) \mid u(i) = \langle x(i), y(i) \rangle\}_{i=1}^{t-1}$$

  where $x(i) = (x_1(i), x_2(i), ..., x_m(i))$ represents the value vector of input variables at time step i, and $y(i)$ is the respective target class label.

- $W_b^t(u(i))$ is an example weighting factor governing the contribution (relevance) of each training example $u(i) \in O^t$, for estimating the model at time step t. Such example weights would potentially change over time. A commonly utilised factor for evaluating an example's weight is the point in time i at which the example is collected (i.e. an example's weight decreases over time). Note that for models that do not support example weighting, $W_b^t(u(i)) = 1, \ \forall i$.

- $P_b^t(Y|X)$ is the estimated model at time step t, reflecting the approximated probability of obtaining an output label $\hat{y}$, given an input vector x. It captures the approximation, at time step t, of the underlying function between the target class label and the respective input data. This approximation is typically estimated by applying some learning function L, over available past observations $u(i) \in O^t$, while accounting for their weights $W_b^t(u(i))$, i.e.

$$P_b^t(Y|X) = L(\ \{\ \langle W_b^t(u(i)), u(i) \rangle\ \}_{i=1}^{t-1}\ ) \tag{5.2}$$

The type of this learning function depends on the classification model considered. Example instantiations corresponding to two commonly used classification models in the literature are presented in the following subsections.

5.2.1 Naive Bayes Classifier

In the basic Naive Bayes classification model (see Chapter 3), which we denote here with the subscript n, approximated probability $P_n^t(Y|X)$ of obtaining class label $\hat{y}$, given an input vector $x = (x_1, x_2, ..., x_m)$, is given by,

$$p_n(\hat{y}|x) = p_n(\hat{y}) \times \prod_{j=1}^{m} p_n(x_j|\hat{y}) \tag{5.3}$$

Note that superscript t is omitted from probabilities p_n above for readability, but these probabilities are still time-dependent (i.e. may differ at different time steps depending on the knowledge available to the model).

The estimation of probabilities $p_n(\hat{y})$ and $p_n(x_j|\hat{y})$ is easily achieved via maintaining corresponding value counts, as follows,

$$p_n(\hat{y}) = \frac{count(\hat{y})}{|O^t|} \tag{5.4}$$

$$p_n(x_j|\hat{y}) = \frac{count(x_j \wedge \hat{y})}{count(\hat{y})} \tag{5.5}$$

where $|O^t|$ is the total number of past observations, $count(\hat{y})$ is the number of past observations with class label $\hat{y}$, and $count(x_j \wedge \hat{y})$ is the number of past observations with both class label $\hat{y}$ and value x_j for input attribute X_j. Functions $count(\hat{y})$ and

$count(x_j \wedge \hat{y})$ are given as follows,

$$count(\hat{y}) = |\{\langle x(i), y(i)\rangle \in O^t \mid y(i) = \hat{y}\}| \tag{5.6}$$

$$count(x_j \wedge \hat{y}) = |\{\langle x(i), y(i)\rangle \in O^t \mid y(i) = \hat{y} \quad \wedge \quad x_j(i) = x_j\}| \tag{5.7}$$

To account for the case of zero frequencies in Equations 5.4 and 5.5, the Laplace correction can be utilised adjusting the counts as follows,

$$p_n(\hat{y}) = \frac{1 + count(\hat{y})}{|Y| + |O^t|} \tag{5.8}$$

$$p_n(x_j|\hat{y}) = \frac{1 + count(x_j \wedge \hat{y})}{|X_j| + count(\hat{y})} \tag{5.9}$$

where $|Y|$ is the number of possible class labels, and $|X_j|$ is the number of possible values for input variable X_j.

Now, the above basic Naive Bayes classifier can be rewritten in terms of our generic (base) classification model of Equation 5.1, as follows. According to the basic Naive Bayes classifier, all available data examples are considered equally important. Thus,

$$\forall u(i) \in O^t, \quad W_b^t(u(i)) = 1 \tag{5.10}$$

The approximated probability $P_b^t(Y|X)$ is as in Equation 5.3,

$$p_b(\hat{y}|x) = p_b(\hat{y}) \times \prod_{j=1}^{m} p_b(x_j|\hat{y}) \tag{5.11}$$

where probabilities $p_b(\hat{y})$ and $p_b(x_j|\hat{y})$ are given by rewriting probabilities $p_n(\hat{y})$ and $p_n(x_j|\hat{y})$ of Equations 5.8 and 5.9 in terms of weights W_b^t, as follows,

$$p_b(\hat{y}) = \frac{1 + \sum_{i=1}^{t-1} check(y(i) = \hat{y}) \times W_b^t(u(i))}{|Y| + \sum_{i=1}^{t-1} W_b^t(u(i))} \tag{5.12}$$

$$p_b(x_j|\hat{y}) = \frac{1 + \sum_{i=1}^{t-1} check(y(i) = \hat{y}) \times check(x_j(i) = x_j) \times W_b^t(u(i))}{|X_j| + \sum_{i=1}^{t-1} check(y(i) = \hat{y}) \times W_b^t(u(i))} \tag{5.13}$$

where function $check(a = b)$ returns 1 if $a = b$, and returns 0 otherwise.

In the case of continuous input data, weights are introduced by computing the weighted mean and weighted variance from the training data (assuming data is normally distributed as explained in Chapter 3).

5.2.2 Logistic Regression

In the Logistic Regression classifier (see Chapter 3), which we denote here with the subscript l, the approximated probability $P_l^t(Y|X)$ of obtaining class label $\hat{y}$, given an input vector $x = (x_1, x_2, ..., x_m)$, is given by,

$$p_l(\hat{y} = 1|x) = \frac{e^{\beta^T z}}{1 + e^{\beta^T z}} \tag{5.14}$$

where $z = (1, x_1, x_2, ..., x_m)$, and β is the vector of regression parameters that need to be estimated to fit the current training data points O^t as well as possible. Again superscript t is omitted from probability p_l and parameters β for readability (but these differ at different time steps).

Parameters β are derived by maximising the following log-likelihood function,

$$LogLik_l = \sum_{i=1}^{t-1} \varphi(\beta|u(i)) \tag{5.15}$$

with function $\varphi(\beta|u(i))$ being given in Equation 3.8 (see Chapter 3).

In the above Logistics Regression classifier, learning from past examples is conducted via estimating parameters β. Within this estimation, all examples are considered equally important in the log-likelihood function. This classifier can thus be rewritten in terms of our generic (base) classification model of Equation 5.1, as follows. Since the model does not discriminate between training examples, it follows that,

$$\forall u(i) \in O^t, \quad W_b^t(u(i)) = 1 \tag{5.16}$$

Based on this, the log-likelihood function can be rewritten in terms of these weights as,

$$LogLik_b = \sum_{i=1}^{t-1} W_b^t(u(i)) \times \varphi(\beta|u(i)) \tag{5.17}$$

Finally, $P_b^t(Y|X)$ is given as in Equation 5.14, utilising $LogLik_b$ for estimating parameters β.

5.3 Context-Aware Classification Model

Based on the contextual knowledge identified in Chapter 4, we can define the vector of *relevant* context variables, as,

$$(C_1, C_2, ..., C_r), \quad \text{such that} \ \ \forall j = 1...r, \quad rl(C_j) > 0 \tag{5.18}$$

where $rl(C_j)$ is the relevance degree of context variable C_j (as explained in Chapter 4).

We propose to utilise knowledge of such relevant contextual clues in order to scale the contributions of past examples when learning the predictive model. In particular, we extend the base classification model of Equation 5.1 to account for context information. We denote this context-aware extended version with the subscript c, and it is given as

follows,

$$\langle O_c^t, W_c^t(u(i)), P_c^t(Y|X)\rangle \tag{5.19}$$

where:

- O_c^t is the set of past data observations (labelled training examples) up to the current time step t, extended with context information. That is,

  $$O_c^t = \{\langle u(i), c(i)\rangle\}_{i=1}^{t-1}$$

  where $c(i) = (c_1(i), c_2(i), ..., c_r(i))$ is the context instance under which example $u(i)$ was collected. It corresponds to the value vector of the relevant context attributes $(C_1, C_2, ..., C_r)$, at time step i.

- $W_c^t(u(i))$ is the *context-aware* example weighting factor, which is a combination of the original weighting scheme of the base classification model and of our proposed weighting by context information,

  $$W_c^t(u(i)) = W_b^t(u(i)) \times csim(c(i), c(t)) \tag{5.20}$$

  where $csim(c(i), c(t))$ is the degree of similarity between the context instance under which example $u(i)$ was collected, $c(i)$, and the context instance at the current time step t, $c(t)$. As indicated above, by context instance we refer to the value vector of the relevant context variables at a particular time step. The intuition behind this is that data examples that were collected under circumstances more similar to the current circumstances should be considered more relevant for the current prediction, and thus should have a higher impact on the current learning process (at time step t).

Similarity degree $csim(c(i), c(t))$ can be estimated as follows,

$$csim(c(i), c(t)) = \sum_{j=1}^{r} rl(C_j) \times veq(C_j, c_j(i), c_j(t)) \tag{5.21}$$

where $rl(C_j)$ is the relevance degree of context attribute C_j determining its importance (as explained in Chapter 4), while $veq(C_j, c_j(i), c_j(t)) \in [0, 1]$ is the *value equivalence* function, which determines the degree of equivalence (similarity) between values $c_j(i)$ and $c_j(t)$ of context attribute C_j. We propose two approaches for implementing function veq, which are discussed in Section 5.3.1 and Section 5.3.2.

- Finally, $P_c^t(Y|X)$ is the *context-aware* estimated model at time step t, reflecting the approximated probability of obtaining an output label $\hat{y}$, given an input vector x. It incorporates the contextual information underlying training examples in order to adjust their effect on the learning process (learning the underlying function between the target class label and the respective input data). In particular, $P_c^t(Y|X)$ applies the same learning function of the base classification model (i.e. as in Equation 5.2), but replaces the original weighting scheme W_b^t, with that extended by context information W_c^t. That is,

$$P_c^t(Y|X) = L(\ \{\ \langle W_c^t(u(i)), u(i)\rangle\ \}_{i=1}^{t-1}\ ) \tag{5.22}$$

5.3.1 Simple Approach for Value Equivalence

At a basic level, in order to detect the degree of equivalence between two values v_1 and v_2 of context attribute C_j, i.e. $veq(C_j, v_1, v_2)$, we can compute the direct difference

between v_1 and v_2, as follows,

$$veq(C_j, v_1, v_2) = \begin{cases} 1 - \frac{|v_1 - v_2|}{max(C_j) - min(C_j)} & \textbf{if } C_j \text{ is numerical} \\ 1 & \textbf{if } C_j \text{ is categorical } \textbf{and } v_1 = v_2 \\ 0 & \textbf{if } C_j \text{ is categorical } \textbf{and } v_1 \neq v_2 \end{cases} \tag{5.23}$$

where $min(C_j)$ and $max(C_j)$ are the minimum and maximum values, respectively, of context attribute C_j.

However, this simple realisation of function veq may not necessarily reflect the value equivalence of interest in our approach. This is because we are interested in identifying two values of variable C_j as equivalent if their associated underlying concepts are the same, regardless of the actual direct difference between these values. Hence, an alternative realisation of the value equivalence function would be to *learn* the equivalence among context values from historical data, based on how these values discriminate between occurring concepts. More details are presented next.

5.3.2 Learning-based Approach for Value Equivalence

Two different values v_1 and v_2 of context variable C_j should still be considered similar if the two probability distributions, $P(Y, X|v_1)$ and $P(Y, X|v_2)$, do not differ much under those values. Here, Y and X are the target variable and the vector of input variables, respectively. The intuition behind this is that, as mentioned above, we want to identify two values of a context variable as equivalent if their underlying *concepts* are similar. That is, the degree of equivalence between values v_1 and v_2 of context variable C_j corresponds to the similarity between their underlying distributions $P(Y, X|v_1)$ and $P(Y, X|v_2)$.

To compare the two distributions $P(Y, X|v_1)$ and $P(Y, X|v_2)$, we utilise two alternative information theoretic measures suitable for this purpose: Entropy Absolute Difference

(EAD) [17] and Kullback-Leibler Divergence (KL) [24]. These measures are applied to derive equivalence degrees among context values in an offline mode, assuming existence of a representative training set. However, these measures can also be similarly applied in an online mode, following the availability of new training data, to update the equivalence degrees with more accurate estimations.

The utilised measures are presented below, where probabilities $p(.)$ are estimated based on value frequencies (a discretisation is first applied in the case of continuous context variables).

5.3.2.1 Entropy Absolute Difference

The Entropy Absolute Difference (EAD) measure [17] detects changes between two distributions by measuring the dispersions of the differences between these distributions. In particular, and with reference to our problem, the EAD measure is defined by:

$$H(P(Y,X|v_1)||P(Y,X|v_2)) = \\ -\sum_{y}\sum_{x}|p(y,x|v_1)-p(y,x|v_2)|log_2|p(y,x|v_1)-p(y,x|v_2)| \quad (5.24)$$

Smaller values of this measure indicate smaller differences between distributions. Moreover, if the two distributions are identical, the measure is equal to zero. The other two important properties of this measure are *non-negativity* and *symmetricity*, i.e. the distance from distribution $P(Y,X|v_1)$ to $P(Y,X|v_2)$ is equivalent to the distance from distribution $P(Y,X|v_2)$ to $P(Y,X|v_1)$.

The degree of equivalence between two values v_1 and v_2 of context attribute C_j, i.e. $veq(C_j, v_1, v_2)$, based on EAD is given as follows,

$$veq(C_j, v_1, v_2) = 1 - \frac{H(P(Y, X|v_1)||P(Y, X|v_2))}{maxH} \tag{5.25}$$

where $maxH$ is the maximum value of measure $H(P(Y, X|v_l)||P(Y, X|v_k))$ among any two values v_l and v_k of the context variable. Note that, in the case of an online learning mode, the values of the EAD measure will be updated after each observation, and the maximum value will be determined based on the values observed so far.

5.3.2.2 Kullback-Leibler Divergence

Another commonly used information theoretic-based distance measure that can be utilised in our analysis is relative entropy or Kullback-Leibler divergence (KL) [24]. It is one of the most commonly used measures for quantifying the difference (non-symmetric distance) between distributions, and is computed as follows [24].

$$KL(P(Y, X|v_1)||P(Y, X|v_2)) = -\sum_x \sum_y p(x, y|v_1) log_2 \frac{p(x, y|v_1)}{p(x, y|v_2)} \tag{5.26}$$

This measure is also non-negative and equals zero if and only if the distributions are identical. On the other hand, unlike EAD measure, KL is not symmetric, i.e. $KL(P(Y, X|v_1)||P(Y, X|v_2)) \neq KL(P(Y, X|v_2)||P(Y, X|v_1))$. A commonly used method to overcome this is to take the average of both distances, which is usually referred to as J-divergence [24].

The degree of equivalence between two values v_1 and v_2 of context attribute C_j, i.e. $veq(C_j, v_1, v_2)$, based on KL is given as follows,

$$veq(C_j, v_1, v_2) = 1 - \frac{\text{J-divergence}(P(Y, X|v_1)||P(Y, X|v_2))}{maxJ} \tag{5.27}$$

where $maxJ$ is the maximum value of measure J-divergence$(P(Y,X|v_l)||P(Y,X|v_k))$ among any two values v_l and v_k of the context variable.

5.4 Evaluation

In this section, we conduct empirical evaluation of the proposed context-aware example weighting approach, focusing on its influence on classification performance in terms of avoiding performance degradation and producing accurate predictions in dynamic settings. In particular, we test the behaviour of the approach under varying conditions, and compare its performance against a number of adaptation strategies existing in the literature. For this purpose, we base our evaluation on both artificial and real-world datasets, including Stagger, Hyperplane, Gauss, Sine1, *Elec* and *Elist* datasets. For artificial datasets, the results reported are averaged over 30 runs.

5.4.1 Objectives

In the following experiments, our aim is to achieve the following objectives.

Objective 1. Study the influence of various factors affecting the behaviour of the proposed context-driven weighting approach in adapting the classification model to concept changes.

Objective 2. Compare the proposed context-driven weighting approach under two realisations of the value equivalence function: the simple (direct-difference-based) and the learning-based approaches.

Objective 3. Compare the proposed context-driven weighting approach against incorporating context variables as additional input variables into the classification model.

Objective 4. Compare the proposed context-driven weighting approach to other data selection strategies (including window-based and time-based-weighting approaches) under varying concept recurrency settings.

Objective 5. Test the behaviour of the proposed context-driven weighting approach in a static environment.

Further details regarding the experimental setup and results are presented next.

5.4.2 Factors Affecting Context Utilisation

In this section, we conduct a number of experiments to test the behaviour of the proposed approach under the following conditions: varying contextual importance, varying context dimensionality, and the presence of noise. To achieve this, we generate 3000 examples and 5 concepts according to the data generation framework introduced in Chapter 3, following concept sequence: 1-2-3-4-5-1-2-3-4-5, with 300 examples per concept. More details are provided below.

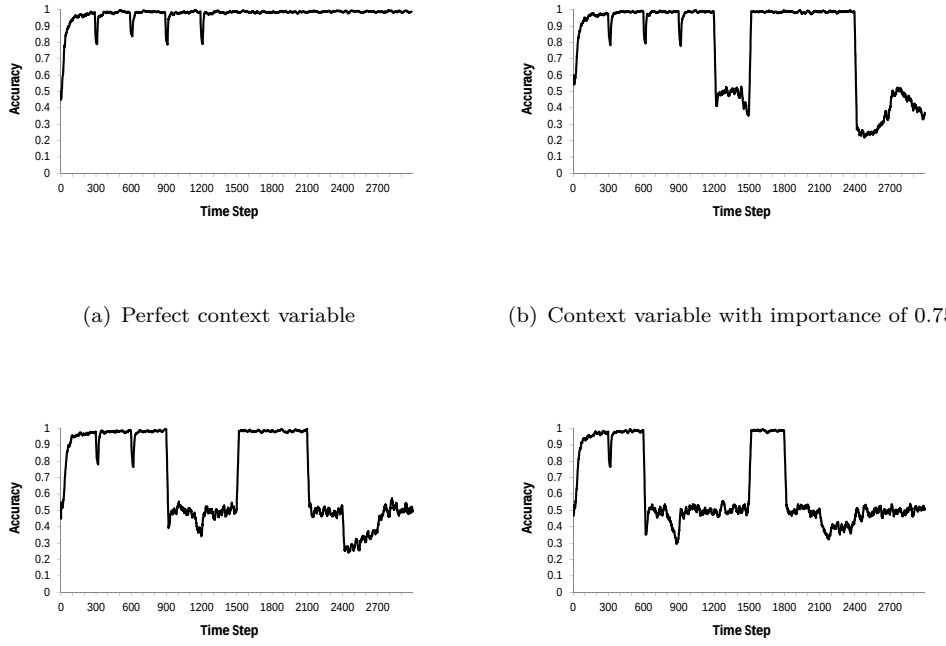
5.4.2.1 Imperfect Context

This section studies the influence of having imperfect contextual knowledge on adapting the classifier to concept changes. First, we introduce one normally distributed context variable, where its contextual importance is made to vary according to the following values (on a scale between 0 and 1): 1, 0.75, 0.5, and 0.25 (see the context generation framework in Section 3.2.1.2). The corresponding predictive accuracy of the classifier in the considered 4 cases are presented in Figure 5.1, where the base classifier adopted is Naive Bayes classifier.

