Universidad del País Vasco
Euskal Herriko Unibertsitatea

New Perspectives and Methods for Stream Learning in the Presence of Concept Drift

Author:
Jesús López Lobo

Supervisors:
Prof. Javier Del Ser
Prof. Miren Nekane Bilbao

Doctor of Philosophy

October, 2018

"The question of whether a computer can think is no more interesting than the question of whether a submarine can swim."

Edsger W. Dijkstra

UNIVERSITY OF THE BASQUE COUNTRY UPV/EHU

Abstract

Doctor of Philosophy

New Perspectives and Methods for Stream Learning in the Presence of Concept Drift

by Jesús López Lobo

Online Learning and Concept Drift are two of the hottest topics in the recent literature related to Machine Learning due to their relevance for the so-called *Big Data* paradigm, where nowadays we can find an increasing number of applications based on continuously produced training data, named as *data streams*. This thesis provides a complete overview of several related fields, and tackles some open challenges that have been identified in the very recent state of the art. Concretely, it presents an innovative way to generate artificial diversity in ensembles, a set of necessary adaptations and improvements for evolving spiking neural networks in order to be used in online learning scenarios, and finally, a drift detector based on this former brain-inspired algorithm. All these approaches constitute an innovative work aimed at presenting new perspectives and methods in the field.

Acknowledgements

Thanks firstly go to my supervisors Nekane and Javi, and especially to this latter man, who taught me to decipher AI and scrutinise the contents, who generously not only educated me in the art of research but also in pedagogy, who turned out to be a close mate too. This thesis would never have been completed without his continuous help on both my study and life. Thanks Javi!

To TECNALIA (OPTIMA group in ICT Division), the workplace in which I have been working for more than 11 years, and where I had the good fortune to get a research work. Especially to Isidoro Cirión, Iñigo Arizaga, Elena Urrutia, and Joseba Laka for providing me with the necessary time and resources, and for giving priority to this thesis work over other duties.

To my JRL mates, Ibai, Izaskun, Eneko, among others, always willing to provide me with a break away from the academic bubble in the Txalaparta restaurant. Especially to Ibai for being my research mate from the beginning of the PhD period, and also during my stay at New Zealand.

Thanks also for countless enjoyable experiences over my time in Auckland, New Zealand, where I had the opportunity to be Visiting Researcher at the Knowledge Engineering and Discovery Research Institute (KEDRI) at the Auckland University of Technology (AUT). The time I spent in Aotearoa there was an unforgettable experience for me and my family. Much gratitude is due to Professor Nikola Kasabov for his warm welcome, research support, and collaboration. Thanks KEDRI mates for all the stimulating debates, coffee breaks, life advice, and funny lunches, especially to Elisa who was a great support during my stay there, and turned into a close friend for me and for my family.

Last but not least, this thesis is for my Mum, my Dad, my wife Lucía, and my daughter Udane. Family is the one sure thing in life, and quite simply they put me where I am today.

Contents

List of Figures

List of Tables

List of Abbreviations

ADWIN	**AD**aptive sliding **WIN**dow
AI	**A**rtificial **I**ntelligence
AllKNN	**A**ll **KN**earest **N**eighbors
ARF	**Adaptive R**andom **F**orest
ASGP	**A**daptive **S**elf-**G**enerating **P**rototypes
AUC	**A**rea **U**nder the **C**urve
CNF	**C**onjunctive **N**ormal **F**orm
CNN	**C**ondensed **N**earest **N**eighbor
CUSUM	**C**umulative **S**um
DDM	**D**rift **D**etection **M**ethod
DOD	**D**egree **O**f **D**rift
DRED	**D**iversity gene**R**ator **E**volutionary technique with **D**ensity estimation
DRT	**D**ata **R**eduction **T**echnique
DT	**D**istance to the **D**rift
eSNN	**e**volving **S**piking **N**eural **N**etwork
ECDD	EWMA for **C**oncept **D**rift **D**etection
ECOS	**E**volving **C**onnectionist **S**ystems
EDIST2	Error **DIST**ance based approach for drift detection and monitoring 2
EDDM	**E**arly **D**rift **D**etection **M**ethod
ENN	**E**dited **N**earest **N**eighbor
EWMA	**E**xponential **W**eighted **M**oving **A**verage
FN	**F**alse **N**egatives
FP	**F**alse **P**ositives
GAN	**G**enerative **A**dversarial **N**etworks
GB	**G**igaByte
GNB	**G**aussian **N**aive **B**ayes
GRF	**G**aussian **R**eceptive **F**ield
HNBT	**H**oeffding **N**aive **B**ayes **T**rees
HT	**H**oeffding **T**rees
kNN	**k N**earest **N**eighbors
KDE	**K**ernel **D**ensity **E**stimation
LIF	**L**eaky **I**ntegrate-and-**F**ire

LML	**L**ifelong **M**achine **L**earning
LSTM	**L**ong **S**hort **T**erm **M**emory
MOT	**M**ulti-objective **O**ptimization **T**echnique
NSGA-II	**N**ondominated **S**orting **G**enetic **A**lgorithm II
OeSNN	**O**nline **e**volving **S**piking **N**eural **N**etwork
PG	**P**rototype **G**eneration
PHT	**P**age-**H**inkley **T**est
PL	**P**aired **L**earners
PS	**P**rototype **S**election
PSP	**P**ost-**S**ynaptic **P**otential
RAM	**R**andom **A**ccess **M**emory
RDDM	**R**eactive **D**rift **D**etection **M**ethod
RENN	**R**epeated **E**dited **N**earest **N**eighbor
RF	**R**andom **F**orest
RNN	**R**ecurrent **N**eural **N**etworks
ROC	**R**eceiver **O**perating **C**haracteristics
SGP	**S**elf-**G**enerating **P**rototypes
SGP2	**S**elf-**G**enerating **P**rototypes 2
SNN	**S**piking **N**eural **N**etwork
SSMA	**S**teady-**S**tate **M**emetic **A**lgorithm
STEPD	**S**tatistical **T**est of **E**qual **PD**proportions
SVM	**S**upport **V**ector **M**achines
TCNN	**T**omek **C**ondensed **N**earest **N**eighbor
TP	**T**rue **P**ositives

List of Symbols

Symbol	Description
K	Number of models in the ensemble
W	Window size
λ^{high}	High diversity λ
λ^{low}	Low diversity λ
p_{th}^{high}	High probability p
p_{th}^{low}	Low probability p
h	Bandwidth parameter
L	Number of classes (labels)
Pop	Population size
Gen	Number of generations
P_c	Cross-over probability
P_m	Random Mutation Probability
$N^{\boxplus}$	# of synthetic samples for labeling
$N^{\boxtimes}$	# of synthetic samples for performance estimation

Dedicated to Lucía and Udane. To Lucía, because of her support, understanding, patience, and empathy during this period of time; without her it would not have been possible to complete this challenge. To Udane, because without her knowledge or consent, I have stolen some of her playing time with me during the achievement of this goal.

Chapter 1

Introduction

This first chapter introduces the problems addressed in this thesis and gives an overview of subsequent chapters. Section 1.1 presents the problem of *online learning* and *concept drift*. In Section 1.2, we describe the motivation behind this Thesis. Section 1.3 summarizes a selection of the significant contributions derived from the research findings, and finally Sections 1.4 and 1.5 describe the publications resulting from this thesis and the content in each subsequent chapter respectively.

1.1 Online Learning and Concept Drift

Applications that generate huge amounts of data [1] in the form of fast streams from non-stationary environments, that is, those where the underlying phenomena change over time, are becoming increasingly prevalent. In this kind of environments the probability density function of the data-generating process may change over time, producing a *drift*. This causes that predictive models trained over these stream data become obsolete and do not adapt suitably to the new distribution. Specially in *online learning* scenarios, there is a pressing need for new algorithms that adapt to this change as fast as possible, while maintaining good performance scores. Examples of these applications include making inferences or predictions based on financial data, energy demand and climate data analysis, web usage or sensor network monitoring, and malware/spam detection, among others [2].

Learning in non-stationary environments requires adaptive or evolving approaches that can monitor and track the underlying changes, and adapt a model to accommodate those changes accordingly. Adaptation strategies can be applied proactively (active approaches or informed methods) detecting first the concept drift, and the model gets updated only when a drift is detected, or reactively (passive

approaches or blind methods) by updating the model continuously every time new data samples are received. Ensembles of learners are an extended solution to deal with the learning in non-stationary environments. The success of these ensembles lies in taking into consideration their complementarity and diversity [3] of its learners. Empirical evidence has shown that a good strategy should be based on maintaining highly diverse ensembles and use them shortly after the drift occurs in order to achieve good performance scores. To achieve a successful adaptation, additional mechanisms are also necessary for the ensemble when adapting to the new concept as soon as possible.

Besides, learning in an online mode imposes a set of restrictions [4] that every adaptive technique must meet (i.e. limited processing per arriving sample, limited amount of memory, etc.). Regarding the limited amount of memory, one of the most relevant requirements that we have to consider when we design an online learning system, data reduction techniques become a very useful tool to reduce our memory needs. They aim to obtain a representative training set with a lower size compared to the original one, yet with similar or even higher classification accuracy [5].

Finally, recent researches [6], [7] lead us to consider the temporal dependence that frequently real data streams exhibit. They suggest that a naive classifier considering the temporal dependence can outperform a lot of classifiers in the literature. They propose to pick ideas from time series analysis in order to adapt them to drift detection in presence of temporal dependence.

In this effort, we provide in this thesis a comprehensive state-of-the-art approaches as well as we identify the most relevant open challenges in the literature, while focusing on addressing three of them by providing innovative perspectives and methods.

1.2 Motivation

This thesis provides an overview of the current state of the *online learning* and *concept drift* fields. Besides, it identifies the current challenges of them, and addresses those of high relevance for the field because prominent authors highlighted in their very recent publications. By addressing these challenges we provide new perspectives and methods that take a step further beyond the state of the art. These addressed challenges will be explained in detail in Chapter 2.7. Now, we introduce briefly our motivations behind this thesis.

1.2.1 Diversity Generation Methods for Drift Adaptation

Diversity management and generation methods are an essential tool to achieve the required level of diversity in ensembles when applied to online learning scenarios [8], where only one sample at each time is available to train the classifiers. Here, handling with methods based on input manipulation (e.g. Online Bagging or Online Boosting), output manipulation (e.g. Class-Switching), base learner parameters manipulation, or heterogeneous base learners (e.g. Stacking) is primordial to encourage different levels of diversity in ensembles. With the added difficulty of having just one sample at each time, it should be also considered that there is theoretical and empirical evidence that the maximum diversity in ensembles is not usually achievable with the original data on which models are trained. Due to all these reasons, understanding how to manage (and produce) the diversity remains as an open topic in the field. In addition to the diversity generation in ensembles, we should also achieve at the same time a good classification performance. Therefore, we must find an equilibrium between both objectives, which entails a challenging paradigm.

1.2.2 Data Reduction Techniques for Drift Adaptation

The inherent storage limitation imposed to online learning approaches enforces researchers to derive novel methods to deal with a very reduced amount of samples. One of the requests from the research community [9] is the implementation of more algorithms based on data reduction techniques for drift adaptation. These techniques provide a representative training set with a lower size when compared to the original one, and with similar or even higher generalization capabilities, which may reduce the computational complexity of the learning algorithm. This benefit, coupled with the fact that evolving spiking neural networks (eSNNs) are one of the most promising techniques because their flexibility to easily incorporate new data into a classification model when new data are presented, makes them an attractive choice for massive data stream scenarios under severe computational restrictions.

1.2.3 Capturing Temporal Dependence for Drift Detection

Recently, several prominent studies [6], [7] have encouraged the research community to consider the temporal dependence, which frequently appears in real data streams, when building new algorithms

for drift detection. The inherent ability of the eSNNs to capture the temporal dependence of data, makes them a perfect candidate to act as a drift detector technique. Besides, all existing drift detectors utilize a base learner to analyze its performance score and detect drifts; this observation motivates to think about exploiting the structural changes of eSNNs to develop an embedded drift detector.

1.3 Thesis Contributions

In response to the research hypotheses postulated above, and addressing those selected challenges which have shown a wider impact on the stream learning field (see Sec. 2.7.2), this thesis elaborates on a number of significant contributions to the field of *online learning* and *concept drift*, which will be discussed in more detail in Chapter 6.1.

- In Challenge 1, we have addressed the field of drift adaptation, by creating a new technique (DRED) that strengthens any traditional diversity generation scheme (such as bagging, boosting, and class-switching), allowing them to achieve a better classification performance under a wide range of drift conditions. This Chapter contributes to the state of the art by:
  - using for the first time the Bhattacharyya distance to measure the diversity of an ensemble in data streams for *concept drift* scenarios.
  - developing the first technique that can be hybridized with any other diversity-inducing method to optimize the level of diversity and the classification performance in ensembles building for online learning scenarios in the presence of *concept drift*.
  - providing an innovative perspective on diversity generation, based on optimally labeling synthetic distances as measured by a quantitative metric of ensemble diversity.
- In Challenge 2, we have also addressed the field of drift adaptation, but this time by:
  - developing the first eSNN implementation that considers in a realistic and practical way the constraints imposed by the online learning scenarios, and

  - carrying out an analysis of the impact of applying data reduction techniques to eSNNs in order to limit the neurons repository size.

- Finally, in Challenge 3, we have addressed the field of drift detection by developing a novel drift detection algorithm based on capturing the temporal dependence of the data stream, taking advantage of the inherent evolving characteristic of the eSNNs to adapt to changes.

1.4 Publications during this Thesis

1.4.1 Related to this Thesis

The findings related to the above motivations and selected challenges have been published or submitted to relevant research journals and presented to international conferences.

Journal Publications

- Lobo, J. L., Del Ser, J., Bifet, A., Kasabov, N. "Spiking Neural Networks and Online Learning: An Overview and Perspectives". In: *IEEE Signal Processing Magazine* (2018). (Submitted)

  JCR: 7.451 (2017), Q1 (9/260), Engineering, Electrical and Electronic.

- Lobo, J. L., Laña, I., Del Ser, J., Bilbao, M. N., Kasabov, N. "Evolving Spiking Neural Networks for online learning over drifting data streams". In: *Neural Networks* 108 (2018), pp. 1–19. (Accepted)

  JCR: 7.197 (2017), Q1 (7/132), Computer Science and Artificial Intelligence.

- Lobo, J. L., Del Ser, J., Bilbao, M. N., Perfecto, C., Salcedo-Sanz, S. "DRED: An evolutionary diversity generation method for concept drift adaptation in online learning environments". In: *Applied Soft Computing* 68 (2018), pp. 693–709. (Accepted)

  JCR: 3.907 (2017), Q1 (17/132), Computer Science and Artificial Intelligence.

Conference Publications

- Lobo, J. L., Del Ser, J., Laña, I., Bilbao, M. N., Kasabov, N. "Drift Detection over Non-stationary Data Streams using Evolving Spiking Neural Networks". In: *International Symposium on Intelligent and Distributed Computing*. Springer. 2018. (Accepted)

- Lobo, J. L., Del Ser, J., Villar-Rodriguez, E., Bilbao, M. N., Salcedo-Sanz, S. "On the Creation of Diverse Ensembles for Non-stationary Environments Using Bio-inspired Heuristics". In: *International Conference on Harmony Search Algorithm*. Springer. 2017, pp. 67–77. (Published)

- Lobo, J. L., Del Ser, J., Bilbao, M. N., Laña, I., Salcedo-Sanz, S. "A Probabilistic Sample Matchmaking Strategy for Imbalanced Data Streams with Concept Drift". In: *International Symposium on Intelligent and Distributed Computing*. Springer. 2016, pp. 237–246. (Published)

1.4.2 Other Publications

The findings related to the projects carried out at Tecnalia Research and Innovation (where I work as Data Scientist) during my thesis period, or developed at the KEDRI department of the Auckland University of Technology (AUT), have been also published or submitted to relevant research journals and presented to international conferences.

Journal Publications

- Ibai Laña, Lobo, J. L., Capecci, E., Del Ser, J., Kasabov, N. "Adaptive Long-Term Traffic Forecasting with Evolving Spiking Neural Networks". In: *Transportation Research Part C: Emerging Technologies* (2018). (Submitted)

  JCR: 3.968 (2017), Q1 (6/35), Transportation Science and Technology.

- Capecci, E., Lobo, J. L., Ibai Laña, Espinosa-Ramos, J. I., Kasabov, N. "Modelling Gene Interaction Networks from Time-Series Gene Expression Data using Evolving Spiking Neural Networks". In: *Evolving Systems* (2018). (Accepted)

- Lobo, J. L., Santos, O. C., Boticario, J. G. "Identifying recommendation opportunities for computer-supported collaborative environments". In: *Expert Systems* 33.5 (2016), pp. 463–479. (Accepted)

  JCR: 1.180 (2016), Q3 (91/133), Computer Science and Artificial Intelligence.

Conference Publications

- Aróstegi, M., Torre-Bastida, A. I., Lobo, J. L., Bilbao, M. N., Del Ser, J. "Concept Tracking and Adaptation for Drifting Data Streams under Extreme Verification Latency". In: *International Symposium on Intelligent and Distributed Computing*. Springer. 2018. (Accepted)

- Ibai Laña, Capecci, E., Del Ser, J., Lobo, J. L., Kasabov, N. "Road Traffic Forecasting Using NeuCube and Dynamic Evolving Spiking Neural Networks". In: *International Symposium on Intelligent and Distributed Computing*. Springer. 2018. (Accepted)

- Landa-Torres, I., Lobo, J. L., Murua, I., Manjarres, D., Del Ser, J. "Multi-objective heuristics applied to robot task planning for inspection plants". In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2017, pp. 1621–1628. (Published)

- Lobo, J. L., Del Ser, J., De Simone, F., Presta, R., Colina, S., Moravek, Z. "Cognitive workload classification using eye-tracking and EEG data". In: *International Conference on Human-Computer Interaction in Aerospace*. ACM. 2016, pp. 16. (Published)

- Del Ser, J., Lobo, J. L., Villar-Rodriguez, E., Bilbao, M. N., Perfecto, C. "Community detection in graphs based on surprise maximization using firefly heuristics". In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2016, pp. 2233–2239. (Published)

1.5 Structure of the Thesis

This initial chapter has briefly introduced some preliminaries for subsequent chapters: online learning and concept drift. We have also described the motivation behind this thesis and summarized

the main contributions of this thesis. Finally, we have listed the publications resulting from this research work. Now, we detail the structure of this thesis work, which is visually organized in Figure 1.1.

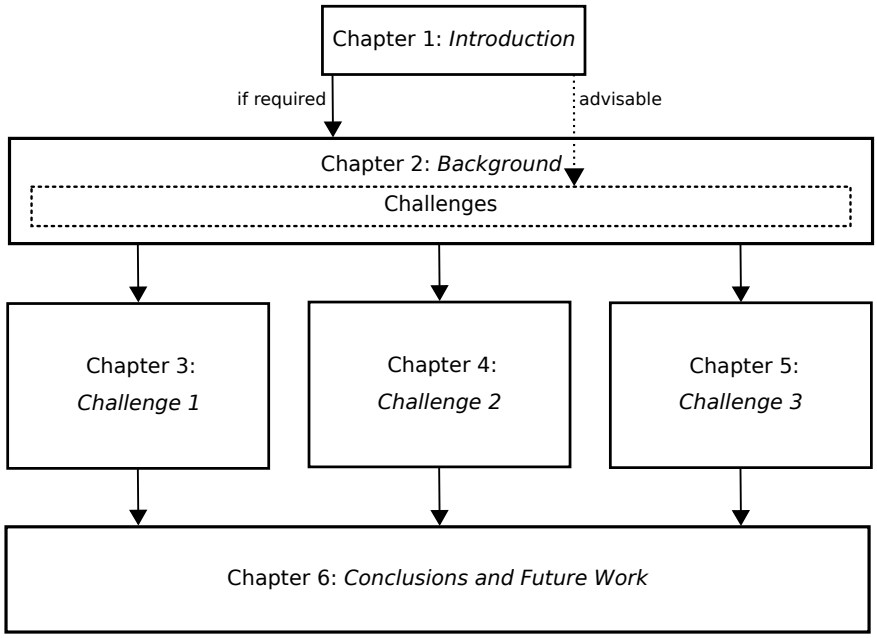


FIGURE 1.1: Structure of this thesis.

In Chapter 2, we review the literature related to this thesis and give an overview of subsequent chapters. Concretely, Section 2.1 introduces a complete taxonomy of methods for concept drift adaptation, presenting the most relevant concepts of the field. Section 2.2 goes into detail on the stream learning topics that are subsequently presented. In Section 2.3, we enter the problem of concept drift, and Section 2.4 is aimed at explaining the importance of base learners. Section 2.5 shows how ensembles are a key aspect on stream learning, whereas Section 2.6 reveals the different evaluation methods for stream learning scenarios. Section 2.7 brings to light the existing challenges in the state of the art, and highlights those which have been selected to be addressed by this thesis. Finally Section 2.8 summarizes the whole introductory chapter.

In Chapter 3, Section 3.1 introduces the work carried out in this challenge. Section 3.2 provides an overall insight on the different tools and techniques embedded in DRED, whereas Section 3.3 delves into the details of the proposed approach. Section 3.4 presents the experiments, while Section 3.5 discusses the performed numerical

experiments. And, finally, Section 3.6 provides a summary of the whole first challenge.

In Chapter 4, Section 4.1 introduces the work carried out in this challenge. Section 4.2 provides a general introduction to the evolving spiking neural networks and their relevance in online learning scenarios. Section 4.3 delves into the data reduction methods used in the proposed approach. Section 4.4 provides a detailed description of the proposed approach, while Section 4.5 presents the experimental setup designed to assess its performance. Section 4.6 presents and discusses the obtained results from such experiments and finally, Section 4.7 gives an overall sketch of the whole second challenge.

In Chapter 5, Section 5.1 introduces the work carried out in this challenge. Section 5.2 details the proposed approach. Section 5.3 presents the experimental setup designed to assess eSNN-DD performance. Section 5.4 presents and discusses the experimental results, and finally Section 5.5 sketches the third challenge.

Finally, Chapter 6 ends the thesis by summarizing its conclusions and by suggesting several future research lines.

Chapter 2

Background Material

2.1 Introduction

The concept of *Big Data* has been gaining progressive momentum during the last decade [1], due to the fact that we can collect data from almost any source and analyze it to find answers that enable cost and time reductions, new product developments, optimized offerings, or smart decision making, among others profits. From a data analytics perspective, *Big Data* has come to be defined by the four V's: Volume, Velocity, Veracity, and Variety. In what refers to Velocity, stream data are produced at rates faster than those that can be handled by traditional algorithms and systems. As exposed in the introductory chapter of this thesis, an increasing number of applications are based on training data continuously available, known as *data streams*, and applied to real scenarios [2], such as mobile phones, sensor networks, industrial process controls and intelligent user interfaces, among others. Recently, this field has grasped the interest of the research community in new approaches capable of dealing with these fast evolving information flows, which unleash a set of challenges when designing algorithms to be performed in real time.

The fields of *stream learning* and *concept drift* are vast. As a result, we can often find different confusing taxonomies trying to cover the fields entirely, which is not an easy task due to the fact that these fields are growing and changing constantly, and that can be tackled from different perspectives. This thesis tries to shed light on this issue by merging known embraced taxonomies such as [10] and [11], in order to produce a new one with some original ingredients (e.g. ensemble training methods, base learners, evaluation methods, and labelling). On the one hand, in [10] the taxonomy is distributed into modules that present adaptive learning systems as consisting of modular components, which can be permuted and combined with

each other. On the other hand, [11] defines several dimensions relevant to the applications facing concept drift. Figure 2.1 shows a global view of adaptive learning methods that comprises most of the current subfields and research lines of this field. This thesis will deal with some of these modules (e.g. Learning, Drift detection, Evaluation methods, and Memory). In what follows some related concepts will be described in detail in order to ease the understanding of the scope of the addressed challenges. Next sections of this chapter will show a detailed taxonomy for each of these challenges.

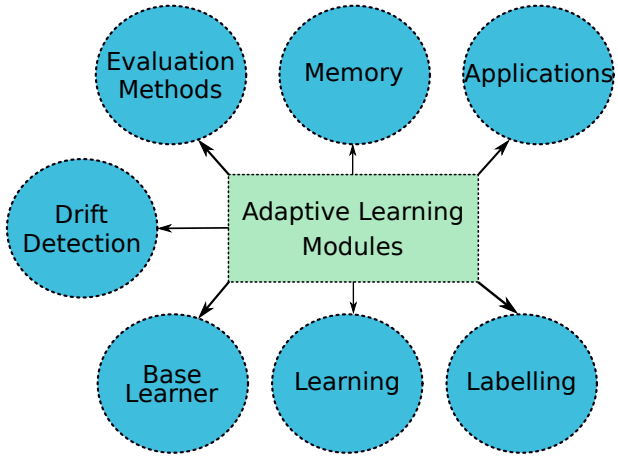


FIGURE 2.1: Taxonomy of methods for concept drift adaptation.

Before starting with the details of the taxonomy (Figure 2.1), we would like to briefly outline a very recent trend that frames the online learning in non-stationary environments field within the term *Lifelong Machine Learning (LML)*, or *Continuous Learning*, or *Continual Learning*. This new emerging paradigm [12], [13] focuses on developing versatile systems that accumulate and refine their knowledge over time, considering also other research fields such as *transfer learning*, *reinforcement learning*, *online learning*, *multi-task learning*, and *knowledge representation and maintenance*. *Transfer learning* is based on the idea of how to take models from one area and apply them to another. *Reinforcement learning* is a research field related to unsupervised machine learning where an AI model learns how to optimally interact with an environment (e.g. a game) entirely through trial and error. The *multi-task learning* field tries to build a model than can be applied to different problems, by learning from different sources of information. Finally, *knowledge representation and maintenance* aims at

designing different ways of representation for the knowledge flows of systems and applications. Thus, as we see the online learning and concept drift field can be also seen from this new perspective, where it is considered a subfield of a major taxonomy and shares research lines (e.g. forgetting mechanisms, incremental learning, sequential learning, knowledge retention, etc.) with other subfields.

Once mentioned the LML taxonomy, we start now describing the modules of Figure 2.1. Learning from data streams (see Section 2.2) covers all the problems of traditional batch learning, e.g. missing values, noisy data, outliers, and also comprises its own restrictions. The distribution modeling data captured by sensor networks, mobile phones, intelligent user interfaces, industrial machinery and others alike (Figure 2.2) is usually assumed to be stationary along time. However, in many real cases such an assumption does not hold, as the data source itself is subject to dynamic externalities that affect the stationarity of its produced data stream(s), e.g. seasonality, periodicity or sensor errors, among many others. As a result, possible patterns behind the produced data may change over time, either in the feature domain [14] (new features are captured, part of the existing predictors disappear, or their value range evolves), or in the class domain (new classes emerge from the data streams, or some of the existing ones fade along time). As we will define formally in Section 2.3.1, this paradigm is what the literature has coined as *concept drift*, where the term *concept* refers to a stationary distribution relating a group of features to a set of classes.

When the goal is to infer the aforementioned class patterns from data through online learning, incremental models trained over drifting streams become obsolete when transitioning from one concept to another. Consequently, they do not adapt appropriately to the new emerged data distribution, unless they are modified to handle efficiently this unwanted effect. In order to minimize the impact of concept drift on the performance of predictive models, recent studies [6], [15], [16] have been focused on the development of efficient techniques for continuous *drift adaptation* (Section 2.3.2) or, alternatively, in the incorporation of *drift detection techniques* (Section 2.3.3) and concept forgetting mechanisms [17] (see Section 2.5.3). Indeed, online learning in the presence of concept drift has been a very hot topic during the last few years [2], [10], [15], and still remains under active debate in the community [6] due to the fact that there are many relevant open challenges that remain unsolved [7], [14], [18], [19] (Section 2.7).

Most online learning techniques dealing with concept drift are

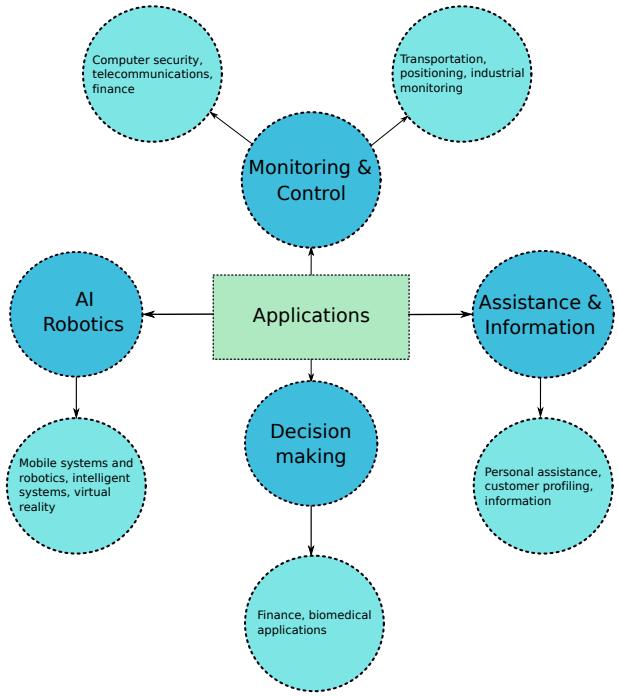


FIGURE 2.2: Applications module of the taxonomy for concept drift adaptation.

based on *ensemble building* (Section 2.5), which has been widely used to improve the accuracy (Section 2.6) of single classifiers in online and incremental learning [18]. Interestingly for the scope of this thesis, *diversity* (Section 2.5.2) among the constituent learners in ensemble models has been shown to be crucial when dealing with concept drifts [3]. This former study provides empirical evidence that the diversity plays an important role before and after a concept drift occurs: before the drift, ensembles with less diversity obtain better test errors, while shortly after the drift more diverse ensembles use to score lower test error rates. Their difference in terms of test error performance when compared to lower diverse ensembles is usually more significant when the severity of the concept change is higher. Therefore, it is a good strategy to maintain highly diverse ensembles and utilize them shortly after the drift (independently from the type of drift) to obtain good performance scores [3]. But this is not sufficient to achieve adaptation, as additional mechanisms are also necessary for the ensemble to adapt to the new concept as soon as possible. Thereby, a well-designed approach which maintains more than one ensemble with different diversity levels should be able to

converge in the absence of drifts and get lower test error soon after a drift [8].