The case of perfect contextual knowledge (i.e. with a context variable importance of 1) is demonstrated in Figure 5.1(a). As can be observed, having perfect contextual knowledge enables fast recovery after encountering new change points (points 300, 600, 900, and 1200), and helps avoiding performance degradation in the case of recurring concepts (starting from time step 1500). In particular, when a change occurs, only the most recent relevant examples with context similar to the current context are included for learning the predictive model (i.e. all examples belong to the same concept), which eliminates the effect of older irrelevant examples on the current prediction. Moreover, in the case of recurring concepts, all older observations with similar context are immediately utilised for prediction, without experiencing performance degradation while training the classifier on new observations.

On the other hand, in Figures 5.1(b), 5.1(c), and 5.1(d) (the case of imperfect contextual knowledge), we can see that the classifier suffers from performance degradation. In particular, in Figure 5.1(b), the classifier exhibits a deep accuracy drop at time step 1200 due to the inability of the context variable to distinguish between Concepts 4 and 5, which causes the utilisation of observations irrelevant for the current concept. In addition, when concepts re-appear, we can see the performance of the classifier deteriorates at time step 2400 (when Concept 4 re-appears), due to utilising data that belongs to both Concepts 4 and 5. Similar behaviour is observed in Figures 5.1(c) and 5.1(d). For example, in Figure 5.1(d), where the context is only able to distinguish one change point, we can see that the classifier suffers from accuracy drop starting from time step 600 (when moving from Concept 2 to 3).

Similar observations are exhibited when utilising Logistic Regression as the base classifier, which is demonstrated in Figure 5.2. For simplicity, in the rest of the chapter we depict the results with Naive Bayes being the base classifier.



(a) Perfect context variable

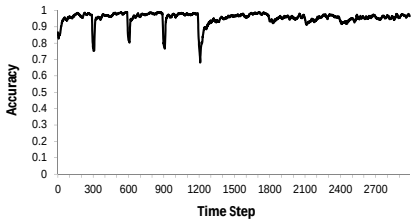
(b) Context variable with importance of 0.75

(c) Context variable with importance of 0.5

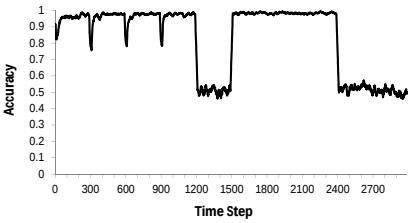
(d) Context variable with importance of 0.25

FIGURE 5.1: Performance evolution given different levels of contextual importance with Naive Bayes classifier.

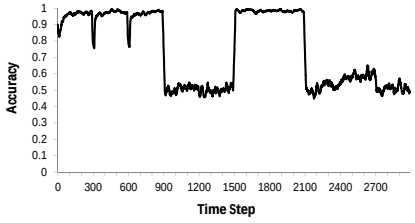
While normally utilising imperfect context results in decreased classification performance as shown above, there could be cases where a combination of imperfect context variables would actually result in an improved performance. This is the case of imperfect context variables that *complement* each other. To illustrate this, we generate 4 context variables, where each context variable has an importance of 0.25 such that the first one distinguishes between Concepts 1 and 2, the second variable distinguishes between Concepts 2 and 3, the third variable distinguishes between Concepts 3 and 4, while the last variable distinguishes between Concepts 4 and 5. The results are presented in Figure 5.3.



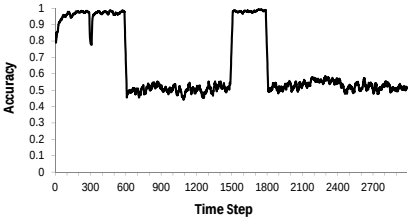
(a) Perfect context variable



(b) Context variable with importance of 0.75



(c) Context variable with importance of 0.5



(d) Context variable with importance of 0.25

FIGURE 5.2: Performance evolution given different levels of contextual importance with Logistic Regression classifier.

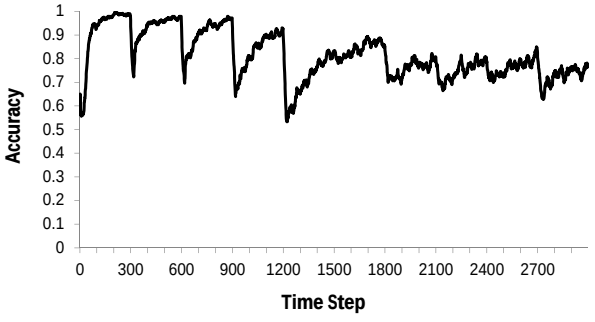


FIGURE 5.3: Complementarity among context variables.

As can be seen, although each variable individually is able to distinguish only one concept change point, utilising these variables collectively help to recognise all change points and recurring concepts, resulting in a corresponding positive effect on performance (i.e. fast recovery of the classifier in the case of new concepts and avoiding performance degradation with recurring concepts). The classifier, however, still achieves lower performance than the case of perfect contextual knowledge (Figure 5.1(a)). This is because, despite having one discriminating context variable at each change point, the other non-discriminating context variables will still carry an impact on example weights at that point, boosting the weights of irrelevant examples, which therefore have some impact on the prediction process.

5.4.2.2 High Dimensionality and Noise

In this section, our goal is to demonstrate the effect of having redundant context variables on the performance of the classification model. For this purpose, we assume one perfect context variable, and generate noise via introducing a number of imperfect context variables of varying levels of correlations with the perfect one. For this purpose, we consider two cases. In the first case, we introduce 6 context variables, where the first variable is perfectly contextual, while the other 5 ones are redundant. In the second case, we increase the number of redundant context variables from 5 to 10. Figure 5.4 reports the results. Note that, we limit the tested dimensionality to 5 and 10 variables since increasing the dimensionality beyond 10 achieve similar results but with further degradation in the classifier performance.

As can be seen, including imperfect redundant variables negatively affects the performance of the classifier. This is because, according to Equation 5.21, such imperfect redundant variables would still carry some influence on the context similarity measure,

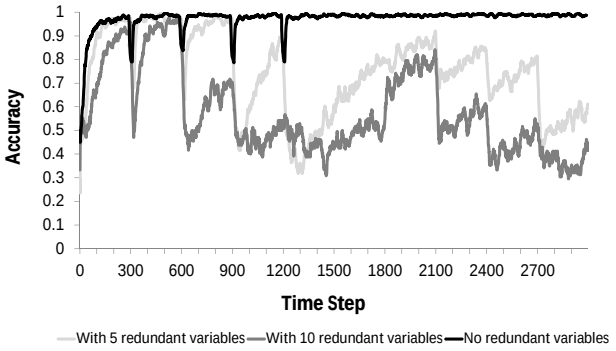


FIGURE 5.4: Effect of redundancy.

reducing the influence of the perfect context variable(s) in weighting the training examples. This negative influence of the imperfect redundant variables increases as the number of these variables increases (the case of 10 redundant variables in the figure). As a result, examples that are not relevant for the current concept will also have some impact when adapting the classifier to changes, leading to decreased performance.

The above results highlight the importance of eliminating redundancy. Redundancy elimination can be achieved by applying the proposed context selection algorithm introduced in Chapter 4 (see Section 4.3.4) prior to utilising the identified context variables in classification.

5.4.3 Evaluation of Value Equivalence Function

In this section, we compare the performance of the proposed context-driven weighting approach under the alternative variations proposed for the value equivalence function: the simple direct difference approach and the learning-based approach. For the

learning-based approach, both EAD (*EAD-based*) and KL (*KL-based*) are utilised in the analysis.

The experiments are conducted on the Stagger dataset, which exhibits abrupt changes. Note that it is not intuitive to apply the learning-based approach with gradual changes. This is because, in the case of gradual changes, the target concept is shifting slightly at each time step, and so does the associated context values. Given this semantics, the direct difference approach is more intuitive in this case.

We generate 1200 examples and 3 concepts, with each concept occurrence lasting for 200 examples following concept sequence 1-2-3-1-2-3. As context, we introduce a perfect context variable, $C_1 \in \{v_1, v_2, v_3, v_4, v_5, v_6\}$, where each value is associated with a certain concept. Specifically, $C_1 = v_1 \vee v_2$ under Concept 1, $C_1 = v_3 \vee v_4$ under Concept 2, and $C_1 = v_5 \vee v_6$ under Concept 3. This context variable is exposed to 5% noise (i.e. has a 5% probability to take on values not associated with the current concept). In all experiments, the equivalence among the values of the context variable is first learned separately in an offline mode (the first 600 examples), and then incorporated during classification into the adaptation process (the following 600 examples).

First, we report the identified value equivalence (veq) between the values of the context variable utilising the learning-based approach (see Tables 5.1 and 5.2). All results are normalised on a scale between 0 and 1. Ideally, the approach should recognise the equivalence between values v_1 and v_2, values v_3 and v_4, and values v_5 and v_6.

We can see that *EAD-based* is able to better recognise equivalence among values compared to *KL-based* approach. For example, with *KL-based* approach (Table 5.2), the equivalence degree between values $v1$ and $v2$ that appear under the same concept is identified as 0.52, which is similar to the equivalence degree identified between values $v1$ and $v4$ (0.51) that belong to different concepts. On the other hand, with *EAD-based*

TABLE 5.1: Identified equivalence among context values with *EAD-based* realisation of the value equivalence function.

Values	v_1	v_2	v_3	v_4	v_5	v_6
v_1	1	0.56	0.24	0.18	0.01	0.10
v_2	0.56	1	0.21	0.13	0	0.08
v_3	0.24	0.21	1	0.65	0.25	0.24
v_4	0.18	0.13	0.65	1	0.24	0.26
v_5	0.01	0	0.25	0.24	1	0.49
v_6	0.10	0.08	0.24	0.26	0.49	1

TABLE 5.2: Identified equivalence among context values with *KL-based* realisation of the value equivalence function.

Values	v_1	v_2	v_3	v_4	v_5	v_6
v_1	1	0.52	0.40	0.51	0.34	0.18
v_2	0.52	1	0.56	0.45	0.41	0.06
v_3	0.40	0.56	1	0.59	0.51	0.20
v_4	0.51	0.45	0.59	1	0.52	0
v_5	0.34	0.47	0.51	0.52	1	0.27
v_6	0.18	0.06	0.19	0	0.27	1

approach (Table 5.1), the equivalence degree between values $v1$ and $v2$ is identified as 0.56 compared to 0.18 between values $v1$ and $v4$.

We further compare the performance of *EAD-based* and *KL-based* approaches in terms of the accuracy of the classifier achieved. As can be seen from Figure 5.5, the classifier with *EAD-based* approach outperforms the classifier with *KL-based* approach. This is because, as demonstrated by the equivalence degree tables above, the latter has lower ability in differentiating equivalent from non-equivalent context values. Thus it suffers from accuracy drops during periods of concept stability due to treating no-drift cases (a change in a context value to another equivalent value) as a concept drift.

Based on the above, in the following experiments, we utilise *EAD-based* approach for learning the equivalence degrees among the values of context variables.

Now, we compare the *Simple Difference* and the *EAD-based* approaches. Figure 5.6

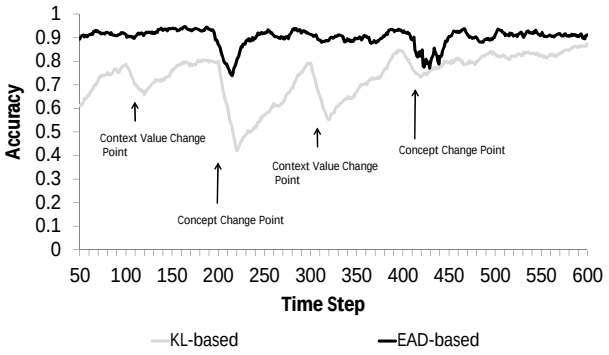


FIGURE 5.5: Comparison of *EAD-based* and *KL-based* realisations of the learning-based value equivalence function: Stagger dataset.

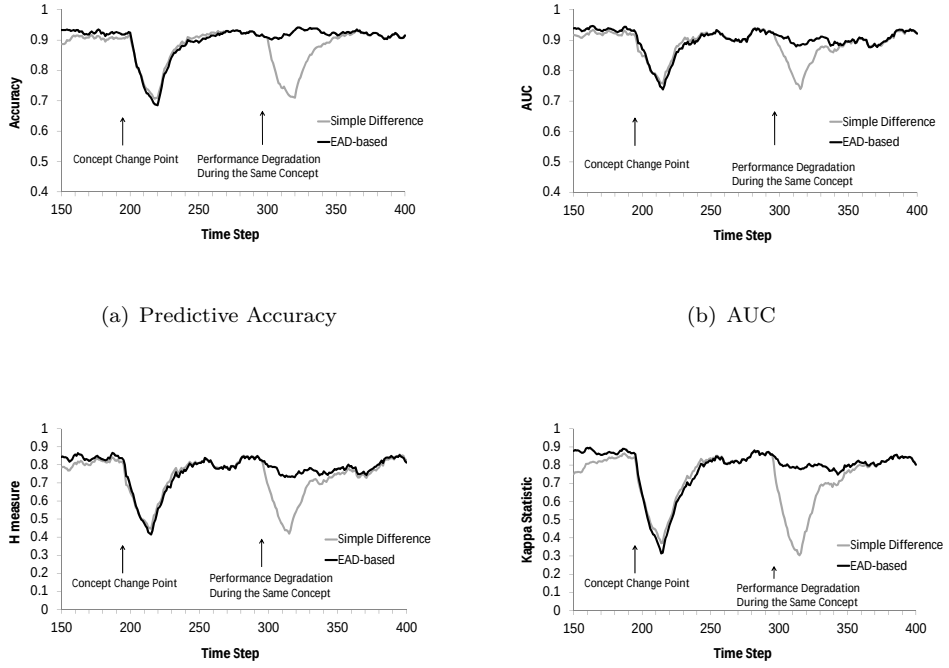
reports the corresponding results. The advantage of using *EAD-based* over *Simple Difference* is evident, with the former outperforming the latter in terms of AC (accuracy) and other performance metrics. To further analyse the results, Figure 5.7 studies the accuracy evolution over time of both approaches after a new concept (Concept 2 in the figures) is encountered. As can be seen, the *Simple Difference* suffers from an accuracy drop during the period of concept stability (in particular, after time step 300, when the context variable switches from value v_3 to v_4 while still being under Concept 2). This is because the appearance of a new value for the context variable is interpreted as an indication of a new concept, thus neglecting relevant examples prior to this value change (i.e. examples under context value v_3). In contrast, the *EAD-based* approach recognises the new value of the context variable as an equivalent one (belonging to the same concept), avoiding such degradation in the predictive accuracy.

5.4.4 Comparative Analysis Against Other Approaches

In this section, we compare the following classification strategies.

Value Equivalence	AC	KS	AUC	H
Simple Difference	0.86 (0.01)	0.71 (0.03)	0.91 (0.01)	0.69 (0.02)
EAD-based	0.89 (0.01)	0.77 (0.03)	0.92 (0.01)	0.73 (0.02)

FIGURE 5.6: Comparison of different realisations of the value equivalence function (standard deviations are indicated in parentheses)



(a) Predictive Accuracy

(b) AUC

(c) H measure

(d) Kappa Statistic

FIGURE 5.7: Comparison of different realisations of the value equivalence function over time: Stagger dataset.

Standard Incremental. The classification model is re-estimated based on all the examples observed so far, assigning equal weights to all examples.

Context as Input. The classification model is similar to the *Standard Incremental*, but with considering relevant context variables as additional input variables.

Sliding Window. The classification model is re-estimated based on a fixed window of the latest observed examples (assigning equal weights to all examples).

Dynamic Window. The classification model is re-estimated based on a dynamic window of the latest observed examples. Three alternative realisations of the dynamic window strategy are considered in the analysis: the change detection approach proposed by [6] (which is the default Dynamic Window approach in our analysis), the statistical Page-Hinkley test [81], and the drift detection method (referred to as EDDM) proposed by [7].

Time-based Weighting. The classification model is re-estimated utilising a time-based weighting scheme inspired by the approach proposed in [12], such that the weight of each example decreases with time.

Context-based Weighting. The classification model is re-estimated utilising the proposed context-based weighting scheme (i.e. the proposed context-driven classifier of Section 5.3).

Figure 5.8 reports the corresponding results for Stagger (abrupt changes) and Hyperplane (gradual changes) datasets, with concept sequence settings being the same as in the previous section.

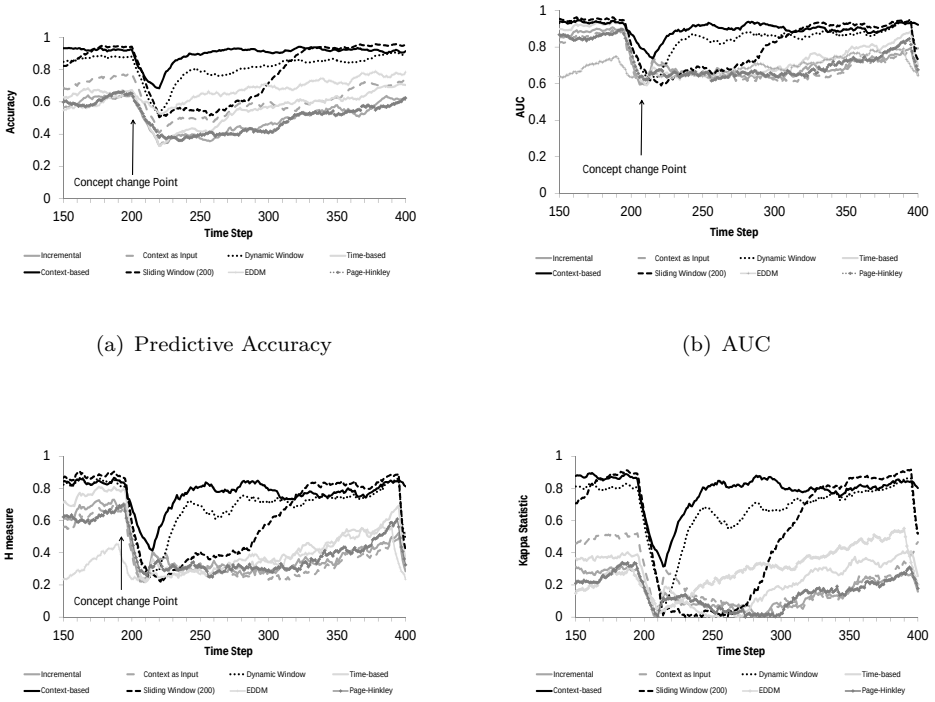
As can be seen, in the case of abrupt changes (Stagger dataset), *Context-based Weighting* outperforms all the other considered strategies (including *Context as Input*). Further analysis of this case is provided in Figures 5.9 and 5.10, where we depict the

Classification Strategy	Stagger				Hyperplane			
	AC	KS	AUC	H	AC	KS	AUC	H
Standard Incremental	0.57 (0.02)	0.15 (0.03)	0.64 (0.02)	0.09 (0.02)	0.79 (0.03)	0.65 (0.05)	0.90 (0.02)	0.56 (0.05)
Context as Input	0.70 (0.02)	0.41 (0.04)	0.75 (0.02)	0.26 (0.04)	0.82 (0.02)	0.64 (0.05)	0.89 (0.02)	0.54 (0.05)
Sliding Window	0.68 (0.01)	0.37 (0.02)	0.87 (0.01)	0.24 (0.14)	0.84 (0.01)	0.67 (0.03)	0.89 (0.01)	0.57 (0.03)
Dynamic Window	0.81 (0.09)	0.62 (0.19)	0.86 (0.07)	0.54 (0.05)	0.82 (0.02)	0.64 (0.04)	0.89 (0.02)	0.53 (0.04)
Time-based Weighting	0.59 (0.01)	0.19 (0.03)	0.67 (0.02)	0.12 (0.02)	0.83 (0.02)	0.66 (0.04)	0.90 (0.02)	0.56 (0.04)
Context-based Weighting	0.91 (0.01)	0.81 (0.01)	0.92 (0.01)	0.73 (0.02)	0.82 (0.02)	0.64 (0.04)	0.89 (0.02)	0.54 (0.05)

FIGURE 5.8: Comparison of different classifiers for Stagger and Hyperplane datasets (standard deviations are indicated in parentheses)

evolution of predictive accuracy over time, distinguishing the cases of new and recurring concepts. In particular, when a new concept is encountered (Figure 5.9), all the considered strategies suffer from performance degradation. However, *Context-based Weighting* achieves the fastest recovery. This is because, as mentioned previously, the utilisation of contextual evidence enables capturing only relevant observations for learning the new concept. In particular, the change in the value of the considered context variable signals the classifier to forget older observations, and to consider only the most recent data observations that belong to the new concept, with context similar to the current context. In contrast, time-based strategies suffer from slower reaction to changes since the effect of older irrelevant observations takes longer to be forgotten. Furthermore, when encountering a recurring concept (Figure 5.10), and unlike other strategies, *Context-based Weighting* always maintains high accuracy without experiencing performance degradation after a change point, due to its ability to utilise old (but relevant again) data.

In the case of gradual changes (Hyperplane dataset in Figure 5.8), the examples become less relevant gradually (as opposed to abrupt changes where the relevance of examples drastically decreases after the change point). Hence, recency based approaches achieve good prediction accuracy via gradually forgetting outdated data. Our proposed



(a) Predictive Accuracy

(b) AUC

(c) H measure

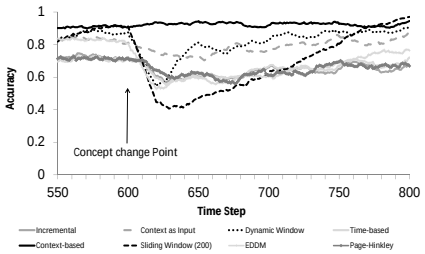
(d) Kappa Statistic

FIGURE 5.9: Performance Evolution of different classifiers for Stagger dataset: encountering new concept.

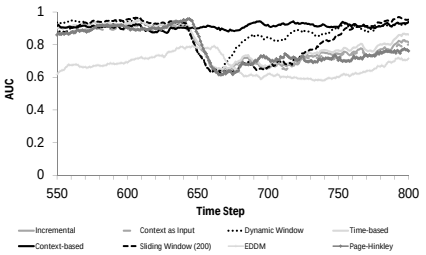
context-driven approach also achieves good results, approximating the performance of recency based approaches.

The advantage of the proposed approach is also demonstrated with Gauss and Sine1 datasets, with their results being reported in Figure 5.11. As in the previous experiments, the proposed context-based weighting outperforms all the other approaches.

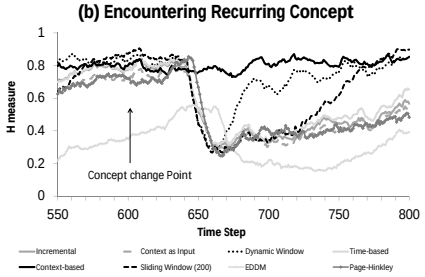
Similar observations hold for the real-world datasets, with the results being reported in Table 5.3. For these datasets, we utilise the McNemar's test [5] to evaluate the statistical significance of the performance differences obtained.



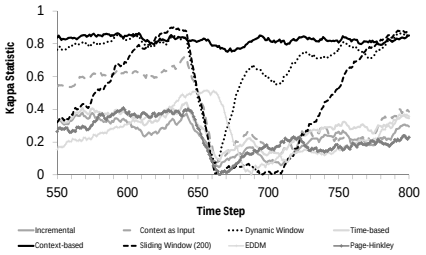
(a) Predictive Accuracy



(b) AUC

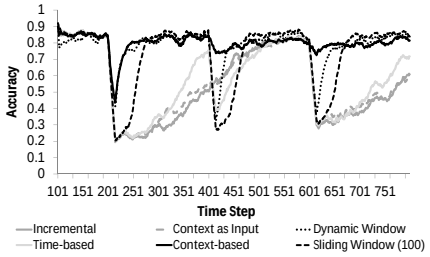


(c) H measure

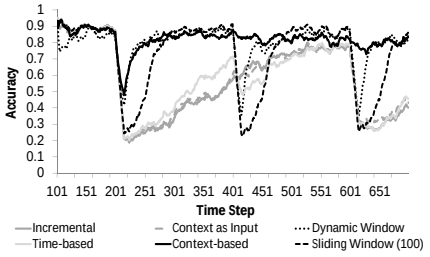


(d) Kappa Statistic

FIGURE 5.10: Performance evolution of different classifiers for Stagger dataset: encountering recurring concept.



(a) Gauss



(b) Sine1

FIGURE 5.11: Performance Evolution of different classifiers for Gauss and Sine1 datasets.

TABLE 5.3: Comparison of different classifiers for Elist and Elec datasets

Classification Strategy	**Elist**		**Elec**	
	AC	**KS**	**AC**	**KS**
Expanding Window	0.62*	0.23	0.63*	0.26
Context as Input	0.69*	0.38	0.64*	0.28
Sliding Window	0.62*	0.24	0.64*	0.26
Dynamic Window	0.70*	0.39	0.66	0.29
Time-based Weighting	0.62*	0.24	0.64*	0.27
Context-based Weighting	0.81	0.61	0.67	0.34

* Significant difference according to McNemar Test (0.05 significance level)

5.4.5 Static Environment

In this section, we generate a static artificial dataset by fixing the mean of the data generating distribution to be the same at each time step. In such settings (i.e. in the case of no concept changes), all previously observed examples remain relevant. In Figure 5.12, we compare the behaviour of the proposed approach against the Standard Incremental (expanding window) strategy. The results show that both approaches behave similarly, demonstrating the validity of our approach in static domains.

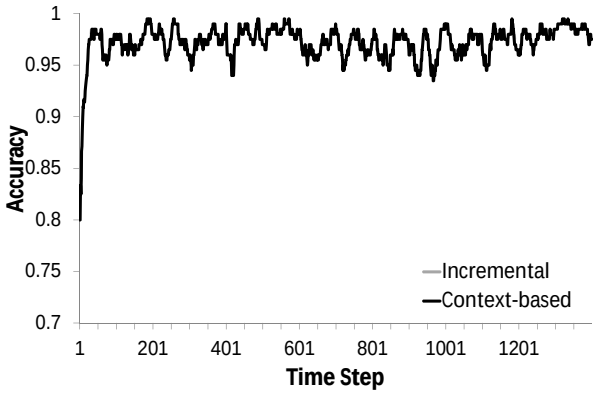


FIGURE 5.12: Accuracy evolution of the context-based weighting in a static dataset.

5.5 Conclusion

The chapter presented a context-based weighting approach for adapting classification models to concept changes, where each example is weighted according to the degree of similarity with the current context. When assessing the degree of similarity between two contextual instances, context variables with higher imprtance carry a higher impact. Moreover, equivalence between two values of a context variable is determined by either assessing the direct numerical difference between these values, or by alternatively learning their conceptual similarity (i.e. learning the degree of similarity between their underlying concepts). The latter is achieved in an offline mode, assuming the existence of a representative training sample. The experimental analysis demonstrated that the learning-based approach achieves advantages over the direct difference one in settings with abrupt concept changes where the numerical difference between the context values is not necessarily informative.

The experimental analysis also demonstrated the advantage of the proposed approach compared to other existing approaches. The benefit of external context utilisation (as opposed to incorporating context as an input) and the ability of the approach to perform in a static environment are also illustrated.

The presented weighting approach, however, requires iterating over the past data to evaluate the degree of similarity between the context under which each example is collected and the current context. As the size of the historical data increases, especially in the case of data streams which have infinite length, it becomes impractical and computationally expensive to store all the historical data and do multiple passes over this data for the purpose of weight re-evaluation. Therefore, an interesting extension would be to adapt the weights of training examples without the need to store or revisit the contextual information of the examples each time, by applying some form of

incremental learning.

The proposed weighting-based approach assumes a continuous drift, re-weighting the data examples every time the classifier is updated, without identifying if the drift actually occurred. In the next chapter, an alternative context-based adaptation approach based on drift detection is presented.

Finally, in the thesis, we assume that the number of classes is fixed. Yet, there could be situations where new classes might emerge over time, a problem know in the literature as concept evolution. The currently assumed offline mode for learning the importance of context variables will not be suitable in this case, as potential classes cannot be known in advance. Hence, to tackle this problem, the importance of the context variables has to be learned incrementally. Moreover, to keep the example weighting approach applicable in this case, an additional variable may need to be added such that it changes values every time a novel class is detected. The detection of new classes can be conducted using proposed techniques in the literature [121], [120]. However, this is out of the scope of this thesis.

Chapter 6

Context Exploitation Model - Drift Detection

6.1 Introduction

Drift detection is an important step towards achieving more effective concept drift handling, and the reason is twofold. First, it facilitates choosing the right data instances for adjusting the classification model (e.g. data preceding the detection point may no longer be relevant). Second, it eliminates the need for the costly revisiting of past data samples to re-assess their relevance every time the classifier is updated (as seen in the previous chapter). In particular, the classification model built from data following the drift point remains relevant, and any new data can be simply incorporated into this model to increase its accuracy potentially applying efficient incremental update rules.

Existing approaches to drift detection mainly focus on monitoring the performance accuracy of the classification model to identify the point where the drift has occurred.

This is a costly process (in terms of performance degradation) since it usually requires a sufficient number of misclassified instances before drift can be confirmed.

In this chapter, we propose an adaptive learning model equipped with a drift detection strategy that relies on monitoring changes in relevant contextual variables. Such utilisation of contextual information will enable a faster (less costly in terms of performance degradation) and more reliable method to recognise drift. Moreover, to account for the possibility of concept recurrence during the learning process, a context-based recurrency detection strategy is proposed. In particular, the proposed learner maintains a set of previously-learned prediction models (i.e. it is multi-model), each corresponding to an identified different concept. When a previously-encountered concept reoccurs, the respective stored prediction model is reused to make future predictions, facilitating a faster adaptation to drift (as opposed to re-learning the concept from new examples), and consequently enables improving the predictive accuracy. Two realisations of such an adaptive multi-model learner are proposed: a purely-contextual learner and a hybrid learner that additionally incorporates a performance degradation monitor.

The remaining of this chapter is organised as follows. The purely-contextual multi-model learner is presented in Section 6.2. In Section 6.3, the hybrid learning model is introduced. Evaluation setup and results are reported in Section 6.4, while Section 6.5 concludes the chapter.

6.2 Purely-Contextual Multi-Model Learner

In the first realisation of the multi-model learner, drift detection and the recognition of concept recurrence (with the corresponding classifier selection) are only based on the utilisation of contextual information. That is, change detection is based on monitoring any deviations in the values of the context variables. Similarly, the value(s) of the

context variable(s) are utilised for recognising the recurrence of a previously observed concept, and the selection of the appropriate classifier (in terms of context similarity). More details of this learner are provided below.

6.2.1 Learner Configuration

The configuration of the multi-model learner at a particular moment is a tuple:

$$(\mathbb{M}, m^a, ctxcnd(m))$$

where the components are as follows.

$\mathbb{M}$ is a library of classification models, containing the set of previously learned models up to the current time point. Each model $m \in \mathbb{M}$ characterises a different concept (i.e. a different function between the variable to be predicted and the respective input data). Note that the approach is not restricted to any specific type of classification models; $m^a \in \mathbb{M}$ is the currently active classification model, i.e. the one utilised to predict the outcome of future unlabelled examples; $ctxcnd(m)$ is the context condition function, assigning to each model $m \in \mathbb{M}$ the contextual condition under which the model is applicable. In particular, given the relevant context attributes $(C_1, C_2, ..., C_r)$, each condition $ctxcnd(m)$ is a conjunctive clause of the form:

$$ctxcnd(m) = c_1 \ \wedge \ c_2 \ \wedge \ ... \ \wedge \ c_r$$

where values $(c_1, c_2, ..., c_r)$ correspond to a particular restriction on the values of context attributes $(C_1, C_2, ..., C_r)$. For example, given two context attributes $(Season, Day)$, a possible condition is $Winter \wedge Saturday$, where $Winter$ and $Saturday$ are the values restricting $Season$ and Day, respectively.

Drift	**Context Variable C**
Concept 1	$v_1 \vee v_2$
Concept 2	v_3
Concept 3	v_4

FIGURE 6.1: Perfect context knowledge: individual concept distinction.

Drift	**Context Variable C_1**	**Context Variable C_2**
Concept 1	v_1	v_3
Concept 2	v_1	v_4
Concept 3	v_2	v_4

FIGURE 6.2: Perfect context knowledge: collective concept distinction.

Drift	**Context Variable C**
Concept 1	v_1
Concept 2	v_1
Concept 3	v_2

FIGURE 6.3: Imperfect context knowledge.

6.2.2 Drift Detection

The purely-contextual realisation of the multi-model learner mainly relies on available contextual knowledge in order to detect the occurrence of a concept drift. It monitors changes in the values of relevant context attributes and exploits the relation between these changes and concept changes.

In order to be able to identify changes in the values of context attributes, we first define the following equivalence relation between context values.

Definition 1. *Value Equivalence* $\equiv^v$. Given two values v_1 and v_2 of context attribute C_i, these values are considered equivalent, i.e. $v_1 \equiv^v v_2$, if: $veq(C_i, v_1, v_2) \geq eqThresh$. Here, veq is the learning-based value equivalence function proposed in Chapter 5, and $eqThresh$ is the value equivalence threshold.

Now, the *drift detection principle* assumed by the purely-contextual multi-model learner can be defined as follows. Given the relevant context attributes $(C_1, C_2, ..., C_r)$, there is a concept drift at time step t **if and only if** there is a change in the value of at least one of these attributes C_i. That is:

$$\text{concept drift at time step } t \quad \Leftrightarrow \quad \exists C_i,\ \neg[c_i(t) \equiv^v c_i(t-1)] \tag{6.1}$$

where $c_i(t)$ is the value of context attribute C_i at time step t. Note that, in order to account for random noisy value occurrences, such a change in the value of context attribute C_i is confirmed only if it persists for at least $chThresh$ time steps.

The intuition behind the sufficiency condition (i.e. that there is a concept drift **if** there is a change in a context value) can be justified by the semantics of the context value equivalence function (upon which the equivalence, and thus the change, among context values is determined). In particular, and as explained in Chapter 5, the degree of equivalence between values v_1 and v_2 of context attribute C_i, $veq(C_i, v_1, v_2)$, corresponds to the similarity between their underlying distributions $P(Y, X|v_1)$ and $P(Y, X|v_2)$. That is, the higher the equivalence degree, the higher the chance that v_1 and v_2 appear under similar concepts, whilst the lower the equivalence degree, the higher the chance that v_1 and v_2 appear under different concepts. This justifies the signalling of a drift when context attribute C_i switches between values v_1 and v_2 and these values are non-equivalent (i.e. the case where $veq(C_i, v_1, v_2)$ is low). Clearly, the value equivalence threshold $eqThresh$, which governs the assessment of whether two context values are distinct, plays an important role, and its effect is studied in Section 6.4.1.

On the other hand, the necessity condition (i.e. that there is a concept drift **only if** there is a change in a context value) requires the available contextual attributes to perfectly distinguish among all occurring concepts, either individually (e.g. see Figure 6.1) or collectively (e.g. see Figure 6.2). Such perfect contextual knowledge, however, may

not be available in reality, resulting in cases where a drift may occur despite no corresponding changes in context values (e.g. see Figure 6.3), and thus the purely-contextual learner would fail to identify such a drift. The hybrid approach (see Section 6.3) attempts to account for such cases.

6.2.3 Concept Recurrency Detection

Similarly to drift detection, the recognition of a recurrent concept is also purely guided by contextual knowledge, associating similar context values with similar concepts. Let's first define the following equivalence relation between two context conditions.

Definition 2. *Condition Equivalence* $\equiv^c$. Consider two context conditions, $cnd^1 = {c_1}^1 \wedge {c_2}^1 ... \wedge {c_r}^1$, and $cnd^2 = {c_1}^2 \wedge {c_2}^2 ... \wedge {c_r}^2$, restricting the values of relevant context attributes $(C_1, C_2, ..., C_r)$. Conditions cnd^1 and cnd^2 are considered equivalent, i.e. $cnd^1 \equiv^c cnd^2$, if: $\forall i \in \{1, .., r\}, \quad {c_i}^1 \equiv^v {c_i}^2$.