Bearing in mind the necessity for adapting to the new concept but also remembering the old one, it is important to highlight the importance of the *stability-plasticity dilemma* [20]. As such, this principle refers to the ability of a model to retain acquired knowledge or to learn new concepts, but not being able to do both equally well at the same time. While stability entails accumulating knowledge of the old concept, plasticity requires forgetting some or all of the old acquired knowledge in order to learn the new upcoming concept as fast as possible. Due to this need for adapting the learning model to the drift, it is crucial to select a proper *forgetting mechanism* [21] (Section 2.5.3).

Most algorithms should take the above *stability-plasticity dilemma* as a reference when deciding which models (in case of ensembles) should be kept so as to predict the class for the next stream data. This dilemma results in a trade-off between the amount of diversity in the ensemble and the necessity of maintaining a good accuracy. Furthermore, the literature usually states that lower accuracy should correspond to higher diversity. However, [22] showed that the relationship between accuracy and diversity is not straightforward, i.e. a lower classification accuracy of ensemble members does not correspond to a higher diversity level. The study in [3] considered this issue and carried out a detailed research on this dilemma, concluding that shortly after the drift, more diverse ensembles are always among the best test errors, and the difference in the test error in comparison to lower diverse ensembles is usually more significant when severity is higher. Due to the noted importance of this trade-off along time, there is a latent need for novel mechanisms to optimally balance the diversity in ensemble learning. Likewise, as mentioned in Section 2.2, the inherent storage limitation to online learning approaches enforces the research community to derive novel methods to deal with a very reduced amount of samples, and making use of synthetic samples generation (Section 2.5.2).

Next subsections provide further details on the topics of this thesis in order to achieve a better comprehension of the challenges addressed in subsequent chapters.

2.2 Learning from Data Streams

As in the traditional machine learning field, in learning from data streams there is also the choice of a learning modality, such as supervised, unsupervised, or semi-supervised [23], [24] (Figure 2.3). Secondly, the rate at which data arrive should be also taken into account (Figures 2.4 and 2.5). The most common specific case of learning from streams is *classification*, where a set of samples is distributed into classes according to their relations. Given a set of samples (training data set), each sample is characterized by $(\mathbf{x}, y)$, where $\mathbf{x}$ is the attribute set (predictor, independent variable, input) and y is the class label (response, dependent variable, output). Then, the classification task consists of mapping each unlabeled sample $\mathbf{x}$ into one of the predefined class labels y. In *classification* data sets are therefore assumed to contain all information necessary to learn the relevant concepts belonging to the underlying generating function. Models derived from this case, however, have proven to be unrealistic for many real-world scenarios, e.g., intrusion detection, spam detection, fraud detection, loan recommendation, climate data analysis or long term epidemiological studies. Instead of all training data being available from the begining, data are often received over time in streams of samples or batches.

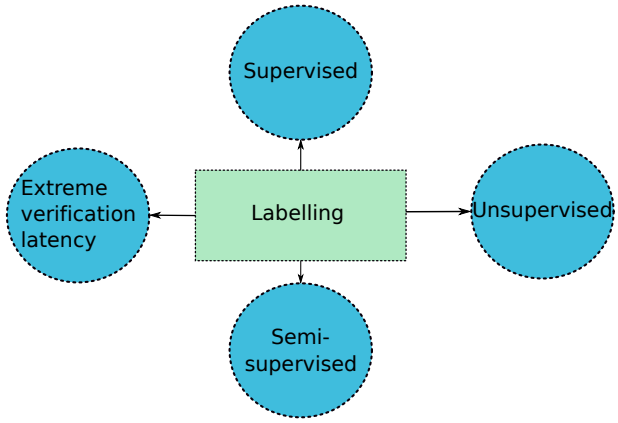


FIGURE 2.3: Labelling module of the taxonomy for concept drift adaptation.

Thereby, data may arrive in batches or chunks of data (*batch learning*) or in an online manner, i.e., one single sample at a time (*online learning*). In the case of online learning, only a single sample is provided to the learning algorithm at every time instant, which

incrementally updates with each new sample. However, in batch learning an entirely accessible group of samples (batch) is provided, and the learning algorithm is allowed to scan the batch before building/updating the model (see Figure 2.4).

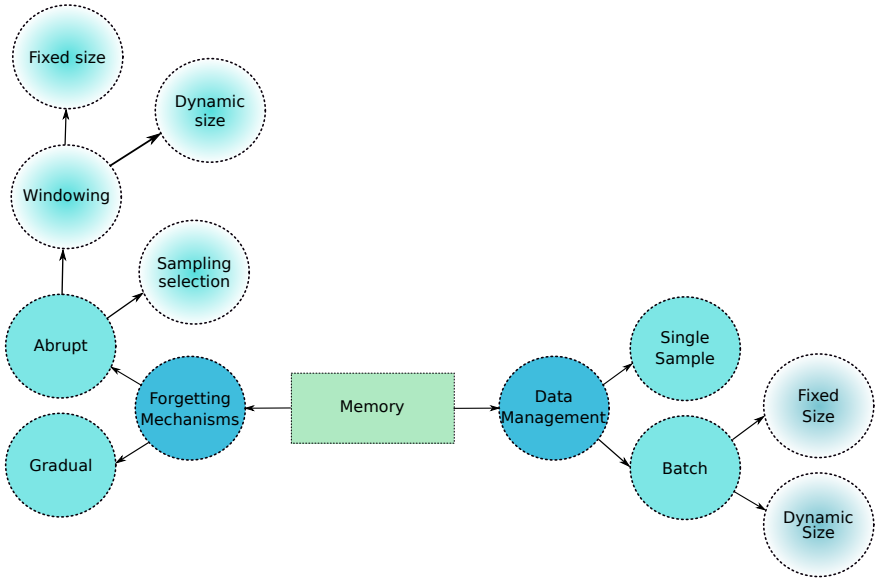


FIGURE 2.4: Memory module of the taxonomy for concept drift adaptation.

A data stream environment has different constraints [4] from the traditional batch learning setting:

- Each sample must be processed only once *on arrival*. Models must be able to process samples sequentially accordingly to their arrival. Although there is no specification of the number of buffered samples for a restricted amount of time, the learning algorithm must not put the memory space and processing time restrictions at risk.

- The processing time of each sample must be small and constant not to exceed the ratio in which new samples arrive. If the processing time exceeds this restriction, newly samples should be discarded or stored until the process become paralyzed.

- The algorithm should use only a preallocated amount of main memory, which is assumed to be finite. Its usage must be optimized, namely, models must be designed according to the hardware capacity.

- A valid model must be available at every scan of the streams of data.
- The algorithm must produce a model that is equivalent to the one that would be produced by a batch processing algorithm.

The evaluation procedure of the learning algorithm is determined by the set of samples used for training and testing, and the question raised here is how to build a picture of accuracy over time. One of the most used schemes is *test-then-train* (see Section 2.6.1), where each individual sample is used to test the model before it is used for training, and then the accuracy can be incrementally updated. This scheme has the advantage that can be applied when memory is restricted and there is no holdout set for testing, get the most out of the available data set.

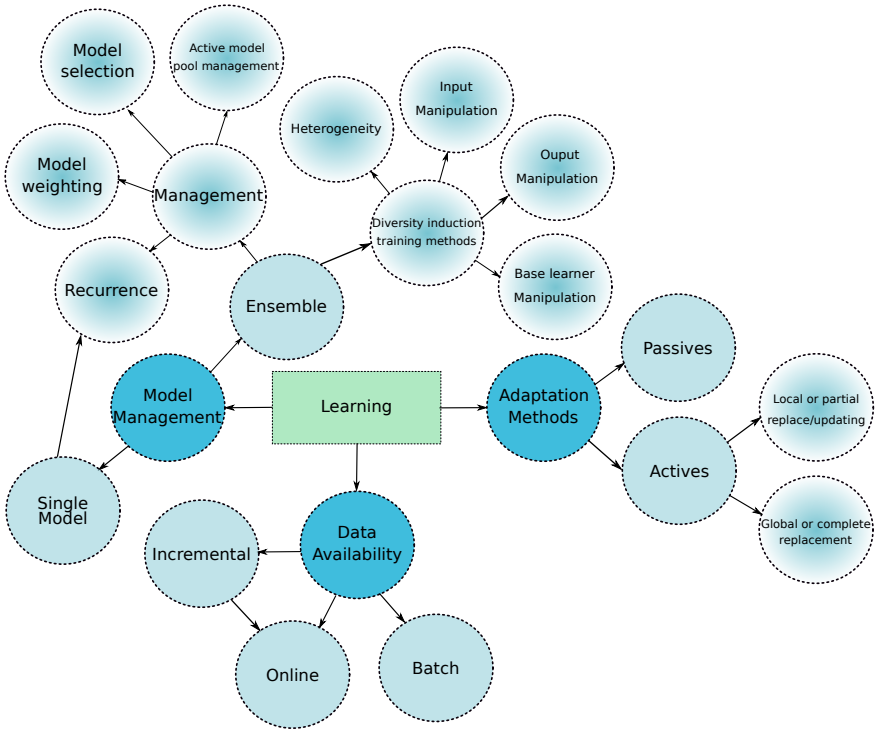


FIGURE 2.5: Learning module of the taxonomy for concept drift adaptation.

2.3 Concept Drift

2.3.1 Definition

As it has been argued, in dynamically changing environments where non-stationarity is present, the underlying distribution of streaming data may change over time. This change can be usually found in the feature domain [14] (new features are captured, part of the existing predictors disappear, or their value range evolves), or in the class domain (new classes emerge from the data streams, or some of the existing ones fade along time).

More formally, concept drift between time step ts_0 and ts_1 can be defined as:

$$\exists X\colon p_{ts_0}(\mathbf{x}, y) \neq p_{ts_1}(\mathbf{x}, y), \tag{2.1}$$

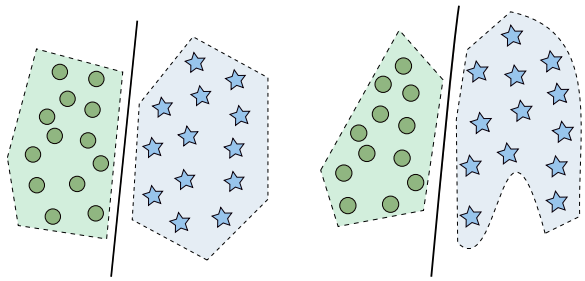
where $p_{ts_0}(\mathbf{x}, y)$ and $p_{ts_1}(\mathbf{x}, y)$ are the joint probability distributions at time step ts_0 and ts_1, respectively, between the input variables $\mathbf{x}$ that conform a data sample and the target variable y. A change in the relation between $\mathbf{x}$ and y can be represented as:

- the prior probabilities of classes $p(y)$ may change;
- the class conditional probabilities $p(\mathbf{x} \mid y)$ may change; and
- as a result, the posterior probabilities of classes $p(y|\mathbf{x})$ may change affecting the prediction.

For predictive tasks only the changes that affect the prediction decision need adaptation, and as Figure 2.6 depicts we can distinguish two types of drifts considering the reason of change:

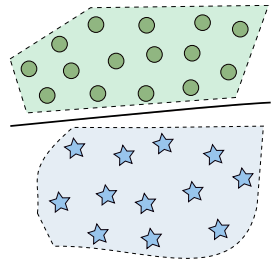
- *Real* drift, when the posterior probabilities of classes $p(y|\mathbf{x})$ vary over time independently from variations in $p(\mathbf{x})$, and may have an impact on unconditional probability density functions; and
- *Virtual* drift, when the distribution of the incoming data $p(X)$ changes without affecting the posterior probabilities of classes $p(y|)$, but affects only the conditional probability density functions.

Concept drift can be also categorized in terms of speed and severity [25], [26]. On the one hand, in the case of speed, an *abrupt* drift may occur when a change happens suddenly between two classification contexts, whereas a *gradual* drift represents the case when



(a) Original concept

(b) Virtual concept: $p(\mathbf{x}|y)$ changes



(c) Real concept: $p(y|\mathbf{x})$ changes

FIGURE 2.6: Types of concept drift depending on the impact on learned classification boundaries.

dealing with a smooth transition between two concepts. When there are several intermediate concepts in between the old and the new concept, the change is *incremental*. Likewise, if previously known concepts reoccur after some time, it is considered as a *recurrent* drift. Finally, a challenge may emerge when an outlier (blip) can be mixed with concept drift; in this case no adaptation is needed because it is a temporary event that does not affect the future data and thus the subsequent learning of the algorithm. Figure 2.7 shows these types of drifts.

On the other hand, severity can be regarded as the amount of changes that a new concept causes; therefore, a measure of severity can be computed as the percentage of the input space whose target class has changed after the drift [8].

Finally, it is worth mentioning that many efforts have been devoted to the characterization of concept drift [6], [16]. However, it is still an open issue in the state of the art due to the complexity of characterizing manifold types of data changes over time [18].

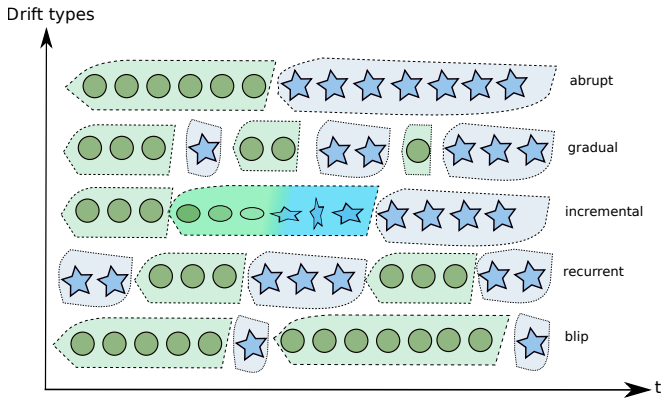


FIGURE 2.7: Types of drift according to severity and speed of changes, and noisy blips. Here the stars and circles represent the prevailing concept at every time instant.

2.3.2 Adaptation to the Drift

The adaptation to the drift (Figure 2.5) can be carried out proactively by first detecting concept drift, so that only the model gets updated when a drift is detected (*active* approaches or *informed* methods), or updating the model continuously every time new data samples are received (*passive* approaches or *blind* methods).

In monitoring and control applications, it is primordial to detect anomalous activities and out-of-control behaviors, and this problem is formulated as a detection task where the drift needs to be detected. Here, *active/informed* mehtods are recommended. They are reactive; when a drift is detected they can either relearn the model from scratch or update it using a recent selection of data (data window).

Regarding the *passive/blind* mehtods, they implicitly adapt the learner to the current concept at regular intervals without any drift detection mechanism. They discard old concepts at a constant speed independently of whether changes have happened or not. *Passive* and *blind* methods are recommended for handling gradual continuous drifts where the dissimilarity between consecutive data sources is not quite relevant to trigger a change. Three main mechanisms can be noted in the literature:

- Windowing: a sliding window over the last data samples is used as the exclusive training set for the learning algorithm [27]–[30].