Based on this, the *recurrency detection principle* assumed by the purely-contextual multi-model learner can be defined as follows. Given a drift at time step t, the upcoming concept (i.e. the concept staring from time step t) is recurrent **if and only if** its respective context values are subsumed by the context condition of a previously-learned classification model. That is:

$$\text{the concept from time step } t \text{ is recurrent} \quad \Leftrightarrow$$
$$\exists m \in \mathbb{M}, \quad ctxcnd(m) \;\equiv^c\; [c_1(t) \wedge c_2(t) ... \wedge c_r(t)] \quad (6.2)$$

where $[c_1(t) \wedge c_2(t) ... \wedge c_r(t)]$ is the context condition corresponding to the conjunction of context values observed for the relevant context attributes $(C_1, C_2, ..., C_r)$ at time step t.

The same discussion provided above regarding drift detection also applies here (with respect to the sufficiency and necessity conditions of the recurrency detection principle). In other words, while the absence of an existing classifier with an equivalent context condition negates concept recurrence (driven by the semantics of context value equivalence), the presence of such a classifier does not necessarily confirm the recurrence if contextual knowledge is imperfect, leading to cases with incorrectly identified recurrency by the purely-contextual learner. For example, consider the imperfect contextual knowledge of Figure 6.3, and assume the following order of concept appearance: concept 1 - concept 3 - concept 2. The purely-contextual learner would *incorrectly* identify concept 2 as recurrent, due to the existence of a library classifier with a similar context condition (the one previously built for concept 1). Again such a case is handled by the hybrid approach (see Section 6.3).

Moreover, given such a recurrency detection principle of the purely-contextual multi-model learner, the following property can be concluded.

Property 1. There can be at most one classifier in the model library per context condition at any particular moment. That is:

$$\forall m_1, m_2 \in \mathbb{M}, \quad \neg[ctxcnd(m_1) \equiv^c ctxcnd(m_2)]$$

Proof. The proof is straightforward, and can be provided by contradiction. Assume the property does not hold, i.e.:

$$\exists m_1, m_2 \in \mathbb{M}, \quad [ctxcnd(m_1) \equiv^c ctxcnd(m_2)] \tag{6.3}$$

Suppose m_1 was built before m_2. From above assumption and the recurrency detection principle, the concept for which m_2 is built would be identified as recurrent (due to the existence of m_1 with an equivalent context condition). This results in the reuse of m_1

for such a concept, without building a new classifier, i.e. $m_2 \notin \mathbb{M}$, which contradicts the assumption in Equation 6.3. Thus, this assumption cannot be true, which proves our property. □

6.2.4 Overall Learning Algorithm

The main learning steps of this approach are summarised below (see Algorithm 6).

1. At each time step, compare the current context (the context of the example to be classified), $ctx(t) = [c_1(t) \wedge c_2(t) ... \wedge c_r(t)]$, with the context of the currently active classifier, $ctxcnd(m^a)$;

2. If $ctx(t)$ is not equivalent to $ctxcnd(m^a)$ for $chThresh$ time steps, a drift is detected and the following two cases are distinguished:

   - *Case 1*: The concept following the drift is new (not recurrent) according to Equation 6.2. In this case, a new classification model, m^n, is instantiated and added to the library of classification models, i.e. $\mathbb{M} \leftarrow \mathbb{M} \cup \{m^n\}$. This classifier becomes the currently active classifier, i.e. $m^a \leftarrow m^n$, and is trained on the latest examples (the examples following the change point). Its context condition is set to $ctx(t)$, i.e. $ctxcnd(m^n) = ctx(t)$.

   - *Case 2*: The concept following the drift is recurrent according to Equation 6.2, with m^{eq} being the existing equivalent classification model found in the library. In this case, the currently active classification model is simply replaced by the existing found model, i.e. $m^a \leftarrow m^{eq}$.

Algorithm 6 ACTX-Simple

Require: Data Stream: D, Context Stream: CTX , Active Model: m^a, Model Library: $\mathbb{M}$, Persistence Threshold: $chThresh$

Initialize: classifier m^n, $counter \leftarrow 0$
for all $x(i) \in D, \quad c(i) \in CTX$ **do**
  if $\neg[ctx(i) \equiv^c ctxcnd(m^a)]$ **then**
    $counter \leftarrow counter + 1$
  else
    $counter \leftarrow 0$
  if $counter = 0$ **then**
    reset classifier m^n to initial settings
    Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
    Update m^a with $(x(i), y(i))$
  else
    if $counter < chThresh$ **then**
      Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
      Update m^n with $(x(i), y(i))$
    else
      $newconcept \leftarrow$ True
      for all $m \in \mathbb{M}$ **do**
        if $(ctx(i) \equiv^c ctxcnd(m))$ **then**
          $m^a \leftarrow m$
          $newconcept \leftarrow$ False
      if $newconcept$ **then**
        $\mathbb{M} \leftarrow \mathbb{M} \cup \{m^n\}$
        $ctxcnd(m^n) = ctx(i)$
        $m^a \leftarrow m^n$
      $counter \leftarrow 0$
      reset classifier m^n to initial settings
      Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
      Update m^a with $(x(i), y(i))$

6.3 Hybrid Multi-Model Learner

6.3.1 Learner Configuration

The configuration of the hybrid multi-model learner at a particular moment is a tuple:

$$(\mathbb{M}, m^a, ctxcnd, prfMntr)$$

where the first three components are similar to those of the purely-contextual multi-model learner, while $prfMntr$ is an *additional* component to monitor the performance of the learner in terms of accurately classifying incoming examples. In particular, $prfMntr(t)$ returns an indication of the error rate of the currently active classification model m^a at time step t, thus providing additional evidence for detecting the occurrence of a drift besides contextual knowledge.

6.3.2 Drift Detection

The hybrid multi-model learner combines context-based drift detection with accuracy-based drift detection to boost the performance of the former in cases where perfect contextual knowledge may not be available. The main idea behind accuracy-based drift detection, a well-known approach in the literature [92], is that if the distribution of the incoming examples is stationary, the probability of making an error will decrease or at least stabilise as more examples become available. A significant increase in this probability indicates that the distribution generating the examples has changed, signalling a drift.

Based on this, the *drift detection principle* assumed by the hybrid multi-model learner can be defined as follows. Given relevant context attributes $(C_1, C_2, ..., C_r)$, there is a concept drift at time step t, **if and only if** *one* of the following is satisfied:

- there is a change in the value of at least one of these context attributes C_i, at time step t (and this change persists for at least $chThresh$ time steps); or
- there is a degradation in the performance of the currently active classification model, signalled by a warning level $wrnThresh$ at time step t, and confirmed by

a further drift level $drfThresh$ at some point in future. That is:

$$(prfMntr(t) \geq wrnThresh) \ \wedge \ (\exists t_f > t, \ prfMntr(t_f) \geq drfThresh) \quad (6.4)$$

While the first point matches the detection principle of the purely-contextual learner, i.e. Equation 6.1, the second one is specific to performance monitoring and is introduced to boost the necessity condition of this equation (to account for cases with imperfect contextual knowledge).

A possible instantiation of $prfMntr$, $wrnThresh$, and $drfThresh$ is to adopt the error rate approach proposed by Gama et al. [6], which is also utilised in Gomes et al.'s learning system [28] against which we conduct our experimental analysis (yet alternative instantiations are also possible). In particular, it is assumed that the error of each incoming example, $u(i) = (x(i), y(i))$, represents a random variable from Bernoulli trials, and that the number of misclassified examples, denoted as E, follows a Binomial distribution. Given this, the probability of making an error, denoted as p_i, and the associated standard deviation, denoted as s_i, are computed incrementally for each incoming example according to the following formulas: $p_i = \frac{E}{i}, s_i = \sqrt{\frac{p_i(1-p_i)}{i}}$. The learner's performance at example i, $prfMntr(i)$, corresponds to $p_i + s_i$, while $wrnThresh$ and $drfThresh$ correspond to $p_{min} + 2 * s_{min}$, and $p_{min} + 3 * s_{min}$, respectively. Here, p_{min} and s_{min} are the minimum error probability and minimum standard deviation, respectively, among the examples observed so far, obtained in the sequence of calculating p_i and s_i for the incoming examples.

6.3.3 Concept Recurrency Detection

Given the drift detection principle of the hybrid learner, and the possibility of imperfect contextual knowledge (e.g. see Figure 6.3), Property 1 should no longer hold. That is,

there may exist two classifications models m_1 and m_2 corresponding to distinct concepts such that $ctxcnd(m_1) \equiv^c ctxcnd(m_2)$ (the case where a drift occurs despite no change in contextual knowledge, and is detected utilising degradation of performance $prfMntr$). Hence, although the absence of an existing library classifier with a context condition similar to that of a detected concept still confirms that the concept is new (driven by the semantics of context value equivalence), the presence of such a classifier does not necessarily confirm that the concept is recurrent, requiring an additional similarity measure to confirm recurrency.

Let's define the following equivalence measure between two classification models.

Definition 3. *Model Equivalence* $\equiv^m$. Given two classification models m_1 and m_2, these models are considered equivalent, i.e. $m_1 \equiv^m m_2$, if: $similarityDegree(m_1, m_2) \geq mdlThresh$. Here, $similarityDegree$ is the degree of similarity between models m_1 and m_2, and $mdlThresh$ is the model equivalence threshold.

A possible way to compute the similarity degree between two classification models is to utilise the *conceptual equivalence* measure originally proposed by Yang et al. [57] (and also utilised by Gomes et al. [28]). In particular, the conceptual equivalence between two models (i.e. whether or not they belong to the same concept) is determined according to their degree of agreement in classifying the data examples. That is, given two classification models m_1 and m_2, and a sample of examples $W_n = \{x(i)\}_1^n$ of size n, the conceptual equivalence between m_1 and m_2 (and hence $similarityDegree(m_1, m_2)$) corresponds to: $\frac{\sum_{x(i) \in W_n} score(x(i), m_1, m_2)}{n}$. Here, $score(x(i), m_1, m_2)$ in the similarity score between the models per each example, and is set to 1 if both models predict the same class label for example $x(i)$, and is set to -1 otherwise. The higher the value of conceptual equivalence, the higher the degree of concept similarity between the models.

Based on this, the *recurrency detection principle* assumed by the hybrid multi-model learner can be defined as follows. Consider a drift at time step t, and a new classification

model m^n, built from $stblExmNum$ examples arriving after the change point t (where $stblExmNum$ is the number of examples required to stabilise the classification model). The concept staring from time step t is recurrent **if and only if** its respective context values are subsumed by the context condition of a previously-learned classification model, and this model is equivalent to m^n, i.e.

$$\exists m \in \mathbb{M}, \quad (ctxcnd(m) \equiv^c ctx(t)) \quad \wedge \quad (m \equiv^m m^n) \tag{6.5}$$

where $ctx(t)$ is the context condition corresponding to the conjunction of context values observed for the relevant context attributes $(C_1, C_2, ..., C_r)$ at time step t.

6.3.4 Overall Learning Algorithm

In what follows, we summarise the main learning steps of the hybrid approach (see Algorithm 7).

1. At each time step, compare the current context, $ctx(t)$, with the context of the currently active classifier, $ctxcnd(m^a)$;
2. Compute the current error rate of the currently active classifier, $prfMntr(t)$;
3. Signal a *warning* level if at least *one* of the following is satisfied:
   - The current context $ctx(t)$ is not equivalent to $ctxcnd(m^a)$ for less than $chThresh$ time steps.
   - The error rate of the currently active classifier is above the warning level threshold but less than the drift level threshold, i.e.

$$wrnThresh \leq prfMntr(t) < drfThresh$$

Algorithm 7 ACTX-Hybrid

Require: Data Stream: D, Context Stream: CTX , Active Model: m^a, Model Library: $\mathbb{M}$, Persistence Threshold: $chThresh$, Warning Level Threshold: $wrnThresh$, Drift Level Threshold: $drfThresh$, Model Equivalence Threshold: $mdlThresh$, Minimum Warning Window Size: $stblExmNum$

Initialise: Classifier m^n, $counter \leftarrow 0$, $warnwindow \leftarrow \emptyset$
for all $x(i) \in D,\ c(i) \in CTX$ **do**
  if $\neg[ctx(i) \equiv^c ctxcnd(m^a)]$ **then**
    $counter \leftarrow counter + 1$
  else
    $counter \leftarrow 0$
  if $(prfMntr(t) < wrnThresh) \wedge (counter = 0)$ **then**
    Status $\leftarrow$ *Normal*
  else
    if $(prfMntr(t) < drfThresh) \wedge (counter < chThresh)$ **then**
      Status $\leftarrow$ *Warning*
    else
      Status $\leftarrow$ *Drift*
  if Status is *Normal* **then**
    $warnwindow \leftarrow$ Empty
    reset classifier m^n to initial settings
    Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
    Update m^a with $(x(i), y(i))$
  else
    if Status is *Warning* **then**
      Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
      Adjust the warning window: $warnwindow \leftarrow warnwindow \cup (x(i), y(i))$
      Update m^n with $(x(i), y(i))$
    if Status is *Drift* **then**
      if $warnwindow < stblExmNum$ **then**
        Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
        Adjust the warning window: $warnwindow \leftarrow warnwindow \cup (x(i), y(i))$
        Update m^n with $(x(i), y(i))$
      else
        $newconcept \leftarrow$ True
        for all $m \in \mathbb{M}$ **do**
          if $(ctx(i) \equiv^c ctxcnd(m)) \wedge (similarityDegree(m, m^n) \geq mdlThresh)$ **then**
            $m^a \leftarrow m$
            $newconcept \leftarrow$ False
        if *newconcept* **then**
          $\mathbb{M} \leftarrow \mathbb{M} \cup \{m^n\}$
          $ctxcnd(m^n) = ctx(i)$
          $m^a \leftarrow m^n$
        $counter \leftarrow 0$
        $warnwindow \leftarrow$ Empty
        reset classifier m^n to initial settings
        Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
        Update m^a with $(x(i), y(i))$

4. Signal a *drift* level if at least *one* of the following is satisfied:

   - The current context $ctx(t)$ is not equivalent to $ctxcnd(m^n)$ for $chThresh$ time steps.
   - The error rate is above the drift level threshold, i.e.

$$prfMntr(t) \geq drfThresh$$

5. If the learning model is in the *warning* level, the following actions are performed:

   - Instantiate a warning window (store the examples following the potential change point in the warning window).
   - Instantiate a new classifier, m^n, based on the examples in the warning window.

6. If the learning model is in the *drift* state (triggered by context changes or performance degradation of the currently active classifier), the following two cases are distinguished:

   - *Case 1*: The concept following the drift is new according to Equation 6.5. In this case, the new classification model, m^n, instantiated during the warning state is added to the library of classification models, i.e. $\mathbb{M} \leftarrow \mathbb{M} \cup \{m^n\}$. This classifier becomes the currently active classifier, i.e. $m^a \leftarrow m^n$. Its context condition is set to $ctx(t)$, i.e. $ctxcnd(m^n) = ctx(t)$.
   - *Case 2*: The concept following the drift is recurrent according to Equation 6.5, with m^{eq} being the existing equivalent classification model found in the library. In this case, the currently active classification model is simply replaced by the existing found model, i.e. $m^a \leftarrow m^{eq}$.

In addition to the context-based hybrid multi-model learner, we introduce another adaptive learning model that is similar to the hybrid model but with no context utilisation during the learning process. That is, only performance degradation and conceptual equivalence are utilised to detect concept drift and concept recurrency, respectively. The learning steps of this model are illustrated in Algorithm 8. It is utilised in our analysis to demonstrate the benefit of context utilisation as we illustrate in Section 6.4.3.

6.4 Evaluation

In this section, we present an empirical evaluation of the proposed adaptive multi-model learner, and study the influence of various factors affecting its performance. Moreover, we compare the performance of the approach against other similar approaches existing in the literature. More details about the experimental analysis and the corresponding experimental setup are provided in the following sections.

Datasets. As in previous chapters, we base our evaluation on artificial and real-world datasets, utilising *STAGGER* and *Emailing List Dataset* (Elist), respectively. Both datasets exhibit abrupt concept drifts and recurrent concepts, relevant for the purpose of evaluating drift detection techniques. The description of these datasets is provided in Chapter 3. To account for any performance differences due to the class imbalance problem, we make the *STAGGER* dataset to have 50% positive and 50% negative examples, whilst for the real-dataset we utilise AUC and Kappa Statistic (see Chapter 3) as additional performance evaluation metrics.

Performance Measures. In the evaluation of the proposed approach, we focus on measuring (1) the predictive accuracy of the multi-model learner (as previously), and (2) the associated efficiency in memory consumption (in terms of the number of stored classifiers in library $\mathbb{M}$). Both accuracy and memory consumption are highly influenced

Algorithm 8 Adaptive Learner - No Context Utilisation

Require: Data Stream: D, Active Model: m^a, Model Library: $\mathbb{M}$, Warning Level Threshold: $wrnThresh$, Drift level Threshold: $drfThresh$, Model Equivalence Threshold: $mdlThresh$, Minimum warning window size: $stblExmNum$

Initialise: Classifier m^n, $warnwindow \leftarrow \emptyset$
for all $x(i) \in D$ **do**
  if $(prfMntr(t) < wrnThresh)$ **then**
    Status $\leftarrow$ *Normal*
  else
    if $(prfMntr(t) < drfThresh)$ **then**
      Status $\leftarrow$ *Warning*
    else
      Status $\leftarrow$ *Drift*
  if Status is *Normal* **then**
    $warnwindow \leftarrow$ Empty
    reset classifier m^n to initial settings
    Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
    Update m^a with $(x(i), y(i))$
  else
    if Status is *Warning* **then**
      Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
      Adjust the warning window: $warnwindow \leftarrow warnwindow \cup (x(i), y(i))$
      Update m^n with $(x(i), y(i))$
    if Status is *Drift* **then**
      if $warnwindow < stblExmNum$ **then**
        Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
        Adjust the warning window: $warnwindow \leftarrow warnwindow \cup (x(i), y(i))$
        Update m^n with $(x(i), y(i))$
      else
        newconcept $\leftarrow$ True
        for all $m \in \mathbb{M}$ **do**
          if $(similarityDegree(m, m^n) \geq mdlThresh)$ **then**
            $m^a \leftarrow m$
            newconcept $\leftarrow$ False
        if *newconcept* **then**
          $\mathbb{M} \leftarrow \mathbb{M} \cup \{m^n\}$
          $m^a \leftarrow m^n$
        $warnwindow \leftarrow$ Empty
        reset classifier m^n to initial settings
        Model prediction: $\hat{y}(i) \leftarrow m^a(x(i))$
        Update m^a with $(x(i), y(i))$

by the ability of the learning model to correctly identify any concept changes and to recognise any recurrency in concepts. Hence, additional quality metrics are relevant in this chapter in order to analyse the performance of the approach, which are detailed below.

- *Recall*: the ratio between the number of correctly detected concept changes (i.e. corresponding to actual change points) and the total number of concept changes that have occurred during the learning process. *Recall* measures the ability of the learning model to detect all concept changes encountered. Low *Recall* indicates that there are changes in the underlying concept that were left undetected, resulting in accuracy drops due to continuing to utilise no-longer relevant classification models following the undetected drifts.

- *Precision*: the ratio between the number of correctly detected concept changes and the total number of detected concept changes during the learning process. *Precision* measures the degree to which the changes signalled by the learning model are correct. Low *Precision* indicates that the learner erroneously detects changes during periods of concept stability. This negatively affects both the accuracy and memory consumption. The former is due to the unnecessary instantiation and utilisation of new, unstable classifiers following the incorrectly detected changes, while the latter is due to the redundant storage of similar classifiers (belonging to the same concept).

- *Recurrency Recall*: the ratio between the number of correctly detected recurrent concepts and the total number of recurrent concepts that have occurred during the learning process. *Recurrency Recall* measures the ability of the learning model to recognise all concept recurrences encountered. Low *Recurrency Recall* indicates that there are recurrent concepts that the learner is not able to recognise as such, e.g. the learner treats such concepts as new concepts. Again, this

results in inefficient memory consumption (redundancy in classifiers), and affects accuracy since older relevant again examples, which boost classification stability, are ignored.

- *Recurrency Precision*: the ratio between the number of correctly detected recurrent concepts and the total number of detected recurrent concepts during the learning process. *Recurrency Precision* measures the degree to which the recurrences signalled by the learning model are correct. Low *Recurrency Precision* indicates that the learner erroneously identifies concepts as recurrent, resulting in the reuse of non-relevant classifiers, thus negatively affecting accuracy (similarly to low *Recall*).

As can be seen, each of the above metrics analyses different aspects in the performance of the learning model, which helps to better analyse its effectiveness in addressing concept changes. The values of each of the above measures range between 0 and 1, where the higher the value, the better.

6.4.1 Effect of Value Equivalence Threshold

In this section, we study the influence of the value equivalence threshold $eqThresh$ (with respect to which equivalence among context values is determined) on the performance of our proposed multi-model learner. In particular, we assume perfect contextual knowledge, and analyse the behaviour of the purely-contextual multi-model learner in detecting concept changes and identifying concept recurrences. Given that such analysis requires pre-existing knowledge about the drift points, we base our evaluation on an artificial dataset (the *STAGGER* dataset). The corresponding experimental design and results are discussed in what follows.

6.4.1.1 Experimental Design

We generate three *STAGGER* concepts, following concept sequence $1-2-3-1-2-3$, with added 5% label noise in each concept (see Chapter 3 for the definition of each concept). Context is generated by introducing an additional variable, $C \in \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$, where each value is associated with a certain concept. Specifically, $C = v_1 \vee v_2 \vee v_3$ under Concept 1, $C = v_4 \vee v_5 \vee v_6$ under Concept 2, and $C = v_7 \vee v_8 \vee v_9$ under Concept 3. In total, we generate 900 examples, with each context value lasting for 100 examples, following sequence $v_1 - v_2 - v_4 - v_5 - v_7 - v_8 - v_3 - v_6 - v_9$. In other words, the first 600 examples encounter 3 distinct concepts, while the last 300 examples exhibit 3 recurring concepts.

We compare the performance of our learner under three different settings for the value equivalence threshold $eqThresh$: a strict threshold ($eqThresh = 1$), a medium threshold ($eqThresh = 0.7$), and a relaxed (loose) threshold ($eqThresh = 0.3$). As mentioned in previous chapter, the equivalence among context values is learned separately in an offline mode, and then incorporated during classification into the adaptation process.

6.4.1.2 Results with Naive Bayes classifier

In this set of experiments, we adopt the Naive Bayes Classifier for the purpose of implementing each classification model $m \in \mathbb{M}$. Figures 6.4, 6.6, and 6.8 report the corresponding results (all results are averaged over 30 runs) for the three different value equivalence thresholds outlined above. As can be seen, setting the threshold to a medium level achieves the best results in terms of all the considered performance metrics.

Performance Measures	***eqThreshold* =0.7**
Overall Accuracy	0.899 (0.012)
Memory	3
Recall	1
Precision	1
Recurrency Recall	1
Recurrency Precision	1

FIGURE 6.4: Evaluation of the purely-contextual learner with medium $eqThresh$ (the standard deviation is in parenthesis).

In particular, when the value equivalence threshold is set appropriately to a medium level, i.e. $eqThresh = 0.7$ (see Figure 6.4), equivalence among context values is correctly identified. That is, values v_1, v_2, v_3 are considered equivalent, values v_4, v_5, v_6 are considered equivalent, and values v_7, v_8, v_9 are considered equivalent. As a result, the learner is able to correctly detect all concept changes (*Recall*=1, *Precision*=1), as well as to correctly recognise all concept recurrences as such (*Recurrency Recall*=1, *Recurrency Precision*=1). This contributes to achieving a high accuracy (0.899), and an optimum memory consumption (only 3 classifiers are stored in the library, corresponding to the three distinct concepts encountered). Figure 6.5 provides further analysis of the accuracy achieved over time in this case (the accuracy evolution is this experiment and all subsequent experiments are reported as a moving average over the last 15 examples). As can be seen, the learner is able to achieve a fast recovery after each change point during the first 600 time steps, due to detecting all such changes and utilising this to capture only relevant examples for learning the new corresponding concepts. As for the last 300 time steps, the learner always maintains a high accuracy by recognising all recurrences, thus eliminating the period of performance degradation after a change point due to reusing an already existing stable classifier from the library.

When the value equivalence threshold is too strict, i.e. $eqThresh = 1$ (see Figure 6.6), all context values $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9$ are considered to be distinct, neglecting existing similarities between these values. Given this, a drift will be detected too often,

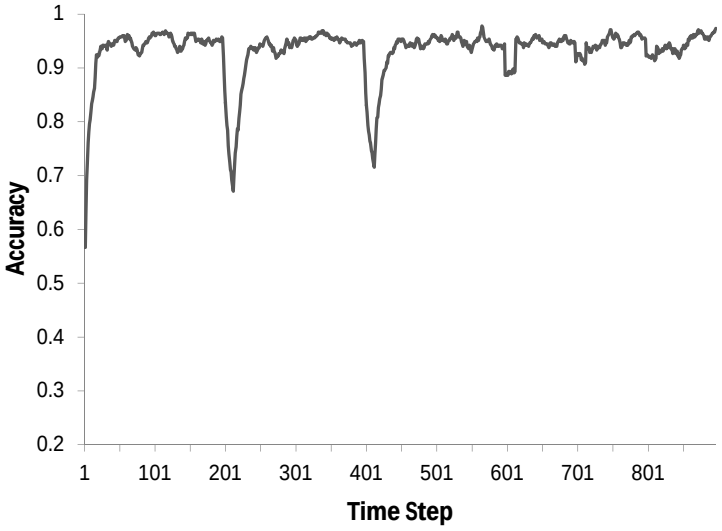


FIGURE 6.5: Accuracy Evolution of the purely-contextual learner with medium $eqThresh$.

whenever the context attribute switches to another value, even if this value is still associated with the same concept, resulting in high *Recall* (1), but low *Precision* (0.625). Moreover, all detected changes are regarded as new concepts, without recognising any recurrences (both *Recurrency Recall* and *Recurrency Recall* are equal to 0). These all lead to more than necessary memory consumption (9 classifiers, each corresponding to a possible context value), and to an effect on accuracy (0.838). Figure 6.7 further elaborates on the accuracy in this case, demonstrating its evolution over time. As can be seen, an accuracy drop is encountered at each change point of a context value to learn a new classifier, even during stability periods (the case at time step 100 and time step 300), and after recurrences (the case at time steps 600, 700, and 800).

On the other hand, when the value equivalence threshold is too relaxed (loose), i.e. $eqThresh = 0.3$ (see Figure 6.8), not only all existing similarities among context values are recognised, but also some dissimilar values are considered equivalent (in our experiments, context values $v_4, v_5, v_6, v_7, v_8, v_9$ are all considered equivalent). Although this eliminates signalling changes during periods of concept stability (highlighted by

Performance Measures	***eqThreshold*=1**
Overall Accuracy	0.838 (0.016)
Memory	9
Recall	1
Precision	0.625
Recurrency Recall	0
Recurrency Precision	0

FIGURE 6.6: Evaluation of the purely-contextual learner with strict $eqThresh$ (the standard deviation is in parenthesis).

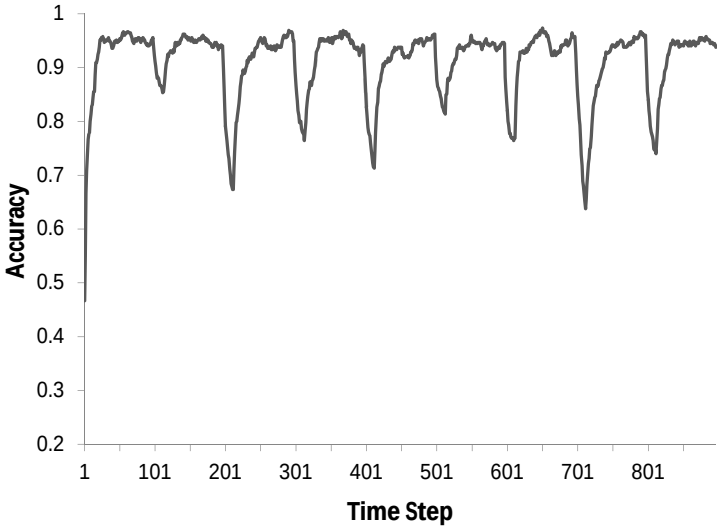


FIGURE 6.7: Accuracy Evolution of the purely-contextual learner with strict $eqThresh$.

the high *Precision* of 1), not all concept changes will be detected (which is evident in the low *Recall* value of 0.6). In particular, the learner is no longer able to distinguish between concept 2 and concept 3 (due to regarding their underlying context values as similar), and therefore suffer from considerable degradation in accuracy after time step 400 when concept 3 is encountered (see Figure 6.9). This is because, the classifier built under concept 2 remains the active classifier under concept 3, despite being no longer valid. Note that the learner here is able to recognise concept recurrences (2 out of 3 recurrences due to the inability to detect concept 3), with only 2 classifiers (less than the number of distinct concepts) being stored in the model library.

Performance Measures	***eqThreshold* =0.3**
Overall Accuracy	0.796 (0.022)
Memory	2
Recall	0.6
Precision	1
Recurrency Recall	0.667
Recurrency Precision	1

FIGURE 6.8: Evaluation of the purely-contextual learner with loose $eqThresh$ (the standard deviation is in parenthesis).

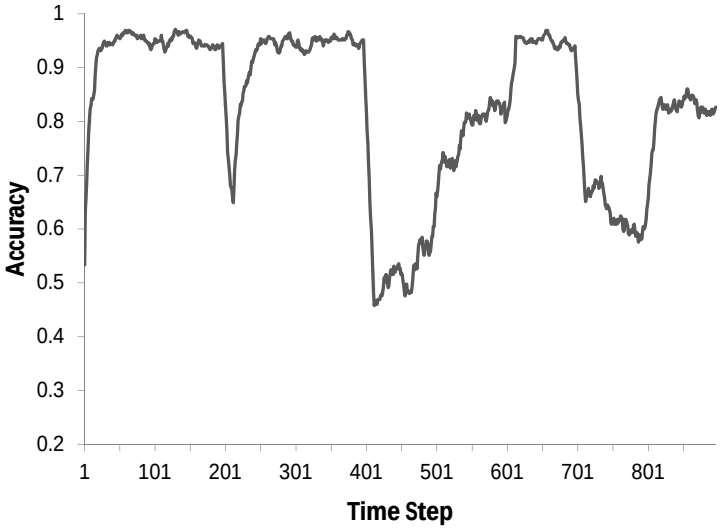


FIGURE 6.9: Accuracy Evolution of the purely-contextual learner with loose $eqThresh$.

Given the above analysis, the following experiments will be conducted with the value of $eqThresh$ being set to 0.7.

6.4.1.3 Results with Perceptron

In order to demonstrate the generality of the multi-model learner with other classification models, we repeat the above experiments with the three alternative values for

Performance Measures	***eqThreshold*=1**
Overall Accuracy	0.828 (0.006)
Memory	9
Recall	1
Precision	0.625
Recurrency Recall	0
Recurrency Precision	0

FIGURE 6.10: Evaluation of the purely-contextual learner with Perceptron Algorithm: strict $eqThresh$ (the standard deviation is in parenthesis).

Performance Measures	***eqThreshold* =0.7**
Overall Accuracy	0.857 (0.034)
Memory	3
Recall	1
Precision	1
Recurrency Recall	1
Recurrency Precision	1

FIGURE 6.11: Evaluation of the purely-contextual learner with Perceptron Algorithm: medium $eqThresh$ (the standard deviation is in parenthesis).

Performance Measures	***eqThreshold* =0.3**
Overall Accuracy	0.808 (0.021)
Memory	2
Recall	0.6
Precision	1
Recurrency Recall	0.667
Recurrency Precision	1

FIGURE 6.12: Evaluation of the purely-contextual learner with Perceptron Algorithm: loose $eqThresh$ (the standard deviation is in parenthesis).

$eqThresh$ with a perceptron algorithm. Similar behaviour is observed as for the multi-model learner when the perceptron algorithm is utilised for each model $m \in \mathbb{M}$, with the corresponding results being reported in Figures 6.10, 6.11, and 6.12.

6.4.2 Effect of Imperfect Contextual Knowledge

The above analysis assumes the availability of perfect contextual variable that is able to distinguish among all occurring concepts. Now, to demonstrate the effectiveness of the hybrid multi-model learner, we consider the case where we have imperfect context

Performance Measures	C_1
Overall Accuracy	0.786 (0.021)
Memory	2
Recall	0.6
Precision	1
Recurrency Recall	0.667
Recurrency Precision	1

Performance Measures	C_2
Overall Accuracy	0.81 (0.013)
Memory	2
Recall	0.6
Precision	1
Recurrency Recall	0.667
Recurrency Precision	1

FIGURE 6.13: Evaluation of the purely-contextual learner with imperfect context knowledge (the standard deviation is in parenthesis).

Performance Measures	C_1
Overall Accuracy	0.886 (0.009)
Memory	3
Recall	1
Precision	1
Recurrency Recall	1
Recurrency Precision	1

Performance Measures	C_2
Overall Accuracy	0.891 (0.009)
Memory	3
Recall	1
Precision	1
Recurrency Recall	1
Recurrency Precision	1

FIGURE 6.14: Evaluation of the hybrid learner with imperfect context knowledge (the standard deviation is in parenthesis).

variable(s). That is, when the concept change occurs with no associated changes in context. Similarly to the above experiments, we utilise the *STAGGER* dataset as described below.

6.4.2.1 Experimental Design

We utilise the same experimental setup described in Section 6.4.1 with regard to concept sequence settings. However, here we introduce 2 context variables (C_1, C_2), with their value distributions being as described in Figure 6.2.

6.4.2.2 Results

Figures 6.13 and 6.14 report the results of applying the two realisations of the multi-model learner in the case of imperfect context knowledge (i.e. when only one of the two variables C_1 and C_2 is available). As can be seen, the advantage of using the hybrid model over the purely-contextual model is evident in terms of accuracy. In

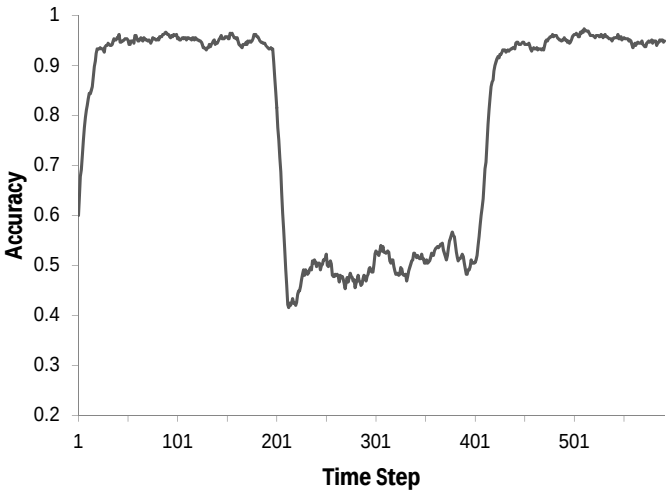


FIGURE 6.15: Accuracy Evolution of the purely-contextual learner with imperfect context knowledge.

particular, when only partial contextual knowledge is utilised, concept changes that are not associated with corresponding context value changes will not be recognised by the purely-contextual learner. As a result, it will fail to respond to such changes and suffer from performance degradation as illustrated in Figure 6.15. In contrast, the hybrid learner successfully detects and responds to all concept changes (even those that are not associated with context value changes). Further analysis is depicted in Figure 6.16. As can be seen, the learner correctly detects two change points: the first change point at time step 200 is detected with the help of monitoring the performance of the currently active model, while at time step 400, concept change is detected based on monitoring context variable value changes similarly to the purely-contextual learner.

In the case where both C_1 and C_2 are available, the two learners perform similarly as shown in Figures 6.17 and 6.18.

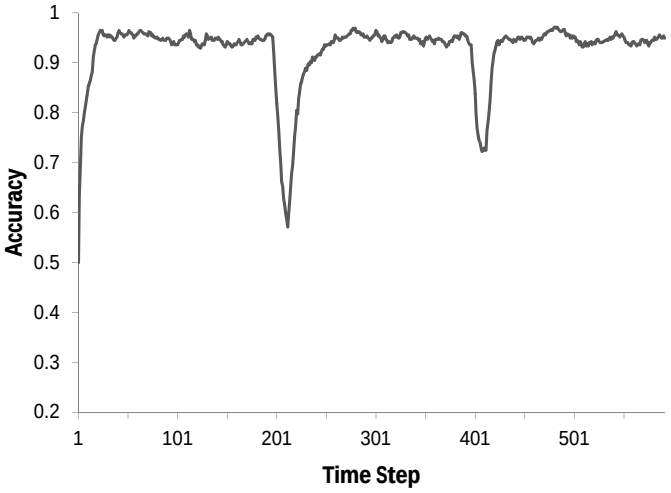


FIGURE 6.16: Accuracy Evolution of the hybrid learner with imperfect context knowledge.

Performance Measures	C_1 **and** C_2
Overall Accuracy	0.929 (0.006)
Memory	3
Recall	1
Precision	1
Recurrency Recall	1
Recurrency Precision	1

FIGURE 6.17: Evaluation of the purely-contextual learner with perfect context knowledge: collective concept distinction (the standard deviation is in parenthesis).