- Weighting: all available samples are considered, but suitably weighted according to their age or relevance in terms of classification accuracy [31], [32].
- Reservoir sampling: a subset of samples is selected for training, and randomly drawn examples from within the reservoir are discarded upon receiving new data [33]–[35].

Several model approaches have been presented in the literature to deal with concept drift adaptation [26], [36]. Some of them base their strategy on adapting the internal structure of their classifiers [8], [37], while others resort to an ensemble of models [38] (see Section 2.5) to which techniques to induce diversity (see Section 2.5.2) are applied [39]. When focusing on drift detection approaches, they can be used as a module inside other classifiers to adapt either the internal structure of the classifiers or the number of classifiers within the ensemble. All of them usually use a specific classifier called *base learner*, and analyze its performance score (in general, accuracy or error rate) to indicate whether a drift has occurred. This base learner is trained on the current sample by means of an incremental process repeated for each incoming sample from the stream. However, the process of learning under concept drift events not only consists of updating the models but also forgetting the old data, as we will explain in Section 2.5.3.

2.3.3 Drift Detection

Drift detection mechanisms quantitatively characterize concept drift events by identifying change points or small-sized periods of time (windows) during which these changes may occur. Concept drift detectors are methods that can detect data distributions changes based on information about a base learner performance or the incoming data. Such changes usually trigger the need for updating, replacing or retraining the model (or the ensemble).

Drift detectors may return not only signals about drift ocurrence, but also warning signals, which are usually conceived as the moment when a change is suspected and a new training set representing the new concept should start being collected. The idea of drift detection is presented in Figure 2.9. Drift detection is not a trivial task because, on the one hand, sufficiently fast drift detection should be ensured to quickly replace the outdated model and to reduce the restoration time (transient to the stable period). On the other hand, it is not convenient to have too many false alarms (there is no real drift

in the stream), because the successive application of drift handling techniques could be counterproductive. Here a challenge emerges because of this mentioned trade-off between different detection metrics (see Section 2.6.2): a drift detector can be tuned to decrease the detection delay, but this may lead to a higher number of false alarms. In light of that, [40] have recently used an assessment methodology that resembles the Receiver Operating Characteristics (ROC) curve, but used to evaluate drift detection methods rather than classifiers, plotting the number of false alarms versus the drift detection delay for all drift detectors, using several different parameter configurations.

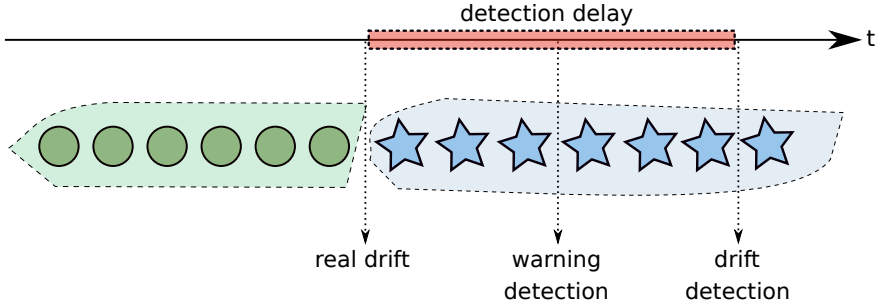


FIGURE 2.8: Drift detection example.

Drift detectors usually work reactively, that is, they act after the occurrence of the drift and model error since they depend on the actual class labels of the input patterns. Drift detection mechanism are generally composed of two modules: the loss estimation and the change detection. The loss estimation module tracks the performance of the learning algorithm and eventually sends information to the change detection module to update the model if necessary (Figure 2.9).

Regarding the loss estimation module, this can be based on a *model dependent* strategy (when the module is based on, for example, the leave-one-out error estimation of an algorithm over a window of samples, e.g. SVM [41]), or on a *model independent* strategy. In the latter case, there are two common approaches: 1) sliding windows, where two sliding windows are compared: a short window containing the most recent information, and a large window, used as reference, containing a larger set of recent data including the data in the short window; and 2) fading factors, where the oldest information becomes the least important, operating like a smooth forgetting mechanism. In the sliding windows approach, the short window is

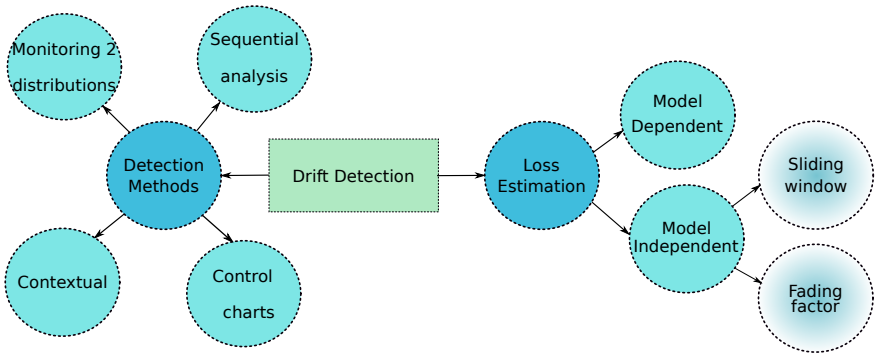


FIGURE 2.9: Drift detection module of the taxonomy for concept drift adaptation.

more reactive while the large window is more conservative, while in the fading factors approach, a smaller fading factor will detect drifts earlier than larger ones.

In [10] detection mechanisms for online learning are categorized in three different categories. To begin with, *sequential analysis* methods are based on the Sequential Probability Ratio Test [42], with the Cumulative Sum (CUSUM) and Page-Hinkley Test (PHT) [43] as their most representative examples. On the other hand, *statistical process control* is a technique that monitors and controls the evolution of the learning process, advancing on similar ideas to the seminal Exponentially Weighted Moving Average (EWMA) approach [44]. Probably Drift Detection Method (DDM) is the most well known representative approach, which assumes that classifier error has to decrease as more training examples are received [29]. Early Drift Detection Method (EDDM) is a modification of DDM to improve the detection of gradual drifts, where the same idea of warning and drift levels is realized with a new proposal of comparing distances of error rates. Finally, *monitoring the distribution* of data on two different windows has lately emerged as one of the most popular drift detection approaches: such a comparison is carried out by using statistical tests to assess the null hypothesis that the distributions in both are equivalent in terms of a certain statistic (e.g. equal medians). If the null hypothesis is rejected, a drift is assumed to be occurring at the start of the detection window. The ADaptive sliding WINdow (ADWIN) [27] is one of the most relevant examples of this last category. In contrast to three different categories mentioned before, different approaches have recently emerged. In [45] rather than

measuring the actual case distribution, they introduce a new competence model that detects differences through changes in competence. And more recently, the work presented in [36] is based on computing multiple model explanations over time and the observation of the magnitudes of their changes.

Once presented the traditional detection mechanisms based on a supervised point of view, it is worth mentioning those techniques which are based on unsupervised approaches. Methods based on unsupervised indicators are useful for detecting changes when the prediction feedback is delayed, which can be of good interest for many real world applications where data are unlabeled. Moreover, they can be of good interest for handling virtual concept drift, as it does not affect the decision boundaries. These methods can be based on:

- Similarity in time: how does data distribution evolve from a time stamp to another?
- Similarity in space: how does data distribution evolve according to the feature space?
- Model Complexity Measure is based on monitoring the structure and/or the parameters of the model
- Others like [46] estimates classifier performance (accuracy) using only the input features of samples from the data stream, basing its method on the degree of overlap between the output certainty distributions of a classifier

Unsupervised approaches are often performed with statistical tests that check whether a current window of data comes from the same distribution as the reference data. Evidently, not all statistical tests are suited for drift detection; the most recommended are non-parametric such as Conjunctive Normal Form (CNF) Density Estimation test [47], the multivariate version of the Wald-Wolfowitz test [48], and two-sample Kolmogorov-Smirnov test, Wilcoxon rank sum test, two-sample t-test [49]. Others approaches are based on semi-supervised indicators, which are useful for detecting concept drift in data streams with scarcely labeled samples.

This being said, the literature has been certainly rich in what refers to comparative studies of wide range of detectors in data sets with different drift characteristics and by using diverse performance metrics. The benchmark in [26] compares some of the most utilized

detectors in the literature, such as DDM [29], EDDM [50], PHT [43], ADWIN [27], Paired Learners (PL) [51], EWMA for Concept Drift Detection (ECDD) [44], Degree Of Drift (DOF) [52] and Statistical Test of Equal Proportions (STEPD) [53]. In [54] a new approach called Reactive Drift Detection Method (RDDM) is proposed and compared to DDM, EDDM and STEPD in several synthetic and real streaming scenarios.

Recently, and unlike the reactive drift detection methods mentioned before, the work in [55] has presented a new approach that works proactively, DetectA, in which after grouping the data, the means and covariance matrices of the previous and current blocks are compared by using statistical tests. If this difference exceeds a certain threshold, drift is signalled.

2.4 Base Learners

The decision of the base learner according to the classification problem is relevant in order to obtain an accurate ensemble. The majority of ensemble construction methods are designed to work with almost any base learner [56], [57], but regarding the ensemble design for data stream learning, the diversity induction strategy (Section 2.5.2) must be considered. For example (Figure 2.10), if the selected strategy is bagging (input manipulation), by which diversity between classifiers is based on training them on different samples, then unstable base learners (those which provide significantly different models despite being trained with similar samples, e.g., Decision Trees) are preferred over stable base learners (those which must be trained on a large set of different samples to differ from one another, e.g., Naive Bayes). Hoeffding Trees [58] are one of the most common base learner since they are not only unstable but also incremental, which makes them ideal for dealing with concept drifts. Other common options are perceptrons and multilayer perceptrons.

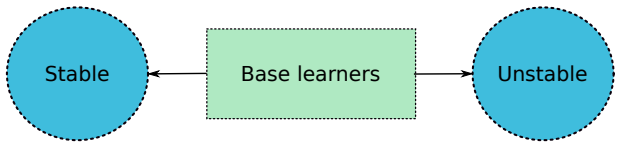


FIGURE 2.10: Base learners module of the taxonomy for concept drift adaptation.

It is worth mentioning that although unstable base learners are preferable to achieve the required level of diversity in the ensemble

for stream learning, it is also still possible to do it by using stable base learners as long as the diversity induction strategy allows for it. Therefore, the final strategy is a combination between the stability of the base learners and the diversity induction force of the strategy, considering the underlying diversity allowed by data.

2.5 Ensembles in Stream Learning

2.5.1 Why and How

The aim of an ensemble method is to combine the predictions of several base learners built with a given learning algorithm in order to improve generalizability and robustness over a single base learner. Ensembles are also an extended and accepted solution to decompose a complex and difficult learning problem into easier sub-problems. The success of ensembles lies in considering the complementarity and diversity (Section 2.5.2) of its base learners. The *no free lunch theorem* states [59], it is not possible to find a single classifier appropriate for all tasks because each algorithm has its own domain of competence. Besides, the selection of classifiers for the ensemble is one of the most relevant key factors in order to achieve a good performance [60]. They should be characterized by high diversity and accuracy [61]. It is generally agreed that both the accuracy and diversity of classifiers are key ingredients for improving the performance of ensembles.

There are different ways of combining the outputs of each base learner in the ensemble to produce a unique output in classification problems. Voting and averaging are two of the easiest ensemble methods. In majority voting (depicted in Figure 2.11), every model makes a prediction for each test samples and the final output prediction is the one that receives more than half of the votes. If none of the predictions get more than half of the votes, we may say that the ensemble method could not make a stable prediction for this sample. Unlike majority voting, where each model votes equally in the predicted outcome, we can increase the importance of one or more models. In weighted voting (Figure 2.12) the prediction of the best models counts in multiple times.

In the case of data streams, classifier ensembles are considered as a very attractive approach due to their natural way of adaptation to changes in data distribution, e.g. by adding new classifiers trained on recent data and discarding those which represent old concepts (see Section 2.5.3). The interest in building ensembles for stream

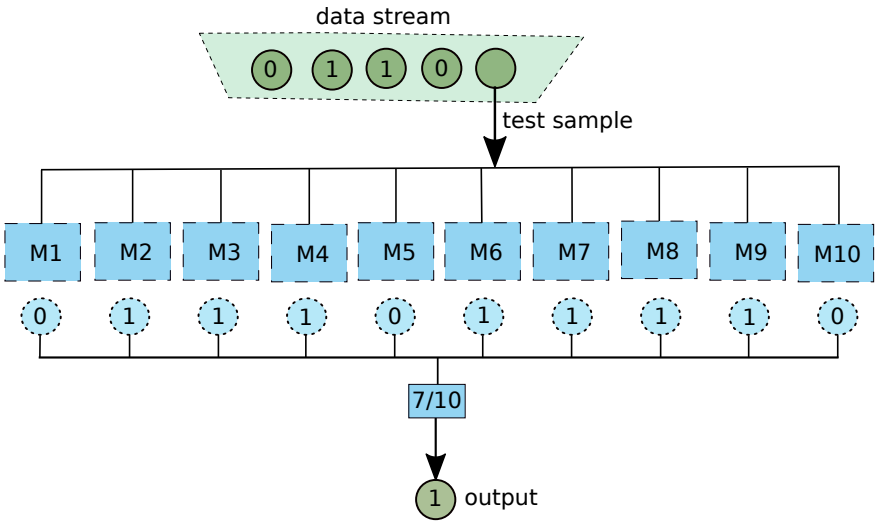


FIGURE 2.11: Majority voting strategy for ensemble building in a binary classification problem.

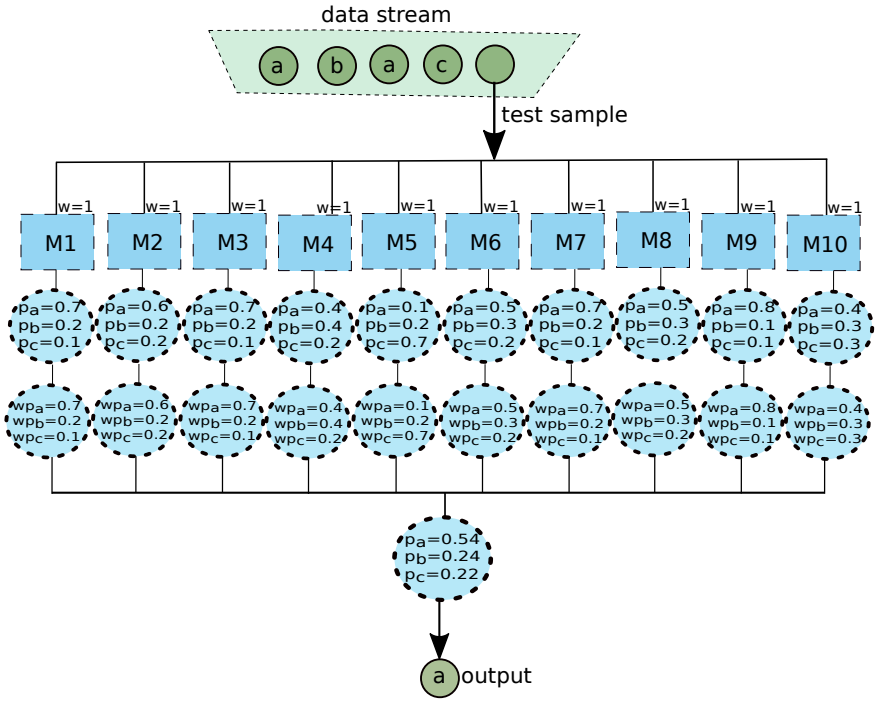


FIGURE 2.12: Majority weighted voting strategy for ensemble building in a classification problem with 3 classes.

learning is widely known [62]–[65], but continues gathering attention from the research community in the last few years [18], [66].

Regarding the ensemble size choice, many algorithms use a fixed size of ensemble. In other methods, a pruning mechanism is introduced to selectively reduce the number of ensemble members. But none of those methods solved the problem on how to decide if the ensemble component should be added until [67]. In this work, the authors contribute with two theorems which allow to determine if by adding an additional member to the ensemble we will obtain an increase of accuracy of the whole ensemble for the entire data stream.

2.5.2 Diversity

Diversity among the constituent learners of ensemble models has been shown to be crucial when dealing with concept drifts. In evolving scenarios, diversity may handle concept drifts by having an ensemble of varying base learners: one may be able to be prepared for the incoming drift, so that at least one of the base learners will have a decision boundary that can be quickly adapted to the new concept.

The findings reported in [3] provide empirical evidence that the diversity plays an important role before and after a drift occurs: before the drift, ensembles with less diversity obtain better test errors, while shortly after the drift more diverse ensembles use to score lower test error rates. Their difference in terms of test error performance when compared to lower diverse ensembles is usually more significant when the severity of the concept change is higher. Therefore, it is a good strategy to maintain highly diverse ensembles and utilize them shortly after the drift (independently from the type of drift) to obtain good performance scores. But this is not sufficient to achieve adaptation, as additional mechanisms are also necessary for the ensemble to adapt to the new concept as soon as possible. Thereby, a well-designed approach which maintains more than one ensemble with different diversity levels should be able to converge in the absence of drifts and get lower test error soon after a drift. This is the rationale for Diversity for Dealing with Drifts (DDD), the approach contributed in [8] that leverages this empirically validated observation. However, as already claimed in [22], diversity does not translate directly into accuracy.

Methods

Although high diversity may not directly indicate high accuracy [22], measuring diversity can be useful to analyze the effectiveness of diversity-inducing methods [68]. In [7] the authors proposed several strategies to induce diversity (Figure 2.5):

- *Input manipulation*, namely, training classifiers with different samples (different chunks of data or with different subsets of features). The *Online Bagging* and *Online Boosting* methods proposed in [69] are two of the most used input manipulation strategies to promote diversity in online learning ensembles. In the *Online Bagging* method the level of diversity in the ensemble varies by presenting a training example a specific number of times to every member, whereas in the *Online Boosting* method, based on the batch boosting algorithm AdaBoost [70], the weight of the misclassified examples is increased when presented to the next base learner; otherwise it is decreased. In terms of dependency (the training of one classifier may depend on the output of other classifiers), *Online Bagging* is often preferred over *Online Boosting* [71]. The drawback of *Online Boosting* is that may lead to overfitting. Moreover, in a stream learning environment it is not clear how to train classifiers in a sequential way. Then, training classifiers independently with *Online Bagging* is deemed easier to implement with better results [72], and allows training classifiers in parallel to cope with Big Data streams [73].

- *Output manipulation*, where the classification problem can be decomposed into smaller, potentially easier, problems. Afterwards, each problem can be mapped to a single classifier, and these classifiers can be diverse, since they interpret the hypothesis space differently. *Class-switching*, as a possible output manipulation strategy, hinges on using original training samples with randomized class labels, where the class label of each example is switched according to a probabilistic threshold,

- *Base learner manipulation*, where the modification of the characteristics (parameters) of each classifier leads to build more diverse ensembles. Even with the same training data, the predictions of the base learners may vary with different parameter sets, which is also subject to the sensitivity of the learning algorithm to such a variability.

- *Heterogeneous base learners (stacking)*, where the ensemble is formed by different classifiers based on different machine learning techniques (not only varying in terms of their parametrization).

In online learning, the presence of concept drift makes diversity requirements change, thus diversity-inducing methods should change accordingly. Many online ensemble approaches based on an input manipulation strategy use some fixed values (e.g. value of λ in a $Poisson(\lambda)$ distribution to produce a specific number of times for the training of every member of the ensemble) to encourage diversity during the stream mining process [8], [69]. The performance of those approaches, which follow a base learner manipulation strategy, suffer from preset parameters configuration. Even the combination strategy (voting) of the base learners predictions might be outdated. In these cases, these parameter sets may no longer be sufficiently good after a drift occurs (especially in case of parameter-sensitive techniques, like Support Vector Machines or Neural Networks). Therefore, *self-configuring* (or self-tuning or self-adapting) streaming ensemble systems may lead to increase classification performance.

Other approaches evinced that it is possible to combine two or more strategies (e.g. HEFT-Stream in [74]). In [75], a base learner manipulation strategy was carried out by using multiple neural networks with different topologies or with the same topology but starting with different weights at the first layer. In [71], the authors proposed the Adaptive Size Hoeffding Trees (ASHT) Bagging algorithm, based on the intuition that smaller decision trees are capable of a rapid adaptation to drifts, whereas bigger decision trees are more useful during stable periods. Therefore, a strategy that mixes both yield different ensemble members, and may also contribute to drift recovery.

2.5.3 Forgetting Mechanisms

As it has mentioned, while stability entails accumulating knowledge regarding the supposedly stationary underlying concept, plasticity requires forgetting some or all of the old acquired knowledge in order to learn the new upcoming concept. Due to the assumed diversity of the base learners in the ensemble, it is expected that at any time some of them are ready to take over and adapt to the changes of the environment. This diversity depends on the information kept by each base learner and on the control of which learner is selected

to be part of the ensemble. In addition, the way the votes of the base learners are combined participates also in the overall solution to the *stability-plasticity* dilemma. Each of these three factors (the memory of each expert, the control of the population of experts in the ensemble, and the weight attached to each base learner in the final decision) plays a role in the way past data is taken into account, yielding what can be referred to as the forgetting strategy [21].

At each time, a set of samples defines a window of the information considered for learning. Based on this window, the forgetting mechanims can be divided into two approaches: abrupt and gradual (see Figure 2.4). On the one hand, in abrupt forgetting a given sample is either inside or outside the training window. The sliding windows models can be sequence-based (when the size of the window is characterized by the number of samples) or timestamp-based (when the size of the window is characterized by its duration time). As an alternative to the windowing strategies, we can find several methods based on sampling, where its goal is to summarize the underlying characteristics of a data stream over long periods of time such that every included sample is drawn uniformly from the stream. On the other hand, in gradual forgetting no samples are completely discarded from the memory, but they are associated with weights that reflect their age (sample weighting).

In the last years several approaches have focused on forgetting strategies, such as [65], [76] or [77]. More recently, the work [66] has focused on excluding uncertain classifiers from the class label prediction.

Diversity Measures

Arguably, [68] is one of the baseline works referring to diversity measures. Here, the authors studied ten statistics which can measure diversity among binary classifier outputs: four averaged pairwise measures (the Q statistic [78], the correlation [79], the disagreement [80], [81] and the double fault [82]) and six non-pairwise measures (the entropy of the votes [83], the difficulty index [84], the Kohavi-Wolpert variance [85], the interrater agreement [86], [87], the generalized diversity [88], and the coincident failure diversity [88]). They concluded that all measures have approximately equally strong relationships with the improvement of the majority vote. Also, the measures were strongly correlated between themselves. In regards to the diversity measure selection, they stated that a choice of measure to recommend can be based on the ease of interpretation.

Here, the research community usually find the Q statistic (aka Yules' Q) as one of the best metrics for easy interpretation and simplicity ([8], [22], [39]). Considering two classifiers C_i and C_j, the Yule's Q statistic metric can be calculated as:

$$Q_{i,j} \doteq \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}, \tag{2.2}$$

where N^{ab} is the number of training samples for which the classification given by C_i is a and the classification given by C_j is b. Regarding a and b, 1 represents a correct classification and 0 is a misclassification. Q varies between -1 and 1. Classifiers that tend to recognize the same objects correctly will have positive values of Q, and those which commit errors on different objects will render a negative Q value. For an ensemble E of M classifiers, the $\bar{Q}$ statistic averaged over all pairs of classifiers is given by:

$$Q_{averaged} \doteq \frac{2}{M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} Q_{i,j}, \tag{2.3}$$

where, as mentioned above, higher/lower $\bar{Q}$ values are associated with lower/higher diversity, establishing an inversely proportional relation.

Recently, in [39] the authors discussed the possibility of calculating ensemble diversity measures in three basic stream processing scenarios: in blocks, incrementally, and prequentially. They also studies the capability of these measures to detect the concept drift.

2.6 Stream Learning Evaluation Methods

In order to quantify the performance of a stream learning method, either to assess whether one model is better than others, or to examine whether a drift detection method performs properly for a given setup, it is necessary to define a selection criterion and performance metrics [10], [77] (Figure 2.13).

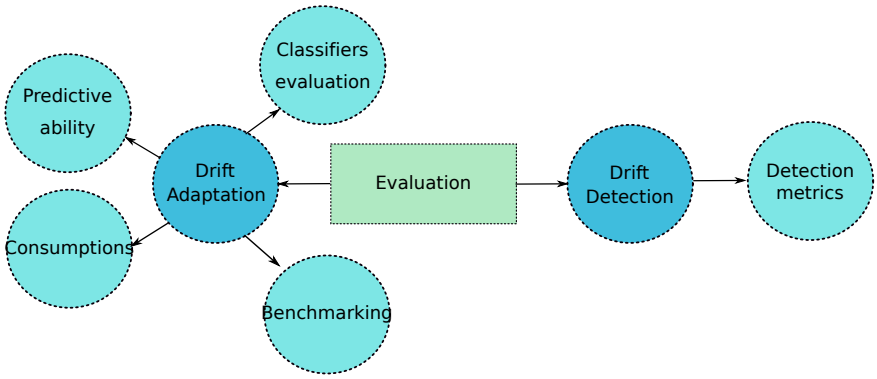


FIGURE 2.13: Evaluation module of the taxonomy for concept drift adaptation.

2.6.1 Drift Adaptation

Predictive ability

In the context of static and batch learning, cross validation is the most used strategy for estimating prediction measures. But for online learning that shows computationally strict requirements and the presence of concept drifts, it is not applicable. Specially in the case of non-stationary data streams in which one sample at a time is presented to the ensemble, *prequential evaluation* method or the *test-then-train* evolution have become the standard choice. The approach is simple: each sample is used first to test the model, and then to train the model, as Figure 2.14 shows.

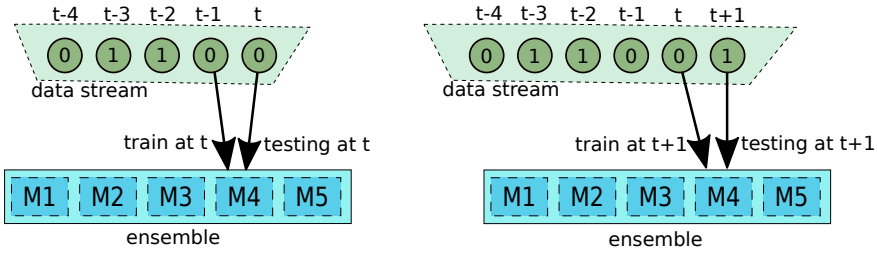


FIGURE 2.14: *Test-then-train* evaluation scheme.

Unlike the *holdout evaluation* scheme, the *interleaved-test-then-train* has the advantage that no holdout set is needed for testing, making maximum use of the available data.

The *test-then-train* strategy may produce biased results due to the fixed order of data in a sequence, and as it runs only one test in a fixed order of data. The study [89] was focused on reducing this risk

by running several tests with randomized copies of a data stream so that different distributions from the original data are approximately preserved in permutations.

Other authors [90] preferred to adapt the popular metric AUC for evaluating classifiers in stream learning, because the original version of AUC is used for static data sets and it requires the entire data set to be calculated. Therefore, they proposed a prequential version of AUC metric to be suitable for streams with skewed distributions.

Consumptions

Regarding the *memory* consumption, in [91] the authors introduced the use of RAM-Hours as an evaluation measure of the resources used by streaming learning methods, which is based on rental cost options of cloud computing services. Every GB of RAM deployed for one hour equals one RAM-Hour.

In what refers to *time* costs, the design of the learning algorithm should consider the amount of time that requires to update its structure and adapt to new data (*update time*). In an ideal situation, the update time should be lower than the arrival time of a new example.It should be also considered the amount of time that a learning algorithm needs to make a decision regarding new samples from the stream (*decision time*) [18].

Classifiers Evaluation

Statistical significance tests are very popular in machine learning when comparing two classifiers in order to validate the differences in the achieved error rates [92]. In the stream learning field there are few approaches to using these mathematical tools [93]. However, one may be critical to such approaches, because they try to transform a dynamic problem into a static one, or only consider local characteristics. So far, there has been no unified statistical significance testing framework proposed for data streams [18]. Despite this critique, we can find several significance tests in the state of the art. The McNemar test [94] is a non-parametric test used in the stream mining to compare two classifiers. When comparing more than two classifiers, the Nemenyi test [95] is appropriate for comparing all classifiers to all classifiers over multiple data sets, being based on the average ranks of the algorithms across all data sets. And it is also recommended after a rejection of the null hypothesis by Friedman [95].

Benchmarking

In order to perform a benchmarking to compare several stream learning algorithms, it is necessary to use large data sets and software implementations of the algorithms. Synthetic data sets are used for testing the algorithms under several controlled conditions and looking for their general applicability. When available, real data sets are interesting to know about the real usability and application of the algorithms under real conditions.

It deserves mentioning that the open-source software MOA (Massive Online Analysis) [96] is very well known and used to run data streaming experiments in the literature, which includes implementations of many drift adaptation and detection algorithms.

2.6.2 Drift Detection

When it comes to the evaluation of drift detection techniques, the metrics vary considerably. Nevertheless, most studies have embraced the recommendations provided in [26]. Therefore, comparisons are carried out in terms of false positives (FP), true positives (TP), false negatives (FN), and distance to the drift (DT) (Figure 2.8). FP indicate the number of drifts identified at time instants when no drift was present in the data set whatsoever; thus, lower values mean that the method does not detect false drift events. Likewise, a high TP value indicates that the method has identified true drifts during a specific period of time. FN indicates the number of missed drift detections, which can easily reach 100% because methods are based on one sample training (test-then-train). For this reason TP are usually computed over samples arriving during the drifting time. Finally, DT indicates the average distance to the drift.

2.7 Open Challenges in the Field

2.7.1 Identification of Challenges

In the last two years, online learning and concept drift have been very hot topics in machine learning literature, and they still remain attracting the attention of many researchers in the field. As many of the recent studies confirm, several open challenges should be addressed in the next years due to the importance of the field in many

real stream data applications. Next, we summarize the most relevant challenges based on very recent studies of prominent authors in the field.