Performance Measures	C_1 **and** C_2
Overall Accuracy	0.911 (0.007)
Memory	3
Recall	1
Precision	1
Recurrency Recall	1
Recurrency Precision	1

FIGURE 6.18: Evaluation of the hybrid learner with perfect context knowledge: collective concept distinction (the standard deviation is in parenthesis).

6.4.3 Effect of Context Utilisation

In this section, our aim is twofold. First, we illustrate the benefit of context utilisation in achieving a faster adaptation to concept drift. To achieve this, we compare the two realisations of the proposed multi-model learner against an adaptive learner that is solely based on monitoring performance degradation (the adaptive learner of Algorithm 8).

In the second part, we compare the proposed approach with another adaptive context-based approach existing in the literature. In particular, we compare our approach against the MReC (Mining Recurring Concepts) learning model proposed by Gomes et al. [28] (see Chapter 2). In MReC model, unlike the proposed approach, context is not utilised for change detection or concept recurrency detection. It is only utilised as part of the model selection procedure for recurrent concepts.

6.4.3.1 Experimental Design

In both experiments, we utilise the real-world Elist dataset with its context variable *Location*. The dataset exhibits the concept sequence 1-2-1-2-1 (with a total of 1500 examples). As in the previous chapter, the statistical significance of the performance differences obtained are evaluated utilising the McNemar's test. The corresponding experimental results are discussed below.

6.4.3.2 Results

Tables 6.1, 6.2 and 6.3 report the corresponding results of applying the two realisations of the multi-model learner and the performance monitoring based adaptive classifier of Algorithm 8 (with *no context utilisation*). As can be seen, all approaches perform

TABLE 6.1: Effect of context utilisation: purely-contextual multi-model learner.

Performance Measures	
Overall Accuracy	0.80
Memory	2
Recall	1
Precision	1
Recurrency Recall	1
Recurrency Precision	1

similarly in terms of *Recall, Precision, Recurrency Recall* and *Recurrency Precision.* However, the multi-model learners outperform the approach with no context utilisation in terms of predictive accuracy. This is because, the utilisation of context enables a faster recognition of concept changes, which helps to avoid delays in concept change detection (i.e. eliminating the period of performance degradation). On the other hand, the approach with no context utilisation requires longer period before drift can be confirmed, and as a result suffers from accuracy drops at concept change points. This is evident in the reported overall accuracy, and is further illustrated in Figure 6.19, where we depict the evolution of predictive accuracy over time. As can be seen, the advantage of context utilisation is evident at concept change points in both cases of new and recurrent concepts.

We can also see that in the case of perfect contextual knowledge, as in the Elist dataset, the hybrid model performs slightly worse than the purely-contextual learner at change points (at time steps 300, 600, 900 and 1200). This is due to the recurrency detection principle assumed by the hybrid model, which requires an additional overhead (accumulating examples in the warning window for computing the conceptual equivalence) before recurrency can be confirmed (see Equation 6.5).

In the second part of the experimental analysis, we report the results of comparing the proposed multi-model learners against the MReC learning model. The results in Table 6.4 show that both realisations of the multi-model learner outperform the

TABLE 6.2: Effect of context utilisation: hybrid multi-model learner.

Performance Measures	
Overall Accuracy	0.79
Memory	2
Recall	1
Precision	1
Recurrency Recall	1
Recurrency Precision	1

TABLE 6.3: Effect of context utilisation: adaptive learner with no context utilisation.

Performance Measures	
Overall Accuracy	0.70
Memory	2
Recall	1
Precision	1
Recurrency Recall	1
Recurrency Precision	1

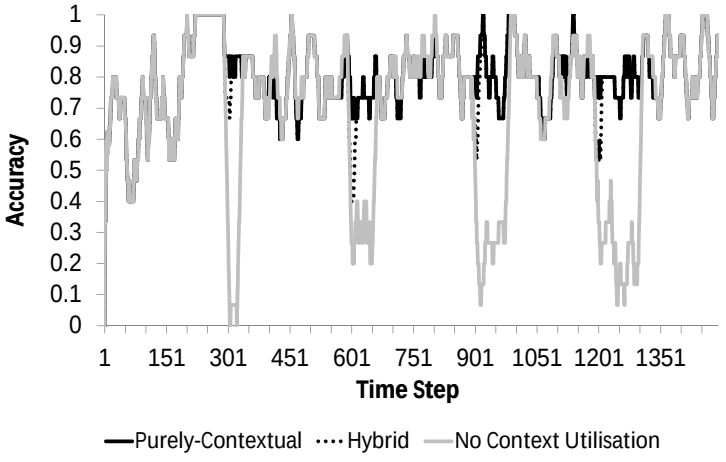


FIGURE 6.19: Accuracy evaluation of the multi-model learner against an adaptive learner with no context utilisation.

MReC model. As can be seen, the MReC model is able to detect all concept changes (*Recall*=1) and all concept recurrences (*Recurrency Recall*=1). However, the drift in this approach is detected too often as is evident in the low *Precision* (*Precision*=0.01) and high memory consumption (8 classifiers), which contributes to an effect on accuracy (0.76 compared to 0.80 and 0.79 for the purely-contextual and multi-model learner, respectively). Further analysis of this case is provided in Figure 6.20, where we depict the evolution of predictive accuracy over time. The benefit of context utilisation in the MReC approach in the cases of recurrent concepts is evident compared to the adaptive learner with no context utilisation in Figure 6.19 (faster selection of an old classifier from the stored models, resulting in a shorter period of performance degradation). However, MReC still relies only on accuracy to detect a drift, taking longer period before drift can be confirmed, as opposed to our learning model with incorporated context-based change detection that enables faster detection and selection of an appropriate classifier from the library.

TABLE 6.4: Comparison of multi-model learner with MReC.

Performance Metrics	**Purely-Contextual**	**Hybrid**	**MReC**
Accuracy	0.80	0.79	0.76
Kappa	0.60	0.58	0.53
AUC	0.88	0.87	0.83
Memory	2	2	8
Recall	1	1	1
Precision	1	1	0.01
Recurrency Recall	1	1	1
Recurrency Precision	1	1	0.63

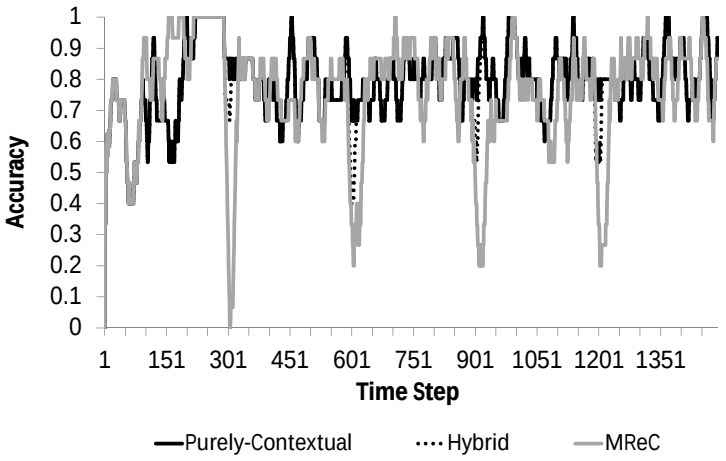


FIGURE 6.20: Accuracy evaluation of the multi-model learner against MReC.

6.5 Conclusion

In this chapter, we have presented an adaptive context-based learning model for handling concept drift in dynamic environments. In particular, the model is equipped with drift detection and recurrency detection strategies based on monitoring changes in relevant contextual variables. To address the challenges of perfect and imperfect contextual knowledge, two realisations of the adaptive model are proposed: a purely-contextual learner and a hybrid learner that supports context-based monitoring with additional monitoring of performance degradation. The latter provides additional evidence to support change detection in the cases where perfect contextual knowledge is not available. Experimental results demonstrated the ability of the proposed model to achieve a faster drift recovery compared to approaches with accuracy-based detection

methods. Moreover, the utilisation of contextual knowledge for detecting concept recurrences during the learning process enables the model to maintain a high accuracy following the change points.

Compared to the example weighting approach (see Chapter 5), which requires a weighted-example version of the prediction model, the proposed multi-model approach allows managing the drift at a meta-level, without making any changes to the base prediction model itself. Thus, it can be applied with a wider range of prediction models (e.g. some models might not support a weighted-example version).

Moreover, the proposed multi-model approach does not necessitate storage and re-processing of historical training examples in order to tackle recurring concepts. Instead, only the parameters of already-built models are stored and later reused, thus improving overall efficiency (in terms of time and storage). In other words, incremental classifiers can be utilised inside the learning model, allowing single data pass, without storing and re-processing past data. This makes the proposed approach also suitable for the case of data stream learning (with infinite size and imposed constraints on memory and computation resources). To further meet these constraints, it is possible to fix the size of the multi-model learner by introducing a strategy to remove one of the previous models whenever a new model is added to the learner.

Other challenges such as concept evolution can also be considered in the proposed learning model. In particular, it is possible to introduce an additional point to the detection principle of Section 6.3.2, which signals the emergence of new classes utilising one of the novel class detection techniques existing in the literature (e.g. [121], [120]).

A main assumption in the proposed learning model is that changes in concepts occur abruptly. Therefore, an interesting extension would be to adapt the model (in terms of drift detection and classifier selection approaches) to address the case of gradual changes. Extending the approach for gradual change is added to the future work (please

see Section 7.3.3). It is also worth noting that, as previously indicated, addressing incremental concept changes when the new concept appears after a transient period will not pose a challenge with the proposed context-aware approach. In particular, the examples related to the transient period between consecutive concepts will not be selected to update the classifier. This is because the contextual conditions of the examples during this period will not match that of the examples during the new concept. Hence, such examples will be eliminated from the selection process. These examples might only be considered at the beginning of the change when the concept is new and there is no sufficient number of examples.

Chapter 7

Conclusions and Future Work

Classification models are utilised in many domains to predict the unknown outcome of future data, thus improving decision making and problem solving. One of the main challenges when building such models is that, in the real world, the concept learned may change over time under the influence of varying contextual factors. This problem, usually referred to as concept drift, requires the ability of the classification model to adapt to such changes in order to be able to produce accurate and reliable classification decisions. The main contributions of this thesis towards addressing the problem of classification under concept drift are summarised in Section 7.1. The limitations of the thesis are provided in Section 7.2, followed by a list of possible future extensions of our work in Section 7.3.

7.1 Research Summary

Concept drift is usually linked to changes in various contextual variables. Therefore, unlike existing approaches, we propose to explicitly model and monitor the factors that cause the drift (i.e. the contextual variables and their changes), rather than monitoring

its consequences (i.e. the performance degradation). In particular, we propose to learn the contextual characteristics of the domain of interest, and exploit the learned knowledge at the classifier level for adapting to changes. For the purpose of the latter, we propose a context-based data weighting model for data selection, and a context-based drift detection model. More details are provided below.

7.1.1 Context Identification

In the first contribution of this thesis, we propose a context learning model that utilises historical data to infer variables that actually affect the concept of interest. For this purpose, an identification criterion based on Shannon's entropy and conditional mutual information measures is proposed. To address the challenge of estimating these measures in the case of continuous variables with high dimensionality, an appropriate approach based on the k-nearest neighbour estimator is presented. An extension of the identification approach for addressing correlations among context variables is also provided via a context selection algorithm. Finally, to address possible computational problems in the case of high context variable dimensionality, implementations of the selection algorithm based on popular approximation approaches are also presented.

Experimental results demonstrate the feasibility of the proposed identification approach and its robustness with respect to a number of factors. Moreover, the advantage of the proposed information-theoretic-based criterion to recognise context is demonstrated against other existing filter-based approaches. On the other hand, although the results show that the approach is able to recognise the relative importance among context variables under both abrupt and gradual changes, the absolute importance of the context variables is better recognised in the case of abrupt changes. Unsurprisingly, increasing noise levels negatively affects the performance of the approach in recognising the importance of context variables compared to noise-free settings. Finally, despite the

efficiency of the proposed context selection approach in addressing redundancy, implementing the approach with approximation methods achieves better results in high dimensionality settings (given limited sample size).

7.1.2 Context Utilisation

In terms of utilising the identified contextual information during classification, the following have been achieved.

7.1.2.1 Context-based Weighting

In the second contribution of this thesis, we propose to utilise context knowledge for discriminating between training examples in terms of their relevance for the current situation. In particular, the impact of each example on model estimation is controlled via a context-driven weighting factor, which can be added on top of an existing classification model that supports example weighting. This weighting factor corresponds to the degree of similarity between the context instance under which the example is collected and the current context instance. The similarity degree between two context instances is affected by the degree of equivalence between the values of relevant context variables in both instances, and the respective importance of these context variables. To assess the equivalence between two values of a context variable, two approaches are proposed. The first approach is simply based on the numerical difference between the values. The second approach is based on a more sophisticated semantics, where the equivalence between two context values is determined based on the similarity between their underlying distributions, which quantifies how these values discriminate among occurring concepts. Such a distribution-based equivalence is learned utilising two alternative information theoretic measures, namely Entropy Absolute Difference

(EAD) and Kullback-Leibler Divergence (KL). The EAD based approach demonstrated (empirically) a better ability to recognise equivalence among context values.

Moreover, experimental results on both artificial, benchmark, and real-world datasets demonstrate the feasibility of the approach, and its ability to improve the prediction accuracy of the classification model under various settings. Moreover, the proposed approach outperforms other state-of-the-art time-based adaptation strategies in both cases of new and recurring concepts. The results obtained also show the advantage of utilising context for weighting compared to incorporating context as additional input variable(s).

7.1.2.2 Context-based Drift Detection

In the third contribution of this thesis, context knowledge is utilised at a meta-level, without affecting the base classification model. In particular, for environments with perfect contextual knowledge, a purely contextual multi-model learner is proposed, which maintains a pool of classification models, each corresponding to different contextual circumstances (i.e. different concept). To detect a drift, the values of relevant contextual variables are monitored, and when a deviation is encountered, a drift is signalled, triggering the need for data re-selection actions. Such actions correspond to either initialising a new classifier, or reusing another stored classifier if concept recurrency is confirmed (based on comparing the current contextual conditions against those of the stored classifiers). For environments where perfect contextual knowledge is not necessarily available, a hybrid version of the multi-model learner is proposed, where contextual knowledge is augmented with performance degradation monitoring. Moreover, to confirm concept recurrency in such case of imperfect context knowledge, similarity in contextual conditions is supported with measuring the conceptual equivalence between classifiers.

Experimental analysis on benchmark and real-world datasets demonstrate the advantage of the proposed detection-based adaptation algorithms compared to other detection-based adaptation approaches in the literature (based on monitoring performance degradation and on explicit context utilisation).

7.2 Limitations

In the proposed context-aware adaptive approach, the identification of relevant contextual knowledge is performed as a preliminary step prior to classification. That is, we assume the availability of sufficient and representative training data to learn context. Moreover, we assume that the context learned remains stable over time. In other words, we assume that the data in the application set is affected by the same context variables as that of the training set.

Another main limitation is the high computational costs of the proposed context-based example weighting approach. In particular, it requires revisiting the contextual information of the examples each time the classifier is updated to assess the degree of similarity between the context of these examples and the current context. This implies no memory restrictions to keep all previous data examples with the contextual information associated with each example. In addition, the efficiency of the approach decreases with the continuous growth of training data.

Moreover, in the experimental analysis, we assume either abrupt or gradual changes to test the behaviour of the proposed approaches. However, in the real world a combination of both changes may occur together.

Finally, in connection to the decision support applications, to which this thesis mainly contributes, some efficiency constraints of data stream such as infinite size are not handled in the thesis. We also assume label availability immediately after classification.

This, however, is not applicable with certain decision support applications (e.g. credit worthiness evaluation).

It is also worth mentioning that in evaluating the learning algorithms presented in the thesis we use prequential analysis, which is commonly utilised for dynamic environments with concept drift problem. However, as indicated previously, validating learning models in dynamic environments is still an open issue [49]. In particular, the challenges introduced by dynamic environments (continuously evolving model, changes in data distributions and continuous data arrival) make standard evaluation methods such as cross-validation and other sampling methods not applicable [49].

7.3 Future Work

In what follows, we list a number of future directions that we believe are interesting extensions and improvements to our work.

7.3.1 Context Identification

The first interesting extension is to study an incremental identification of context variables. That is, we plan to explore how the importance of context variables can be identified over time in the absence of pre-existing training data, i.e. when training data becomes available only gradually. The information theoretic measures proposed enable to perform such incremental learning. Moreover, it would also be interesting to explore the case where the importance of the context variables may change over time. Given this, the weight of the context variables should not remain static once identified, but updated continuously on the availability of new data examples. For this purpose, it is possible to apply some forgetting approaches (windowing or weighting) with respect

to the available past observations, so that context identification would be a continuous learning process.

7.3.2 Example Weighting

To account for resource boundedness, it would be interesting to explore the possibility of an online context-based weighting approach to adjust example weights without the need to reiterate over the past data.

Moreover, the weighting process assumes continuous drift, without attempting to identify when a drift actually occurs, and thus constantly re-weights the training examples whenever the classifier needs to be updated with new data. Therefore, it would also be interesting to explore a triggered-based re-weighting approach, which combines example weighting with drift detection. That is, re-weighting of the training examples can be triggered only if the recent contextual circumstances change. This eliminates the need for the unnecessary reiteration over past examples to update their weights in the absence of drift.

Finally, learning the equivalence among context values incrementally, in the absence of pre-existing training knowledge, is also an interesting extension to our work.

7.3.3 Drift Detection

The context-aware adaptive multi-model learners proposed for drift detection do not account for the case of gradual changes. Therefore, an interesting research direction is to design an appropriate extensions to the proposed drift detection and classifier selection mechanisms (for example introducing a weighting function into the model selection process), so that the model will also operate in gradually changing environments.

Testing the behaviour of the proposed hybrid model learner with other change detection methods would also be interesting. It is also worth noting that the proposed context-based drift detection approach may be beneficial for the case of unlabelled streaming data (i.e. where the class labels of the examples are not readily available). On the other hand, existing change detection approaches mainly depend on monitoring the error rate of the classification model, which makes them not applicable for the unlabelled data case.

7.3.4 Other Research Directions

In addition to the above mentioned future work extensions to the proposed work, it would also be interesting to explore the following research direction. First, test the behaviour of the proposed algorithms with other change types and scenarios (such as local concept changes and virtual changes), the possibility of new class emergence, and the presence of complex concept drift (i.e. when more than one type of drift is present at the same time). The second important future research direction is to address the constraints of data steam learning (e.g. infinite size, limited memory and computation resources, one-shot treatment etc.), which extend the applicability of the proposed algorithms to other application domains.