Online Class Imbalance

Online class imbalance learning is an emergent research topic that often combines the challenges of both class imbalance and concept drift. Despite being a very challenging topic, very little work addresses this combined problem. Authors of the first systematic study of handling concept drift in class-imbalanced data streams [19], noted that there are still many challenges and learning issues in this field that are worth of ongoing research, such as: 1) more effective concept drift detection methods for imbalanced data streams; 2) studying the mutual effect of class imbalance and concept drift; and 3) more real-world applications with different types of class imbalances and concept drifts.

This work [18] also claims that the work with class-imbalanced and evolving streams is still in early stages. Most researchers consider the simplest problem of the imbalanced class ratio, without changes of imbalance ratio over time. Finally, new evaluation measures and more rigorous evaluation procedures are needed for evaluating algorithms in such complex imbalanced streams.

Drift Characterization and Monitoring

The characterization of the different types of concept drift that can emerge during a stream learning process has not previously been subjected to a rigorous definition and analysis. Concretely, while some qualitative drift categorizations have been proposed, just a few have been formally defined, and the quantitative descriptions required for precise and objective understanding of learner performance are still to be found and put to consensus. In [16] the authors presented one of the first framework for quantitative analysis of drift, but more effort on this issue is required in order to lay the foundations of future drift adaptation and detection algorithm developments. The auhors of [18] also claim to have more studies on the nature of some drift types.

It is also interesting the suggestion of [6]: the combination of supervised and unsupervised indicators for monitoring concept drift,

which can be a promising trend in the short future. Combining supervised and unsupervised indicators could be beneficial for two reasons:

- Handling different types of drift at the same time: while supervised indicators would be used for handling real drifts, unsupervised indicators would be used for handling virtual drifting events; and
- Early detection, because some kind of virtual drift may evolve and become real, hence by using unsupervised indicators we could expect the change, and then by using the supervised ones we could confirm its ocurrence.

The authors of the recent review [7] also encourage further research on concept evolution.

Drift Detection

In what refers to drift detection, several challenges are proposed by recent research works:

- Sequential approaches do not require much memory because each sample is processed only once and then discarded. They are suitable for detecting drifts where data streams are infinite. In this case, it is not practical to store all samples, which is the case of most real-world applications. Moreover, they can effectively detect abrupt drifts, but are less effective for handling gradual drifts, which are harder to detect because changes are so small that they are only noticed during a long time period. The limited use of memory of sequential approaches does not allow for early detection mechanisms. According to investigation of [97], sequential approaches can be more affective if they consider the temporal dependence that frequently real data streams exhibit. They also demonstrate that a simple classifier considering the temporal dependence can outperform current state-of-the-art classifiers. They suggest to pick ideas from time series analysis in order to adapt them to drift detection in presence of temporal dependence. For that reason, in [6] suggest to analyze samples through a time window in order to take into account the temporal dependence between samples. The recent review [7] also encourage researchers to consider temporal dependencies.

- As claimed in [6], windowing approaches consider that the drift is uniform, therefore affecting the entire sample space. Hence, they can effectively handle global concept drift where the difference between the old and the new concept is easy to notice, as the drift is affecting the overall sample space. However, they fail to detect local concept drifts where changes occur in only some regions of the sample space. Generally, windowing approaches are unable to deal with the scarcity of samples that represent the drift, because they consider them as noise. Hence, the time until detecting local concept drifts can be arbitrarily long. As a future trend, windowing approaches should integrate a mechanism of sample selection in order to effectively manage the scarcity of samples that represent the local drift.

- In [6] they also noted the lack of approaches tackling detecting drift in categorical time-evolving data.

- In [54], the authors propose to investigate further one of the well-known problems with the EDDM detector: its performance usually degrades when concepts are very large, because it tends to become less sensitive to concept drifts, taking too many samples to detect eventual changes.

- The use of diversity measures to detect drifts was suggested in [39]: the complementary nature of various diversity and performance measures suggests that it might be worth combining multiple detectors, which would monitor and track more than one metric.

Drift Adaptation

In [6] the authors underline several future trends and criticisms to the current state of the art:

- Passive approaches (those which update the model continuously every time new data samples arrive) adapt the learner without requiring drift detection. As a result, they can be good for handling gradual continuous drifts where the dissimilarity between consecutive data samples is not relevant enough to detect a drift. However, the authors underline the main limitation of them: their slow reaction to these drifts, because the updating process is the same whatever the drift is abrupt or

gradual. Moreover, regular model updates can be too costly as the amount of arriving data may be overwhelming. These approaches can work well only if the speed and severity of the change are known beforehand or if we have rigorous instructions provided by an expert about the nature of the drift, but this is rarely the case.

- Regarding active approaches (those which only update the model when a drift is detected), it is necessary to monitor not only the performance of the learning algorithm, but also the number of FP and FN (misdetections). In monitoring and control applications, this may lead to indicate that there are some sensors which do not work properly and that must be replaced. In such case, an unnecessary maintenance is made and may cause waste of time and resources.

- They propose to carry out a deep study of the heterogeneity of the ensembles; different learner types have different intrinsic strengths and biases, which makes them suitable for handling different types of drift. They consider that an heterogeneous ensemble could offer a promising capability to deal with data that contain multiple drifts with pronounced differences in speed and/or severity.

- Concerning the dynamic weighting process based on the ability to handle drifts for ensembles, they claim that base learners should not be only weighted according to their prediction performance, but also to their sensitivity to drifts.

Ensemble Building

As it has been shown in Section 2.5.2, diversity among the constituent learners in ensemble models is primordial when dealing with concept drifts for achieving good performance. Hence, several recent studies have been focused on this issue. In [6] they focused on diversity measures, adducing that they were developed for quantifying the diversity of the ensemble in stationary environments, and thus they only consider as main criterion the difference in performance of base learners or the prediction disagree. Along this direction, developing an appropriate diversity measure for ensemble in non-stationary environments could be of interest for the community. Also in [18] the authors suggested to investigate more dedicated diversity measures for data stream classifier ensembles. Finally, it is

worth highlighting that there is theoretical and empirical evidence that the maximum diversity in ensembles is not usually achievable with the original data on which models are trained [22]. This would motivate the development of new diversity generation methods for stream learning scenarios in the presence of concept drift.

Most online and batch-based approaches use models with parameters being either individually tuned or using some preset values (fixed for the complete analysis process). Nevertheless, as highlighted in Section 2.5.2, parameters configuration may no longer be the sufficiently good after changes appear in the data stream. Therefore, in [18] the authors proposed the development of a new methodology for self-tuning streaming ensemble systems may lead to improved predictive power. Additionally, tuning parameters for single classifiers should take into account that they are components within the ensemble. Thus, more global update methods that can lead to obtain more complementary models seems to be worth exploring.

Finally, in [66] the authors plan to further investigate methods for determining which classifiers should abstain from the voting process as well as to further improve their ability to handle various concept drifts. They also place the focus on adapting their methods to mining recurrent concept drifts, by making members abstain from the voting process on new concepts, and retrieving them when one previously seen emerges again. Besides, the authors of [18] proposed for ensemble pruning more advanced pruning techniques, which could also exploit a multiple criteria analysis, including not only current predictive ability, but also computational efficiency of base models, memory usage or other resources, current diversity of the ensemble and available information on class labels. At the same time, these pruning techniques should imply a minimal computational overhead. Such compound, yet lightweight approaches, should lead to maintaining better ensemble setup and improve adaptation abilities to various types of changes.

Feature Drift

The authors of [14] present a study focused on feature drift adaptation where they analyze the possible changes in the relevance of features through time, a field that has not received much attention to date. They identified several open challenges to tackle in the future:

- Inductive trees (e.g. Very Fast Decision Trees known as VFDT [98]) and decision rule learning techniques do not encompass

strategies for adapting its model to drifts in data, therefore they must be accompanied by drift detectors that periodically reset the entire rule set according to error rates of the classifier.

- Approaches like continuously-changing data streams based on VFDT (also known as CVFDT), heterogeneous ensemble for feature drifts in data streams [99] (HEFT-Stream), and Landmark-based Feature Drift Detector [14] (LFDD) assume that the most discriminative subset of features can be chosen by filters on disjoint batches of samples. The drawback lies in determining the size of these batches, which directly affects the learning process. Small windows lead to a quicker change recognition in the selected subset of features. However, this approach may also lead to the detection of false positives if the stream is noisy. By contrast, larger windows enable a higher amount of data from which to learn, yet fail to quickly detect changes in the most discriminative subset.

Finally, the authors of the recent review in [7] also encourage researchers of stream learning to further research on feature drifst.

Preprocessing

A thoughtful study on stream data preprocessing [9] reviewed some preprocessing techniques on the literature that would deserve a further development in the future:

- Data Reduction Techniques (DRTs) aim to obtain a representative training set with a lower size when compared to the original one, and with similar or even higher generalization capabilities when fed to a predictive model. They highlight the importance of more proposals for online learning from data streams due to the relevance of these techniques in the state of the art.

- Due to the ability of discretizers to reduce the continuous feature space, they found it interesting to perform a deep research on the use of supervised discretizers. Most current methods are unsupervised techniques based on quantiles, using an adaptation strategy with smooth shifts in the definition of intervals and the previous definition of intervals. Adding class information to the discretization process would allow accommodating local drifts, where the properties of only some classes

change. Additionally, they envision the potential of ensemble learning, which will allow using various discretization intervals for training a diverse set of classifiers.

- No pure wrapper-based solutions have been yet proposed for online problems. Efficient implementations of these methods may be challenging due to their increased computational cost, but this may be compensated by the inherent discriminative ability of online learners and their adaptiveness to drifts. One potential solution would be to combine filter and wrapper approaches in order to reduce the number of times the more costly method is used, and to allow for continuous classification even during the wrapper computation. Another potential solution lies in using high-performance solutions based on GPU or distributed computing to reduce the computational load associated to this approach.

- There is a need for further research on feature and sample selection methods that can directly address the problem of concept drift. One way of approaching this would be to combine sample selection approaches with drift detection module that could directly influence the usability of prototypes. Whenever a strong drift is being detected, one may discard the previous prototypes and use only the incoming objects. After stream stabilizes, the sample selection can be repeated to adapt to the current concept. Another potential solution is to have weighted prototypes, where weights would reflect how long time ago they were created and how useful they are to mining the current state of the stream. This would permit to smoothly forget outdated prototypes, while keeping in memory the still useful ones. Local drifts that occur only within a subset of classes should also be considered. In such a case only selected prototypes should be modified in the areas of drift presence. This would require class-based prototype pruning methods and a method to overlook the influence of these drifting prototypes on stationary classes.

- There exist no solutions that directly take into account the possibility of *recurrent* concept drifts. Therefore, it seems promising to develop preprocessing methods that could accommodate the fact that previously used sets of features/samples/discrete bins might become useful once again in the future.

Benchmarking

One relevant open issue is rethinking frameworks and data sets for testing stream algorithms [18]. On the one hand, the number of real-world publicly available data sets for testing, stream classifiers is still too small for the research needs of the community. It limits comparative studies of different streaming algorithms. Moreover, some popular data used in the literature is questioned for their representativeness of real drifts. This is a more difficult situation when compared to the state of available static data sets such as the UCI Machine Learning Repository[1].

On the other hand, popular online analysis frameworks for massive data like MOA [96], SAMOA [73], StreamDM[2], and more recently Scikit-Multiflow [100], allow researches to test their stream learning algorithms. In the case of MOA and Scikit-Multiflow, they are especially useful because allow researchers to simulate drifts (varying their severity and speed) in traditional stationary data sets, facilitating the reuse of these common data sets for non-stationary purposes.

Extreme Verification Latency

The recent review on ensemble learning [18] points out the necessity of developing new approaches to deal with delayed information. Many of the discussed approaches employ a kind of transfer learning, where predictions from models learned from labeled examples are transferred to next unlabeled portions of the data. In general, they are more useful for limited gradual drifts, while more complex scenarios are still open problems. Developing new approaches to deal with delayed information (including ensembles) that would work in the presence of different types of drift is a non-trivial research task. It would be particularly useful for many real life automated systems, where the interaction with human experts is limited and impractical. Finally, delayed information may not refer to target values only, but may concern also incomplete attribute descriptions. The problem of incomplete data is more intensively studied in static, off-line data mining, where different imputation techniques have been developed. In the streaming context, research is scarce in regards to such techniques or other approaches which

[1] https://archive.ics.uci.edu/ml

[2] http://huawei-noah.github.io/streamDM/

could learn classifiers with omitting such incomplete descriptions and then could update the knowledge captured by the classifier.

2.7.2 Challenges Addressed in this Thesis

The following challenges have been selected among others due to their wider impact on the stream learning field. As it will be further explained below, key topics such as diversity, ensemble building, data reduction, single stream learners, and drift detectors have been considered in detail in this thesis:

Challenge 1: Diversity Creation in Ensembles

As it has been highlighted in Section 2.7.1 (Ensemble Building), on the one hand the diversity in ensemble building is primordial when dealing with concept drift for achieving good performance, but on the other hand the design of the ensemble should consider the so-called *stability-plasticity dilemma*. We have also seen how the state of the art recently demands more diversity measures for data stream classifier ensembles (see Section 2.7.1, Ensemble Building). Finally, there is theoretical and empirical evidence that the maximum diversity in ensembles is not usually achievable with the original data on which models are trained. This would motivate the development of new diversity generation methods for stream learning scenarios in the presence of concept drift (see Section 2.7.1, Ensemble Building).

Chapter 3 tackles these challenges by providing an innovative approach to artificially generate an optimal diversity level when prediction ensembles are built once shortly after a drift occurs, considering simultaneously the two conflicting objectives of the *stability-plasticity dilemma* (diversity and classification performance). This approach, coined as Diversity geneRator based on Evolutionary technique powered by Density estimation (DRED), uses a Kernel Density Estimation (KDE) method to generate synthetic data, which are subsequently labeled by means a multi-objective optimization method that allows training each model of the ensemble with a different subset of synthetic samples. Bhattacharyya distance [101] will be used for first time in the literature to measure the diversity. Computational experiments will reveal that the proposed approach can be hybridized with other traditional diversity-induction methods, yielding optimized levels of diversity that render an enhanced recovery from drifts.

Challenge 2: Data Reduction Techniques and New Stream Learning Algorithms

Section 2.7.1 (Preprocessing) has underscored the importance of developing more proposals based on DRTs for online learning from data streams due to the relevance of these techniques in the state of the art. They provide a representative training set with a lower size when compared to the original one, and with similar or even higher generalization capability, which may reduce the time and computational costs. On the other hand, despite the single classifiers generally tend to be more less accurate than ensemble approaches, they can generally provide a lower computational cost [15], which makes them an attractive solution for massive data stream a scenarios under severe computational restrictions. Besides, the flexibility to easily incorporate new data into a classification model when new data are presented (simply by adding new members to the ensemble) has been traditionally assigned by default to ensemble approaches. As it will be further detailed in Chapter 4, this ability must not be solely attributed to ensemble approaches. Finally, the state of the art is always in need for new approaches to deal with online learning and concept drift under new points of view, thereby and providing new innovative solutions for concrete problems.

Recently, neural networks have also been gaining a renewed popularity for online learning in changing environments [102], [103]. Therefore, Chapter 4 tackles these challenges by providing a single model approach based on eSNNs that, combined with DTRs and hybridized with a drift detector, is competitive in terms of its balance between predictive performance and complexity in comparison with traditional ensemble approaches. In addition, eSNNs have revealed themselves as one of the most successful approaches to model the behavior and learning potential of the brain, and exploit them to undertake practical learning tasks. We will show in this challenge how eSNN, unlike most off-the-shelf classification models, does not need to be retrained if it is used in a changing environment and does not fail to scale properly. Taking the existing eSNN model as a basis, a set of adaptations is done to the original technique in order to deploy it in online learning scenarios by fulfilling with the restrictions described in Section 2.2.

Challenge 3: Drift Detection based on eSNN

In Section 2.7.1 (Drift Detection) several authors have encouraged other researchers to consider the temporal dependence between samples that frequently real data streams exhibit. As we will explain in Section 4.2, in essence SNNs leverage spike information representation so as to construct spike-time learning rules that capture temporal associations between a large number of temporal variables in streaming data.

Besides, all the existing drift detectors usually use a specific classifier (base learner), and analyze its performance score (in general, accuracy or error rate) to indicate whether a drift has occurred. As opposed to the majority of these drift detection mechanisms, Chapter 5 elaborates on a new approach `eSNN-DD`, which does not require any additional base learner to analyze its performance metrics towards indicating when a drift has occurred, which simplifies the architecture of drift detectors.

This new drift detection method exploits the structural changes (neurons merges) of an eSNN predictive model to identify the drift point, hence it can be regarded as an *embedded* drift detection approach that stems as a byproduct of the particular learning algorithm of this class of neural networks. Finally, as this drift detector is based on an eSNN, it is able to capture temporal associations between temporal variables of data streams.

2.8 Summary

Recent advances in computational intelligent systems have focused on tackling problems related to the evolving environments. An increasing number of real world applications data are presented as streams that may evolve over time (*concept drift*). Handling concept drift is becoming an attractive topic of research that concerns multidisciplinary domains such that machine learning, data mining, ubiquitous knowledge discovery, statistic decision theory, etc. Therefore, a rich body of the literature has been devoted to the study of methods and techniques for handling drifting data. The main objective of this chapter has been to present an easy understanding of the concept drift issues and related works, in order to help readers from different disciplines to understand the challenges addressed by this thesis.

Chapter 3

Challenge 1: Diversity Creation in Ensembles

3.1 Introduction

This chapter formulates the diversity balance as an optimization problem, making use of an evolutionary approach for the construction of ensembles with different levels of diversity. Specifically, a new strategy is proposed to strengthen the diversity generation procedure of traditional techniques such as online bagging, class-switching, and boosting, proving their performance under a wide number of drift conditions. In particular, the approach called Diversity generator based on Evolutionary technique powered by Density estimation (DRED) resorts to a very short buffer of samples (reservoir) as a prototype supplier that, when training a KDE method, generates synthetic samples to be evolved through a low-complexity multi-objective optimization algorithm driven by the Pareto balance between ensemble diversity and classification performance. Specifically the multi-objective solver decides whether every synthetic sample 1) is not used for training; 2) is used for training with its original label; or 3) is used for training with a switched label. The approach is in line with the theoretical and empirical evidence published in [22], by which it was found that the maximum diversity in ensembles is not usually achievable with the original data on which models are trained.

All this motivates the generative method proposed in this chapter, whose novel contributions over the state of the art can be summarized as follows:

1. The use of KDE methods for online synthetic data generation driven by its bandwidth parameter, which permits increasing or decreasing the degree to which newly generated synthetic

data spreads over the feature space. This aspect of DRED is in line with recent approaches based on KDE for online environments [104], [105], with emphasis on [106] which modifies the original form of KDE to be applied to data streams by maintaining a window of recent data samples to evaluate the KDE algorithm.

2. An innovative perspective on diversity generation, based on optimally labeling synthetic distances as measured by a quantitative metric of ensemble diversity. The literature is rich in what refers to strategies to manipulate the input data prior to ensemble learning, such as randomly eliminating features [107] or modifying the class labels [108]. This last strategy was considered in [109], used in [110] for the so-called DECORATE method, and more recently in [111] and [112]. However, it has not widely used in online classification problems, nor to generate diversity in ensembles for online learning under concept drift.

3. The novel use of a multi-objective solver to achieve a balanced level of diversity and classification performance score. Due to the above noted importance of achieving a good balance between adaptability (diversity) and classification performance along time [113], there is a necessity for novel solutions to optimally balance the diversity in ensemble learning. The portfolio of multi-objective optimization techniques is rich in the particular case of evolutionary schemes, with renowned schemes such as NSGA-II [114], Pareto Archived Evolution Strategy (PAES) [115], Strength Pareto Evolutionary Algorithm-II (SPEA2) [116], Genetic algorithms [117], [118], and other biology inspired techniques [119]. In this challenge we leverage recent findings on computationally efficient implementations of multi-objective schemes [120] towards constructing ensemble classifiers differently trading diversity for stability (performance).

4. The novel use of the Bhattacharyya distance [101] to measure the diversity of an ensemble in data streams for concept drift scenarios. As the recent survey [18] highlights, there is a need for dedicated diversity measures for data stream classifier ensembles, besides, only a few authors directly consider promoting the diversity while constructing an ensemble. In regards to this challenge, the chapter focuses on both directions.

3.2 Core Methods

Studies on the diversity of ensemble classifiers abound in the literature [22], [68], [121]. Among them, the diversity analysis for online learning in the presence of concept drift provided in [3] is particularly insightful, as it concludes that different levels of diversity are required before and after a drift in order to improve generalization on the old or new concepts. Furthermore, the study highlights that diversity can help to reduce the initial increase in error caused by a drift, but does not provide a faster recovery from drifts in long term. The next sections justify and describe in advance the methods embedded in the proposed DRED scheme, which are required to achieve a fast adaptation after drift occurs. Throughout further technical explanations the following mathematical notation will be used: single-dimensional variables will be denoted in uppercase (X), with vector variables in bold ($\mathbf{X}$). Realizations of such variables will be in lower case (x, $\mathbf{x}$). The main symbols used throughout this chapter are summarized in the List of Symbols.

3.2.1 Kernel Density Estimation

In order to achieve a prompt adaptation to drift without decreasing in classification performance, DRED should find an ensemble characterized by a suitable level of diversity using a preallocated amount of main memory. Under this constraint, DRED stores a reduced reservoir of past samples [106] with the goal of using them as prototypes to generate synthetic samples that serve as training data set for the ensemble. As will be further explained Section 3.2.2, the metrics that represent the objectives to optimize (diversity and classification performance) need to be evaluated on a group of samples that cannot be stored within an online learning process. It is at this point when a KDE method helps creating synthetic samples resembling the distribution of samples stored in the reservoir depending on the level of diversity required to face the drift.

In order to generate this synthetic data set, the KDE needs to be adjusted to the input distribution to a greater (when samples distribution need to be more similar to the input distribution, searching for less diversity) or lower extent (when samples distribution need to be less similar to the input distribution, searching for more diversity). This adjustment can be driven by the bandwidth parameter $h \in \mathbb{R}^+$ of the selected kernel function in the KDE estimator. The bandwidth acts as a smoothing parameter, controlling the level

of diversity: large bandwidth values lead to high diversity levels (smooth density distribution), whereas a small bandwidth leads to a very low diversity (sharp density distribution). For instance, a square uniform kernel can be expressed as $\Psi(x;h) \propto 1$ if $x < h$ (0 otherwise), so that the probability estimate at a point x based on a set of samples $\{x_i\}_{n=1}^{N}$ is

$$\widehat{f}_X(x) = \frac{1}{N} \sum_{n=1}^{N} \Psi((x - x_n); h), \tag{3.1}$$

namely, as the value of the scaled sum of N kernel functions located at the samples of the set over which the kernel density is computed. The bandwidth h allows controlling the trade-off between bias and variance in the density estimate. Figure 3.1 illustrates this trade-off with different values of h and a top-hat kernel, resulting in achieving less and more diverse samples respectively.

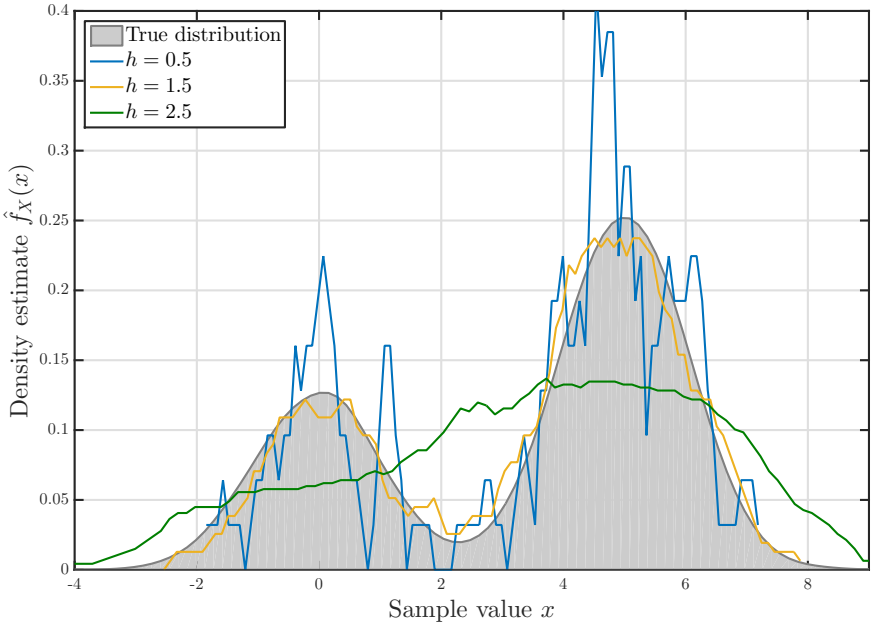


FIGURE 3.1: Density estimate provided by a top-hat KDE with $h \in \{0.5, 1.5, 2.5\}$.

3.2.2 Multi-objective Optimization: Diversity and Performance Metrics

As mentioned in the introductory section, the *stability-plasticity dilemma* can be formulated as a multi-objective optimization problem based on the Pareto between diversity and classification performance, issue already suggested for batch learning in [122]. In DRED the goal is to find a balanced ensemble with the greatest classification performance possible, and at the same time with as much diversity as possible, in order to adapt quickly when the drift occurs. Therefore, a Multi-objective Optimization Technique (MOT) should account for both objectives.

In order to evaluate the first objective of the problem (diversity in a given ensemble), several measures have been proposed [22], [68], [123], most of which require a supervised set of samples for its computation. This section will focus on ensembles composed by Gaussian Naive Bayes (GNB) classifiers, which hinge on the independence assumption between every pair of the input features to yield an extremely fast online learning process [124], [125]. For this particular type of classifiers, a measure of diversity can be based on the Bhattacharyya distance [101], which provides a numerical estimation of the similarity between two probability distributions. For two multivariate normal distributions $\mathcal{N}(\boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A)$ and $\mathcal{N}(\boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B)$, the Bhattacharyya distance $D_{A,B} \in [0, \infty)$ is given by:

$$D_{A,B} \triangleq \frac{1}{8}(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B) + \frac{1}{2}\ln\left(\frac{\det \boldsymbol{\Sigma}}{\sqrt{\det \boldsymbol{\Sigma}_A \det \boldsymbol{\Sigma}_B}}\right), \quad (3.2)$$

where $\{\boldsymbol{\mu}_A, \boldsymbol{\mu}_B\}$ and $\{\boldsymbol{\Sigma}_A, \boldsymbol{\Sigma}_B\}$ are the means and covariances of the distributions, and $\boldsymbol{\Sigma} \triangleq 0.5\,(\boldsymbol{\Sigma}_A + \Sigma_B)$. The previous definition can be used for measuring the diversity between two given GNB classifiers GNB_k and $\text{GNB}_{k'}$: let us denote the mean and variance for feature $m \in \{1, \ldots, M\}$, class $l \in \{1, \ldots, L\}$ and classifier GNB_k as $\mu_k^{m,l}$ and $\sigma_k^{m,l}$, such that $\boldsymbol{\mu}_k^l \triangleq (\mu_k^{1,l}, \ldots, \mu_k^{M,l})^\top$ and:

$$\boldsymbol{\Sigma}_k^l \triangleq \begin{pmatrix} \sigma_k^{1,l} & 0 & \ldots & 0 \\ 0 & \sigma_k^{2,l} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_k^{M,l} \end{pmatrix}, \quad (3.3)$$

where the naive Bayes assumption is satisfied by the diagonal covariance matrix. A pairwise measure of distance between the multivariate distributions estimated by both classifiers for label l can be computed as:

$$D^l_{k,k'} = \frac{1}{8}(\boldsymbol{\mu}^l_k - \boldsymbol{\mu}^l_{k'})^T \left(\boldsymbol{\Sigma}^l\right)^{-1} (\boldsymbol{\mu}^l_k - \boldsymbol{\mu}^l_{k'}) + \frac{1}{2} \ln \left(\frac{\det \boldsymbol{\Sigma}^l}{\sqrt{\det \boldsymbol{\Sigma}^l_k \det \boldsymbol{\Sigma}^l_{k'}}} \right), \tag{3.4}$$

with $\boldsymbol{\Sigma}^l \triangleq 0.5(\boldsymbol{\Sigma}^l_k + \boldsymbol{\Sigma}^l_{k'})$. From the above expression a bounded coefficient of diversity in the range [0,1] can be computed as $C^l_{k,k'} \triangleq 1 - \exp -D^l_{k,k'}$, taking value 1 if the estimated marginals for label l by both classifiers do not overlap with each other. The overall diversity level of the ensemble results from averaging the coefficients first over labels, then over all model pairs within the ensemble, namely:

$$C(\{\mathrm{GNB}_k\}^K_{k=1}) \triangleq \frac{2}{LK(K-1)} \sum_{k=1}^{K} \sum_{k'=k+1}^{K} \sum_{l=1}^{L} C^l_{k,k'}, \tag{3.5}$$

where K denotes the number of weak classifiers within the ensembles and, again, $C(\{\mathrm{GNB}_k\}^K_{k=1}) \rightarrow 1$ if the ensemble is highly diverse.

As for the another objective to be optimized by the MOT, without loss of generality[1] we will use the Area Under the Curve (AUC) score, widely acknowledged in the literature to measure the predictive performance of binary classifiers [10], [17], [126]. Most studies of online learning measure the performance of a classifier by classification accuracy, which is inappropriate for applications where the data are unevenly distributed among different classes. The work on this challenge will embrace this recommendation for comparing performance figures among different ensembles.