Finally, the usefulness of the proposed algorithms can be further tested with other real-world domains that exhibit concept drift and contextual characteristics, such as consumer credit scoring and stock market price prediction. Both these domains are subject to concept drift that is usually linked to changes in different macroeconomic variables, which represent contextual characteristics for these domains.

Appendix A

Evaluation of *k*-Nearest Neighbour Approach

Appendix A

In the following, we test the accuracy of the conditional mutual information (CMI) estimator given in Equation 4.20 and analyse the influence of various factors that could affect this estimate utilising bias and relative bias as a performance criteria.

A.1 Experimental Setup and Results

In order to evaluate the performance of the k-nearest neighbour estimate, we will use multivariate normally distributed variables with zero mean and unit variance, because in this case we are able to compare with analytic results. Specifically, the theoretical value of the conditional mutual information for normally distributed variables can be

derived in terms of Equation 4.31, with the entropies being calculated according to the definition of entropy for a multivariate normal distribution given as follows [24]:

$$H(X_1, X_2, ..., X_d) = \frac{1}{2} \log_2[(2\pi e)^d |\Sigma|] \tag{A.1}$$

where $|\Sigma|$ is the determinant of the covariance matrix Σ.

The bias of the estimates with respect to the theoretical values will be computed in terms of the absolute bias: $|\hat{I} - I|$, and the relative bias: $\frac{|\hat{I}-I|}{I}$, where $\hat{I}$ is the average of the obtained estimates, and I is the theoretical value.

Let us assume that $Z = (C, Y, X)$ is normally distributed random variable of dimension d_z (the variables C,Y,X can be either scalars or multidimensional) with zero mean values and covariance matrix M, where$M_{i,j} = 1$, for$i = j$, and $M_{i,j} = r$, for $i \neq j$, $i, j = 1 : d_z$. Here, we assume that C,Y are scalars, while X is an element of a higher dimensional space, with dimension $d_x = 3$. Each variable is made to be correlated with the other variables with correlation, r, ranging from 0.9 to 0.6 with a step size of 0.1.

Here, with reference to a predictive model, X corresponds to a vector of the primary variables, while C, Y to the candidate context variable, and the target variable, respectively. Three factors will be considered in evaluating the accuracy of the CMI estimator: the value of the parameter k, the dimension of the dataset, and the number of available data samples. The related experiments are detailed next.

A.1.1 Varying Parameter *k*

In the first case, we analyse the influence the chosen value of parameter k has on the estimation accuracy. For this purpose, we generate 3000 examples in 5-dimensional space, as described above, using correlation matrix M, and vary the value of k from 2 to 30 as illustrated in Figures A.1 and A.2.

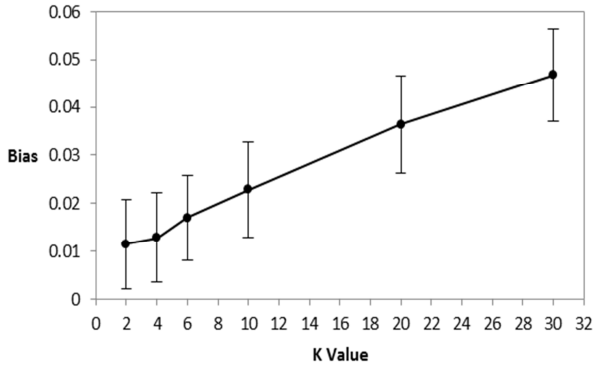


FIGURE A.1: Absolute bias of CMI estimator given various k values averaged over 20 runs (true value= 0.318). The error bars denote the standard deviations.

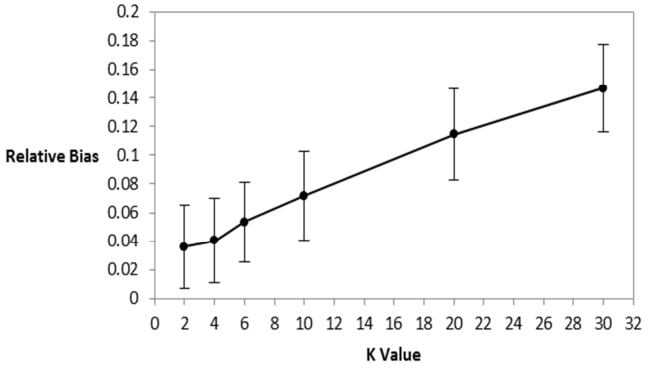


FIGURE A.2: Relative bias of CMI estimator given various k values averaged over 20 runs (true value= 0.318). The error bars denote the standard deviations.

As can be noticed, the biases of conditional mutual information estimator are relatively low (less than 0.05 and 0.15 for absolute biases and relative biases, respectively) for all tested values of k, where smaller values of k get better estimation accuracy (lower biases) with approximately the same variability.

On the other hand, when the relevance between variables is very small, i.e. the true value of CMI is close to zero, small values of k lead to larger variability in the estimates.

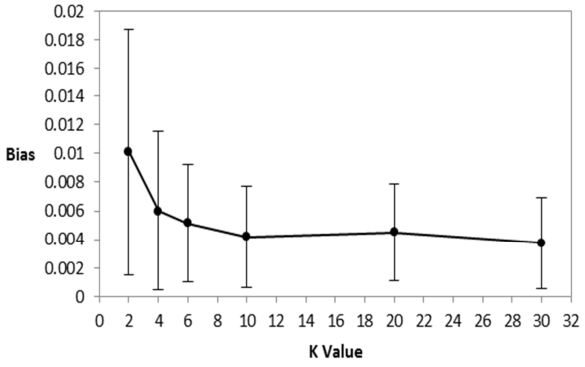


FIGURE A.3: Absolute bias of CMI estimator given various k values averaged over 20 runs (true value= 0.0305). The error bars denote the standard deviations.

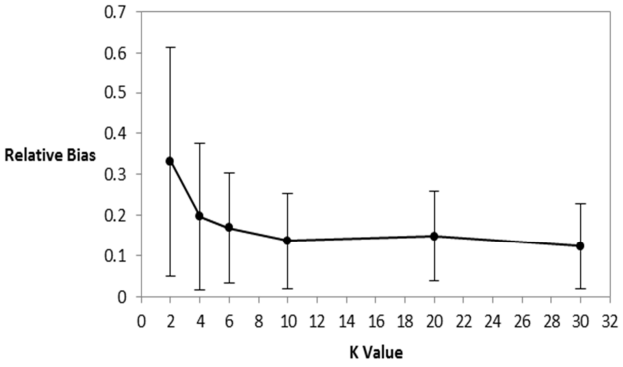


FIGURE A.4: Relative bias of CMI estimator given various k values averaged over 20 runs (true value= 0.0305). The error bars denote the standard deviations.

To illustrate this, we repeat the above experiment, but, this time, fixing the correlation to the same value among all variables, i.e.$M_{i,j} = 1$, for $i = j$ and $M_{i,j} = 0.9$, for $i \neq j$, so that the true value of CMI is 0.0305.

The results in Figures A.3 and A.4 show a different pattern with respect to the values of k, where small values of k lead to higher biases and larger variability. Based on this, a balanced choice for the value of k to account for both bias and variability given

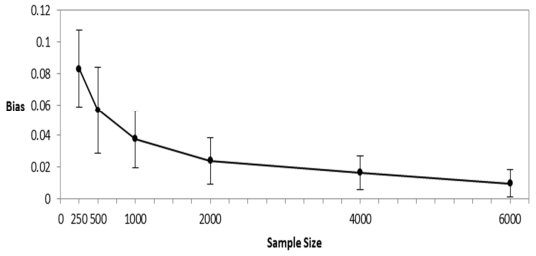


FIGURE A.5: Absolute bias of CMI estimator with different sample sizes averaged over 20 runs (true value= 0.318). The error bars denote the standard deviations.

different values of the CMI, would be to take a mid-range value, e.g. $k = 6$, which will be adopted in the following experiments.

A.1.2 Varying Sample Size

In the second case, we test how CMI estimator is affected by different numbers of available data samples. In order to do this, we fix the value of k to 6, and vary the sample size from 250 to 6000. As in the previous case, we examine the influence of the sample size with respect to two different values of the CMI. Specifically, with varying correlation between variables, i.e. true value of CMI is 0.318 (Figures A.5 and A.6), and the same correlation between variables, i.e. true value of CMI is 0.0305 (Figurse A.7 and A.8).

In general, it can be seen that both bias and variability decrease as the number of samples increases. For example, increasing the sample size from 250 to 500 in Figures A.5 and A.6 reduces the bias and relative bias by around 3% and 8%, respectively. The reduction level gradually decreases to become around 0.005 for bias and 0.02 for relative bias, when increasing the sample size from 4000 to 6000. A similar trend can be observed in Figures A.7 and A.8, which reports the results for small value of the

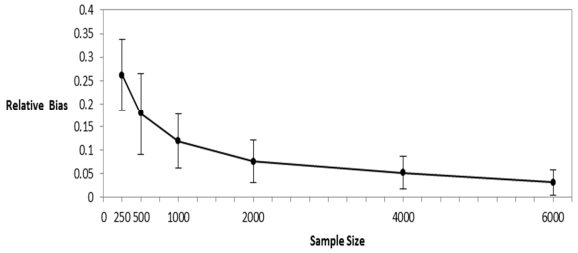


FIGURE A.6: Relative bias of CMI estimator with different sample sizes averaged over 20 runs (true value= 0.318). The error bars denote the standard deviations.

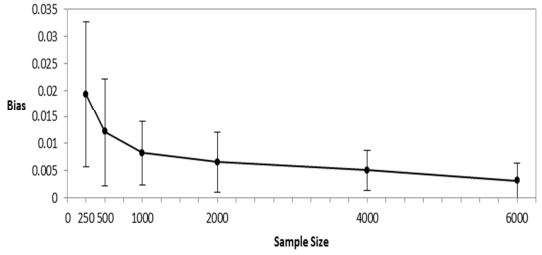


FIGURE A.7: Absolute bias of CMI estimator with different sample sizes averaged over 20 runs (true value= 0.0305). The error bars denote the standard deviations.

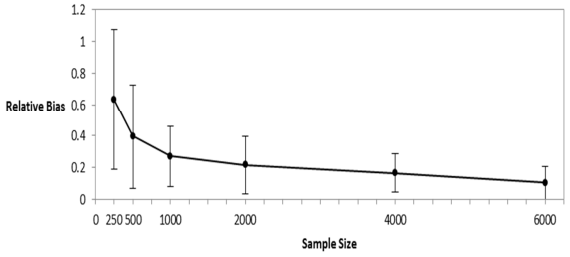


FIGURE A.8: Relative bias of CMI estimator with different sample sizes averaged over 20 runs (true value= 0.0305). The error bars denote the standard deviations.

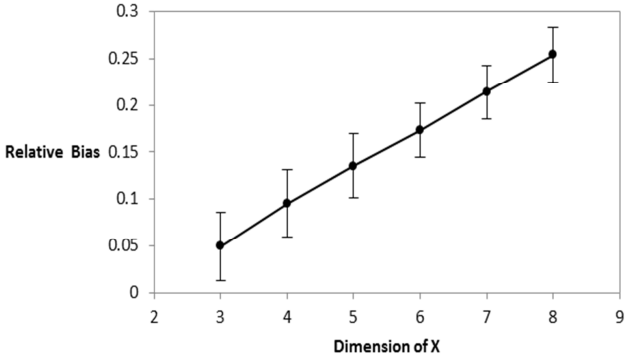


FIGURE A.9: Biases of CMI estimator with various dimensions of X averaged over 20 runs (true CMI> 0).

CMI. In other words, the CMI estimator converges to the true value with the increasing sample size, similarly to the MI estimator reported by Kraskov et al. [29].

A.1.3 Varying Dimension

In the last case, we study the effect of a change in the dimension of Z to the accuracy of the estimates. To do this, we increase the dimension of Z from 5 to 10, i.e. $Z = (C, Y, X_3, ..., X_10)$, and generate 4000 realizations of variable Z with the following covariance matrix $M : M_{i,j} = 1$, for $i = j$, and $M_{i,j} = r$, for $i \neq j$, where r varies from 0.9 to 0.1. In the following, we vary the dimension of X from 3 to 8. The results as illustrated in Figure A.9.

We can see from Figure A.9 that the bias proportionally increases with higher dimensions when CMI> 0, with approximately the same variability. Similarly, when the true CMI have values close to zero, increasing the dimensionality worsens the accuracy of the estimator, but with higher variability as shown in Figure A.10. Note that, since the actual value of CMI also varies with the varying dimensionality (i.e. takes smaller

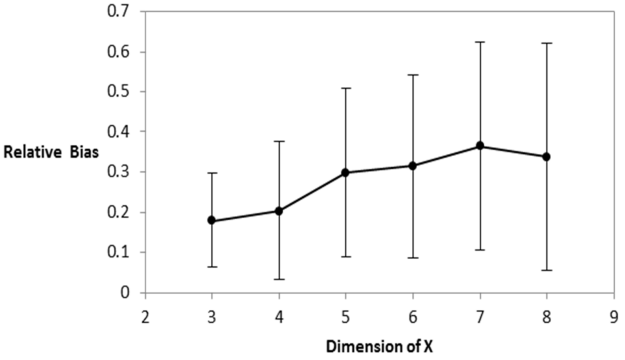


FIGURE A.10: Biases of CMI estimator with various dimensions of X averaged over 20 runs (true CMI around zero).

values as the dimensionality increases), we only report here the results for the relative bias.

A.2 Conclusion

The experimental results show that the influence degree of the various factors tested on the accuracy of the k-nearest neighbour approach in estimating conditional mutual information is linked to the true value of the estimate (i.e. the degree of dependency between variables). For example, when the variables are independent (true CMI= 0), small values of k lead to higher bias and larger variability. In contrast, when variables are not independent (true CMI$>$ 0), the estimation accuracy decreases with larger values of k. Furthermore, although the estimator performs reasonably well with higher dimensions, the experiments show that increasing the dimensionality worsen the accuracy of the estimates. On the other hand, the experimental results with varying sample size indicate that the estimator converges to the true value with the increasing sample size.

Bibliography

[1] L. Barakat. Context Identification and Exploitation in Dynamic Data Mining: an Application to Classifying Electricity Price Changes. In: Bouchachia A. (eds) Adaptive and Intelligent Systems. Lecture Notes in Computer Science, vol. 8779, pages 80-89, Springer, Heidelberg, 2014.

[2] L. Barakat. A Context-Driven Data Weighting Approach for Handling Concept Drift in Classification. In Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015, Springer, Wroclaw, pages 383-393, 2015.

[3] L. Du, Q. Song, and X. Jia. Detecting Concept Drift: an Information Entropy based Method Using an Adaptive Sliding Window. Intelligent Data Analysis. 18, pages 337-364, 2014.

[4] E. Frank, M. Hall, and B. Pfahringer. Locally Weighted Naive Bayes. In: Proc. of the 19th conference on Uncertainty in Artificial Intelligence, pages 249-256, 2003.

[5] J. Gama, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A Survey on Concept Drift Adaptation. ACM Computing Surveys. 46(4), pages 1-37, 2014.

[6] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with Drift Detection. In: Proceedings of the 17th Brazilian Symposium on Artificial Intelligence, LNAI 3171, pages 286-295, 2004.

[7] M. B. Garcia, J. del Campo-Avila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno. Early Drift Detection Method. In: ECML PKDD 2006 Workshop on Knowledge Discovery from Data Streams, 2006.

[8] M. Harries. Splice-2 Comparative Evaluation: Electricity Pricing. Technical report, University of New South Wales, 1999.

[9] M. Harries, C. Sammut, and K. Horn. Extracting Hidden Context. Machine Learning. 32(2), pages 101-126, 1998.

[10] G. Hulten, L. Spencer, and P. Domingos. Mining Time-Changing Data Streams. In Proc. of the 7th ACM SIGKDD int. conf. on Knowledge Discovery and Data Mining, pages 97-106, 2001.

[11] I. Katakis, G. Tsoumakas, and I. Vlahavas. Tracking Recurring Contexts Using Ensemble Classifiers: an Application to Email Filtering, Knowledge and Information Systems. 22(3), pages 371-391, 2010.

[12] I. Koychev. Gradual Forgetting for Adaptation to Concept Drift. In: Proc. of ECAI Workshop Current Issues in Spatio-Temporal Reasoning,pages 101-106, 2000.

[13] M. Kubat. Floating Approximation in Time-Varying Knowledge Bases. Pattern Recognition Letters. 10(4), pages 223-227, 1989.

[14] N. G.Pavlidis, D. K. Tasoulis, N. M Adams, and D. J Hand. λ-Perceptron: An Adaptive Classifier for Data Streams. Pattern Recognition. 44(1), pages 78-96, 2011.

[15] N. G. Pavlidis, D. K. Tasoulis, N. M. Adams, and D. J. Hand. Adaptive Consumer Credit Classification. Journal of the Operational Research Society. 63(12), pages 1645-1654, 2012.

[16] W. N. Street and Y. S. Kim. A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. In: Proc. of the 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pages 377-382, 2001.

[17] R. Sebastiao and J. Gama. Change Detection in Learning Histograms from Data Streams. In Proceedings of the Portuguese Conference on Artificial Intelligence. LNAI 4874, pages 112-123, 2007.

[18] J. Schlimmer and R. Granger. Incremental Learning from Noisy Data. Machine Learning. 1(3), pages 317-354, 1986.

[19] A. Tsymbal. The Problem of Concept Drift: Definitions and Related Work. Computer Science Department, Trinity College Dublin, 2004.

[20] H. Wang, W. Fan, P. S Yu, and J. Han. Mining Concept-Drifting Data Streams Using Ensemble Classifiers. In: Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pages 226-235, 2003.

[21] G. Widmer and M. Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning. 23(1), pages 69-101, 1996.

[22] I. Zliobaite and L. Kuncheva. Determining the Training Window for Small Sample Size Classification with Concept Drift. In Proceedings of the IEEE International Conference on Data Mining Workshop, pages 447-452, 2009.

[23] I. Zliobaite. How Good is the Electricity Benchmark for Evaluating Concept Drift Adaptation. CoRR, abs/1301-3524, 2013.

[24] T. M. Cover and J. A. Thomas. Elements of Information Theory. Wiley, New York, 2006.

[25] A. Chen. Context-Aware Collaborative Filtering System: Predicting the User's Preferences in the Ubiquitous Computing. In: International Workshop on Location- and Context-Awareness, pages 244-253, Oberpfaffenhofen, Germany, 2005.

[26] A. Dey. Understanding and Using Context. Personal and Ubiquitous Computing, 5(1), pages 4-7, 2001.

[27] D. Francois, V. Wertz, and M. Verleysen. The Permutation Test for Feature Selection by Mutual Information. In: Proc. of the 14th European Symposium on Artificial Neural Net-works, pages 239-244, 2006.

[28] J.B. Gomes, P.A. Sousa, and E. Menasalvas. Tracking Recurrent Concepts Using Context. Intelligent Data Analysis, 16(5), pages 803-825, 2012.

[29] A. Kraskov, H. Stogbauer, and P. Grassberger, P. Estimating Mutual Information. Physics Re-view E. 69(6), pages 66138-66154, 2004.

[30] C. Shannon. A Mathematical Theory of Communication. The Bell System Technical Journal 27(7), pages 379-423, 1948.

[31] P. Turney. The Identification of Context-Sensitive Features: A Formal Definition of Context for Concept Learning. In: Proc. of the ICML Workshop on Learning in Context-Sensitive Domains, pages 53-59. Bari, Italy, 1996.

[32] G. Widmer. Tracking Context Changes through Meta-Learning, Machine Learning 27(3), pages 259-286, 1997.

[33] I. Zliobaite. Identifying Hidden Contexts in Classification. In: Proc. of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery, pages 277-288. Springer, 2011.

[34] R. Battiti. Using Mutual Information for Selecting Features in Supervised Neural Net Learning, IEEE Transactions on Neural Networks, 5(4), pages 537-550, 1994.

[35] H. Cheng, Z. Qin, C. Feng, Y. Wang, and F. Li. Conditional Mutual Information-based Feature Selection Analysing for Synergy and Redundancy, Electronic Telecommunication Research Institution, 33(2), pages 210-218, 2011.

[36] P.A. Estevez, M. Tesmer, C. A. Perez, and J. M. Zurada. Normalized Mutual Information Feature Selection, IEEE Transactions on Neural Networks, 20(2), pages 189-201, 2009.

[37] S. Frenzel and B. Pompe. Partial Mutual Information for Coupling Analysis of Multivariate Time Series, Physical Review Letters, 99, 204101, 2007.

[38] V. Gomez-Verdejo, M. Verleysen, and J. Fleury. Information-theoretic Feature Selection for Functional Data Classification, Neurocomputing, 72, pages 3580-3589, 2009.

[39] S. Khan, S. Bandyopadhyay, A.R. Ganguly, S. Saigal, D.J. Erickson, V. Protopopescu, and G. Strouchov. Relative Performance of Mutual Information Estimation Methods for Quantifying the Dependence Among Short and Noisy Data, Physical Review E 76, 026209, 2007.

[40] L. F. Kozachenko and N. N. Leonenko. A Statistical Estimate for the Entropy of a Random Vector, Problems Information Transmission, 23(2), pages 9-16, 1987.

[41] N. Kwak and C. Choi. Input Feature Selection by Mutual Information based on Parzen Window, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(12), pages 1667-1671, 2002.

[42] A. Papana and D. Kugiumtzis. Evaluation of Mutual Information Estimators on Nonlinear Dynamic Systems, Nonlinear Phenom Complex Syst, 11, pages 225-232, 2008.

[43] H. Peng, F. Long, and C. Ding. Feature Selection based on Mutual Information Criteria of Max-Dependency, Max-Relevance, and min-redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(8), pages 1226-1238, 2005.

[44] F. Rossi, A. Lendasse, D. Francois, V. Wertz, and M. Verleysen. Mutual Information for the Selection of Relevant Variables in Spectrometric Nonlinear Modelling, Chemometrics and Intelligent Laboratory Systems, vol. 80, pages 215-226, 2006.

[45] R. Steuer, J. Kurths, C. O. Daub, J. Weise, and J. Selbig. The Mutual Information: Detecting and Evaluating Dependencies between Variables, Bioinformatics 18 (Suppl. 2), S231-S240, 2002.

[46] A. Tsimpiris, I. Vlachos, and D., Kugiumtzis. Nearest Neighbor Estimate of Conditional Mutual Information in Feature Selection, Expert Systems with Applications 39, pages 12697-12708, 2012.

[47] J.R. Vergara and P.A. Estevez. A Review of Feature Selection Methods based on Mutual Information, Neural Computing and Applications, DOI 10.1007/s00521-013-1368-0, 2013.

[48] I. Vlachos and D. Kugiumtzis. Non-uniform State Space Reconstruction and Coupling Detection, Physical Review E, 82, 016207, 2010.

[49] J. Gama. Knowledge Discovery from Data Streams, Chapman & Hall/CRC, 2010.

[50] J. Cohen. A Coefficient of Agreement for Nominal Scales, Educational and Psychological Measurement, 20(1), pages 37-46, 1960.

[51] T. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, Neural Computation, 10(7), pages 1895-1923, 1998.

[52] A. Narasimhamurthy and L. Kuncheva. A Framework for Generating Data to Simulate Changing Environments. In Proceedings of the Twenty Fifth IASTED

International Multi-Conference on Artificial Intelligence and Applications (AIA07), Innsbruck, Austria, pages 384-389, 2007.

[53] M. Hall and G. Holmes. Benchmarking Attribute Selection Techniques for Discrete Class Data Mining, IEEE Transactions on Knowledge and Data Engineering, Vol. 15 (6), pages 1437-1447, 2003.

[54] M. A. Hall. Correlation-Based Feature Subset Selection for Machine Learning, Univ. Waikato, Waikato, New Zealand, 1999.

[55] H. Liu, J. Li, and L. Wong. A Comparative Study on Feature Selection and Classification Methods Using Gene Expression Profiles and Proteomic Patterns, Genome Informatics,13, pages 51-60, 2002.

[56] U. M. Fayyad and K. B. Irani. Multi-interval Discretisation of Continuous-Valued Attributes for Classification Learning. In Proceedings of the Thirteenth International Join Conference on Artificial Intelligence, Morgan Kaufmann, 1993.

[57] Y. Yang, X. Wu and X. Zhu. Mining in Anticipation for Concept Change: Proactive-Reactive Prediction in Data Streams, Data Mining and Knowledge Discovery 13(3), pages 261-289, 2006.

[58] G.H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, 1, pages 338-345, 1995.

[59] P.N. Tan, M. Steinbach, and V. Kumar. Introduction to Data Mining, Pearson Addison-Wesley, 2006.

[60] A. Bifet, J. Read, I. Zliobaite, B. Pfahringer, and G. Holmes. Pitfalls in Benchmarking Data Stream Classification and How to Avoid Them. In Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, pages 465-479, 2013.

[61] D. J. Hand. Measuring Classifier Performance: A Coherent Alternative to the Area Under the ROC Curve. Machine Learning 77(1), pages 103-123, 2009.

[62] L. Kuncheva. Classifier Ensembles for Changing Environments. New York: Springer-Verlag, vol. 3077, pages 1-15, 2004.

[63] R. Duda, P. Hart, and D. Stork. Pattern Classification (2nd Edition). Wiley-Interscience, 2000.

[64] M. Kelly, D. Hand, and N. Adams. The Impact of Changing Populations on Classifier Performance. In KDD '99: Proc. of the 5th ACM SIGKDD int. conf. on Knowledge discovery and data mining, pages 367-371, 1999.

[65] I. Zliobaite, Learning under Concept Drift: an Overview. Faculty of Mathematics and Informatics, Vilnius University, Lithuania, 2010.

[66] N.M. Adams, D.K. Tasoulis, C. Anagnostopoulos and D.J. Hand. Temporally-Adaptive Linear Classification for Handling Population Drift in Credit Scoring. In Proceedings of the 19th International Conference on Computational Statistics, Springer, pages 167-176, 2010.

[67] G. Widmer and M. Kubat. Effective Learning in Dynamic Environments by Explicit Context Tracking. In Proceedings of the Sixth European Conference on Machine Learning, pages 227-243, Berlin: Springer Verlag, 1993.

[68] S. Bach and M. Maloof. Paired Learners for Concept Drift, In Proc. of the 8th IEEE Int. Conf. on Data Mining, pages 23-32. IEEE Press, 2008.

[69] A. Bifet and R. Gavalda. Learning from Time-Changing Data with Adaptive Windowing, In: SDM, pages 443-448, Citeseer, 2007.

[70] R. Klinkenberg and T. Joachims. Detecting Concept Drift with Support Vector Machines. In Proceedings of the Seventeenth International Conference on Machine

Learning (ICML), P. Langley, ed., San Francisco, CA, USA. Morgan Kaufmann, pages 487-494, 2000.

[71] J. Kolter and M. Maloof. Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. Journal of Machine Learning Research, (8), pages 2755-2790, 2007.

[72] P. Turney, Exploiting Context when Learning to Classify. In Proceedings of the European Conference on Machine Learning, ECML-93, pages 402-407. Vienna, Austria: Springer-Verlag, 1993.

[73] P. Turney. Robust Classification with Context-Sensitive Features, In Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-93, pages 268-276, Edinburgh, Scotland: Gordon and Breach, 1993.

[74] P. Turney and M. Halasz. Contextual Normalization Applied to Aircraft Gas Turbine Engine Diagnosis, Journal of Applied Intelligence, 3, pages 109-129, 1993.

[75] P. Turney. The Management of Context-Sensitive Features: A Review of Strategies. In Proceedings of the ICML-96 Workshop on Learning in Context-Sensitive Domains, pages 53-69, 1996.

[76] M. Jia-hui, C. Peng, and C. Jin. Context of the Concept Drift in Data Mining: An Empirical Study on the Regional Economic Influence to the Relation between Demographic Attributes and Credit Card Holder's Loyalty. In Proceedings of the 15th International Conference on Management Science and Engineering, Long Beach, USA, September 10-12, 2008.

[77] I. Koychev. Learning about Users in the Presence of Hidden Context. In R. Schfer, M. E. Mller, and S. A. Macskassy, editors. In Proceedings of the UM2001 Workshop on Machine Learning for User Modeling, pages 49-58, Sonthofen, Germany, July 13-17, 2001.

[78] S. Mandl, B. Ludwig, S. Schmidt, and H. Stoyan. Recurring Hidden Contexts in Online Concept Learning, In Workshop on Planning, Learning and Monitoring with Uncertainty in Dynamic Worlds, pages 25-29, 2006.

[79] S. Mandl, S. Coping with Unconsidered Context in Machine Learning Scenarios. Augsburg University, Department of Computer Science, 2011.

[80] R. Klinkenberg. Learning Drifting Concepts: Example Selection vs. Example Weighting. Intelligent Data Analysis, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift, 8 (3), 2004.

[81] E. S. Page. Continuous Inspection Schemes. Biometrika, Vol. 41:100-115, 1954.

[82] K. Nishida and K. Yamauchi. Detecting Concept Drift Using Statistical Testing. In Proceedings of the 10th international conference on Discovery science. DS07. Springer-Verlag, Berlin, Heidelberg, pages 264-269, 2007.

[83] P. Vorburger and A. Bernstein. Entropy-based Concept Shift Detection. In Proceedings of the 6th Int. Conf. on Data Mining. ICDM. pages 1113-1118, 2006.

[84] J. Gama and P. Kosina. Recurrent Concepts in Data Streams Classification. Knowledge and Information System, 40 (3), pages 489-507, 2014.

[85] W. Fan. Systematic Data Selection to Mine Concept-Drifting Data Streams. In: KDD, pages 128-137. ACM, New York, 2004.

[86] P. Domingos and M. Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. Machine Learning, 29(2):103:130, 1997.

[87] I. Witten, E. Frank, and M. Hall. Data Mining: Practical Machine Learning Tools and Techniques (3rd Edition). Morgan Kaufmann, 2011.

[88] M. Kubat and G. Widmer. Adapting to Drift in Continuous Domains. In Proceedings of the 8th European Conference on Machine Learning, pages 307-310. Springer, Heidelberg, 1995.

[89] M. Lazarescu and S. Venkatesh. Using Selective Memory to Track Concept Drift Effectively. Intelligent Systems and Control, vol. 388, Salzburg, Austria, ACTA Press, 2003.

[90] I. Khamassi, M. Sayed-Mouchaweh, Moez Hammami, Khaled Ghedira. Discussion and Review on Evolving Data Streams and Concept Drift Adapting, Springer, Evolving Systems Journal, October 2016.

[91] I. Khamassi, M. Sayed-Mouchaweh, Moez Hammami, Khaled Ghedira. Self-Adaptive Windowing Approach for Handling Complex Concept Drift, Cognitive Computation, Springer, vol. 7, June 2015.

[92] T. Mitchell. Machine Learning. MacGraw-Hill Companies, Inc., 1997.

[93] J. Beringer and E. Hullermeier. Efficient Instance-based Learning on Data Streams. Intelligent Data Analysis, 11(6), pages 627-650, 2007.

[94] I. Zliobaite. Combining Similarity in Time and Space for Training Set Formation Under Concept Drift. Intelligent Data Analysis, 15(4), pages 589-611, 2011.

[95] V. Ganti, J. Gehrke, R. Ramakrishnan. Mining Data Streams Under Block Evolution. SIGKDD Exploration Newsletter 3 (2), pages 1-10, 2002.

[96] I. Khamassi, M. Sayed-Mouchaweh, Moez Hammami, Khaled Ghedira. Discussion and Review on Evolving Data Streams and Concept Drift Adapting, Springer, Evolving Systems Journal, October 2016.

[97] I. Khamassi, M. Sayed-Mouchaweh, Moez Hammami, Khaled Ghedira. Self-Adaptive Windowing Approach for Handling Complex Concept Drift, Cognitive Computation, Springer, vol. 7, June 2015.

[98] M. Sayed-Mouchaweh. Learning from Data Streams in Dynamic Environments. Springer Briefs in Electrical and Computer Engineering, ISBN 978-3-319-25665-8, January 2016.

[99] M. Sayed-Mouchaweh and E. Lughofer. Learning in Non-Stationary Environments. Methods and Applications,New York:Springer, 2012.

[100] I. Zliobaite. Learning under Concept Drift: an Overview. Technical report, Vilnius University, 2009.

[101] H. Valizadegan and P. Tan. A Prototype-Driven Framework for Change Detection in Data Stream Classification. In: CIDM07: Proceedings of IEEE symposium on Computational Intelligence and Data Mining, pages 88-95,2007.

[102] L. Minku, A. White, and X. Yao. The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift. IEEE Transactions on Knowledge and Data Engineering, 99(1), 2009.

[103] P. Turney. Robust Classification with Context-Sensitive Features. In Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-93, pages 268-276, Edinburgh, Scotland: Gordon and Breach, 1993.

[104] I. Zliobaite, Adaptive Training Set Formation. Department of Computer Science, Vilnius University, 2010.

[105] J. Kim, P. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross. Immune System Approaches to Intrusion Detection-A Review. Natural Computing: an international journal, 6(4), pages 413-466, 2007.

[106] R. Bolton and D. Hand. Statistical Fraud Detection: A review. Statistical Science, 17(3), pages 235-255, 2002.

[107] J. Bakker, M. Pechenizkiy, I. Zliobaite, A. Ivannikov, and T. Karkkainen. Handling Outliers and Concept Drift in Online Mass Flow Prediction in cfb Boilers. In Proc. of the 3rd international workshop on Knowledge Discovery from Sensor Data (SensorKDD-09), pages 13-22. ACM, 2009.

[108] A. Pawling, N. Chawla, and G. Madey. Anomaly Detection in a Mobile Communication Network. Computational and Mathematical Organization Theory, 13(4), pages 407-422, 2007.

[109] D. Billsus and M. Pazzani. A Hybrid User Model for News Story Classification. In Proceedings of the 7th international conference on User Modelling, pages 99-108, Springer-Verlag, 1999.

[110] R. Klinkenberg and I. Renz. Adaptive Information Filtering: Learning Drifting Concepts. In Proceedings of AAAI-98/ICML-98 workshop Learning for Text Categorization, pages 33-40, 1998.

[111] N. Lathia, S. Hailes, and L. Capra. kNN CF: a Temporal Social Network. In RecSys'08: Proceedings of the 2008 ACM conf. on Recommender systems, pages 227-234. ACM, 2008.

[112] J. Ekanayake, J. Tappolet, H. Gall, and A. Bernstein. Tracking Concept Drift of Software Projects Using Defect Prediction Quality. In Proceedings of the 6th IEEE working conf. on Mining Software Repositories, pages 51-60. IEEE Computer Society, 2009.

[113] P. Kumar and V. Ravi. Bankruptcy Prediction in Banks and Firms via Statistical and Intelligent Techniques-A Review. European Journal of Operational Research, 180(1), pages 1-28, 2007.

[114] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic Integration of Classifiers for Handling Concept Drift. Information Fusion, 9(1), pages 56-68, 2008.

[115] R. Yampolskiy and V. Govindaraju. Direct and Indirect Human Computer Interaction based Biometrics. Journal of computers, 2(10), pages 76-88, 2007.

[116] M. Procopio, J. Mulligan, and G. Grudic. Learning Terrain Segmentation with Classifier Ensembles for Autonomous Robot Navigation in Unstructured Environments. Journal of Field Robotics, 26(2), pages 145-175, 2009.

[117] D. Anguita. Smart Adaptive Systems: State of the Art and Future Directions of Research. In Proceedings of the 1st European symposium on Intelligent Technologies, Hybrid Systems and Smart Adaptive Systems, EUNITE 2001, 2001.

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

[119] D. Charles, A. Kerr, M. McNeill, M. McAlister, M. Black, J. Kucklich, A. Moore, and K. Stringer. Player-Centred Game Design: Player Modelling and Adaptive Digital Games. In Digital Games Research Conference 2005, Selected Papers Publication, pages 285-298, 2005.

[120] M. Masud, J. Gao, L. Khan, J. Han, B. M. Thuraisingham. Integrating Novel Class Detection with Classification for Concept-Drifting Data Streams. In: ECML PKDD 2009. LNCS, (5782), pages 79-94. Springer, Heidelberg, 2009.

[121] E.J. Spinosa, A.P. de Leon, F. de Carvalho, J. Gama. Cluster-based Novel Concept Detection in Data Streams Applied to Intrusion Detection in Computer Networks. In: ACM SAC, pages 976-980, 2008.

[122] A.R. Webb. Statistical Pattern Recognition. 2nd edn. Wiley, New Jersey, 2002.

[123] L. Bhuvanagiri and S. Ganguly. Estimating Entropy over Data Streams. In: Azar Y., Erlebach T. (eds) Algorithms-ESA 2006. Lecture Notes in Computer Science, vol. 4168, Springer, Berlin, Heidelberg, 2006.

[124] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang. Data Streaming Algorithms for Estimating Entropy of Network Traffic. Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, pages 145-156, ACM SIGMETRICS, 2006.

[125] F. Keller, E. Muller, and K. Bohm. Estimating Mutual Information on Data Streams. In Proceedings of the 27th International Conference on Scientific and Statistical Database Management,page 3, ACM, 2015.