DRED operates shortly after a concept drift has been detected in the information stream. As it has been mentioned before, DRED operates together with the traditional diversity generation techniques (online bagging, class-switching, and boosting) making them more powerful when producing diverse models, showing that our strategy is able to improve their performance under a wide number of drift conditions. When a change in the data distribution has been

[1]When dealing with multiple class labels other scores can be selected, such as the f1-score.

detected at time t_D, DRED is executed once in order to find a portfolio of ensemble classifiers that differently yet optimally balances between performance and diversity. As will be later explained in detail, this is accomplished by generating an additional set of training samples $\{\mathbf{x}_n^{\boxplus}\}_{n=1}^{N^{\boxplus}}$ based on repeatedly sampling KDE models trained over a W-sized time window of past stream samples, i.e.:

$$\mathbf{X}_n^{\boxplus} \sim \frac{1}{W} \sum_{w=1}^{W} \Psi((\mathbf{x} - \mathbf{x}_{t_D-w}); h), \tag{3.6}$$

where $\{\mathbf{x}_{t_D-W}, \mathbf{x}_{t_D-W+1}, \ldots, \mathbf{x}_{t_D-1}\}$ denotes the aforementioned small-sized reservoir over which the KDE estimator is trained. In order to account for differently labeled samples in this reservoir, a separate KDE model per class is constructed. Thereby supervised samples to compute an estimation of the performance metric can be also produced.

Once such $N^{\boxplus}$ samples are constructed, the MOT aims at optimizing the labels associated to each of such samples when fed to every GNB model compounding the ensemble. If l_k^n denotes the label associated to synthetic sample $\mathbf{x}_n^{\boxplus}$ when input to model GNB_k (with $k \in \{1, \ldots, K\}$), the optimization problem tackled by the MOT focuses on finding the set of labels:

$$\mathbf{L}^{\boxplus} \triangleq \{l_1^1, \ldots, l_1^{N^{\boxplus}}, l_2^1, \ldots, l_2^{N^{\boxplus}}, \ldots, l_K^{N^{\boxplus}}\} \tag{3.7}$$

under the Pareto trade-off between diversity and performance. Mathematically:

$$\begin{aligned} \boldsymbol{L}^{\boxplus} = \arg\limits_{\boldsymbol{L}} [&\max C(\{\text{GNB}_k(\{\mathbf{x}_n^{\boxplus}, l_k^n\}_{n=1}^{N^{\boxplus}})\}_{k=1}^{K}), \\ &\max AUC(\{\text{GNB}_k(\{\mathbf{x}_n^{\boxplus}, l_k^n\}_{n=1}^{N^{\boxplus}})\}_{k=1}^{K})], \end{aligned} \tag{3.8}$$

subject to $l_k^n \in \{0, 1, \ldots, L\}$, with L denoting the number of classes and $l_k^n = 0$ standing for the case where $\mathbf{x}_n^{\boxplus}$ is not fed to classifier $k \in \{1, \ldots, K\}$. In the above formulae $\text{GNB}_k(\{\mathbf{x}, l\})$ stands for the k-th GNB learner being incrementally trained with samples $\{\mathbf{x}\}$ and their labels $\{l\}$.

The above formulation seeks to infer a set of Pareto-optimal ensembles as the one exemplified in Figure 3.2. In essence the goal is to maximize the diversity level for a given performance score or, equivalently, to construct a maximally performing ensemble subject to a certain amount of diversity between its compounding learners.

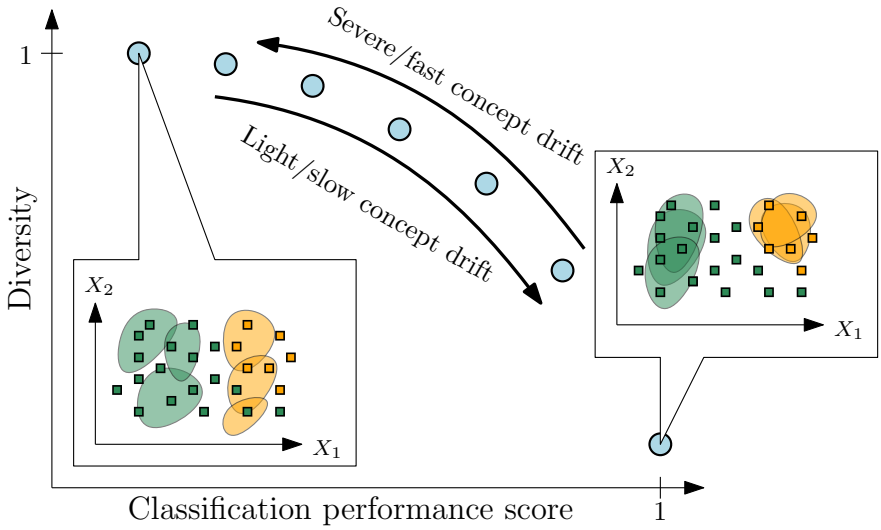


FIGURE 3.2: Example of the estimated Pareto front between performance ($AUC(\{\text{GNB}_k\}_{k=1}^{K})$) and diversity ($C(\{\text{GNB}_k\}_{k=1}^{K})$) provided by the MOT for an ensemble with $K = 3$ GNB classifiers and $L = 2$ labels. The selection of the ensemble to continue predicting the data stream is driven by the estimated characteristics of the drift: more severe/fast drifts should promote diversity (as evinced by a less coherence in the decision regions for the classifiers), whereas light/slow drifts should maintain the ensemble in its more stable configuration.

The rationale behind this Pareto formulation is to let the overall online classifier decide, upon the estimation of the drift severity and speed, whether it should opt for diversity (fast adaptation to drift) or performance (a better predictive score in stable concepts). As can be observed in the plot, sacrificing one objective for the other is a matter of the type of drift, in direct consensus with the empirical findings contributed in [3].

The use of an optimization technique might be deemed incompatible with an online learning setting, where fast decisions are required. In this regard it should be pointed out that this optimization process is performed only once, shortly after the drift occurs. Furthermore, the parameters driving the optimization procedure – namely, population size, number of iterations – are set to values that yield minimal computational costs. Although the insertion of an

evolutionary solver comes along with a penalty in terms of computation resources, we will later show that the improved performance recovery provided by the evolved ensemble after the drift outgains this drawback, making the application of DRED suitable for those cases where information flows arrive at moderate rates and/or computational resources are not severely limited. Furthermore, computation times of hardware implementations of the NSGA-II multi-objective solver have been reported to be as low as 6 milliseconds for a population size of 100 individuals and 250 generations [127], a far more complex configuration than the ones later used for DRED.

3.3 Proposed Approach: DRED

Let us recall that DRED is a new strategy to strengthen and make more powerful the diversity generation procedure of the traditional diversity generation techniques, with the result that the DRED hybrids (`DRED-OB`, `DRED-CS`, and `DRED-BOOST`) are the combinations of DRED with online bagging, class-switching, and boosting respectively.

In all of these combinations, as shown in Figure 3.3, DRED hybrids operate as the traditional techniques, in two modes: before and after drift detection. While drift is not detected, models within the ensemble are trained and tested as imposed by the selected traditional technique. Once a drift is detected, DRED is triggered once to introduce diversity in the ensemble based on KDE and a MOT to efficiently undertake the multi-objective optimization problem in Expression (3.8). To this end DRED stores samples arriving within a very short window of time using a buffer or reservoir (lines 1-7 in Algorithm 1). Once W samples have been collected, they are used to train a KDE model per label (line 8), which will be subsequently exploited to generate synthetic samples for all classes towards two objectives: 1) to compose the set of synthetic samples whose labels will be subsequently evolved by the MOT (line 9); and 2) to yield a supervised data set for evaluating the predictive score of evolved ensembles (line 10). The former objective requires sampling the learned density $N^{\boxplus}$ times only once for the entire DRED procedure, whereas the latter objective queries $N^{\boxtimes}$ samples per generation of the MOT solver in order to prevent ensembles from overfitting.

Once a model $\mathrm{KDE}_l(h)$ has been built for each label $l \in \{1, \ldots, L\}$, labels associated to the synthetic data set $\{\mathbf{x}_n^{\boxplus}\}_{n=1}^{N^{\boxplus}}$ drawn from such models to provide diversity to the ensemble are optimized by means

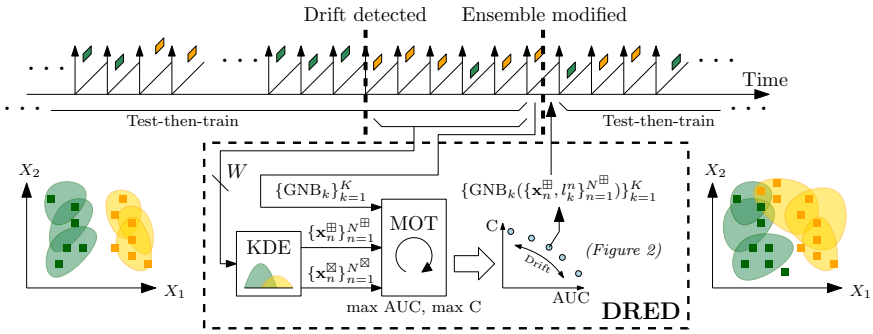


FIGURE 3.3: Schematic diagram of the proposed DRED scheme in a test-then-train online learning scenario: right after a drift has been detected, the method captures the distribution underlying a short data window by means of a KDE model, after which a set of $N^{\boxplus}$ synthetic samples are generated. Labels associated to such samples are evolved via a MOT to balance the Pareto between plasticity (diversity) and stability (performance). The characteristics of the drift finally drive the selection of the *optimized* ensemble to be applied from the drift detection time onwards.

of the selected MOT. The criteria of the problem tackled by the MOT is based on the quantitative measure of diversity as per Expression (3.5) and an estimated performance score of the ensemble trained with $\{\mathbf{x}_n^{\boxplus}\}_{n=1}^{N^{\boxplus}}$ and the set of labels corresponding to every individual. Following the aforementioned formulation, solutions evolved by the MOT are numerically encoded as a vector of $N^{\boxplus} \times K$ integers from the set $\{0, 1, \ldots, L\}$. With each new candidate solution all models of the ensemble are trained incrementally with $\{\mathbf{x}_n^{\boxplus}\}_{n=1}^{N^{\boxplus}}$, but with labels dictated by its integer encoding. As such, value 0 at position $K(k-1)+n$ within a given candidate means that the n-th synthetic sample $\mathbf{x}_n^{\boxtimes}$ cannot be used for training the k-th learner within the ensemble, whereas any other value at this position within the individual sets the label to be used along with $\mathbf{x}_n^{\boxtimes}$ and input to GNB_k for its incremental learning. As a result, each model of the ensemble is trained with the same synthetic samples, yet with different labels. Once the ensemble is trained for a given individual within the MOT search process, the KDE models are again sampled $N^{\boxtimes}$ times to produce new synthetic samples which, along with the labels corresponding to the KDE model from which they are generated, serve as a supervised data set to estimate the performance

score of the trained ensemble. This indicator and the diversity measure as per Expression (3.5) are then used by the MOT to infer the Pareto front between such conflicting objectives.

Algorithm 1: Proposed DRED(h) scheme

Input : Ensemble $\{\text{GNB}_k\}_{k=1}^{K}$; MOT algorithm; reservoir size W; number of synthetic samples for training $N^{\boxplus}$; number of synthetic samples for performance estimation $N^{\boxtimes}$; kernel density estimation method KDE; and bandwidth coefficient for diversity h

Output: Updated ensemble $\{\text{GNB}_k^{\star}\}_{k=1}^{K}$

$w = 0$
while $w < W$ **do**
  Perform the selected traditional scheme: Algorithm 2 (BAGGING), Algorithm 3 (SWITCHING) or Algorithm 4 (BOOSTING)
  Store sample $\{\mathbf{x}_{tr}(t), l_{tr}(t)\}$ in the reservoir. Let $w = w + 1$
end
Infer the distribution of every label $l \in \{1, \ldots, L\}$ by constructing a KDE model $\text{KDE}_l(h)$ base on the samples stored in the reservoir
Draw $N^{\boxplus}$ synthetic samples $\{\mathbf{x}_n^{\boxplus}\}_{n=1}^{N^{\boxplus}}$ from $\{\text{KDE}_l(h)\}_{l=1}^{L}$ considering the distribution of classes in the stream
Run once the MOT using Expression (3.5) to compute the diversity of ensembles and repeatedly sampling $\{\text{KDE}_l(h)\}_{l=1}^{L}$ to produce synthetic examples $\{\mathbf{x}_n^{\boxtimes}\}_{n=1}^{N^{\boxtimes}}$ to estimate their performance over the new concept
Select the set of optimized labels $\mathbf{L}^{\boxplus}$ from the Pareto front produced by the MOT based on the estimated severity and duration of the drift. If no such information can be inferred by the drift detector, opt for the label set that, when fed to $\{\text{GNB}_k^{\star}\}_{k=1}^{K}$, produces the highest diversity within the front
for $k = 1$ *to* K **do**
  Incrementally train the input ensemble with synthetic samples and optimized labels: $\text{GNB}_k^{\star} = \text{GNB}_k(\{\mathbf{x}_n^{\boxplus}, l_n^{\boxplus}\}_{n=1}^{N^{\boxplus}})$
end

At this point a solution must be selected from the portfolio of Pareto optimal individuals found by the MOT: intuitively a sharp

change in the concept would require a very diverse ensemble, as opposed to slow or subtle drifts where the priority is diversify the ensemble while maintaining the learned knowledge as much as possible. All in all, a proper choice of the Pareto-optimal ensemble is strongly linked to the availability of a priori knowledge about the kind of drift, closely alike assumptions taken for finely tuning other online learning schemes in the literature. When lacking such information, the solution maximizing the diversity of the ensemble can be selected as a default choice; nevertheless, DRED produces a whole range of balanced ensembles should e.g. a concept drift detector be able to infer reliably the severity and speed of the concept change. Regardless the criterion used to select a solution within the Pareto front, DRED finally trains the ensemble with the synthetic samples $\{\mathbf{x}_n^{\boxplus}\}_{n=1}^{N^{\boxplus}}$ and the selected set of labels for every compounding learner (lines 12-14), yielding the sought optimized ensemble.

Before proceeding further with the experimental part of this study, it is important to highlight that unlike other schemes from the literature, DRED is triggered only once shortly after a concept drift has been detected. This ensures that once the process is triggered, the models of the ensemble have been diversified to adapt to the drift and, at the same time, learn the new concept maintaining the classification scores in acceptable margins. If triggered at each time step after the drift, the diversity levels provided by DRED might become counterproductive as the classification performance would decrease when the new concept has stabilized (specially when dealing with severe and/or fast drifts), and the complexity could compromise usual processing requirements of data stream mining.

3.4 Computer Experiments

Several computational experiments have been run with a threefold aim: 1) to test and verify the predictive gains attained by the traditional diversity generation schemes powered by DRED, 2) to evince that the Bhattacharyya distance serve as diversity measure to build ensembles, being the basis of an optimization processs focused on keeping the balance between stability and plasticity, and 3) to confirm that a synthetic generation of samples within an optimal and intelligent labeling process improves the overall performance of the ensemble.

3.4.1 Data Sets

When dealing with real data sets it is not possible to know a priori the beginning of the drift, its severity, duration or even determine reliably if the observed change is an evidence of a drift start or an isolated outlier. For this reason, a first analysis of the behavior of the proposed approach discussed in this section will resort to artificial data sets, in which all experiments assume perfect drift detection. After that, a real data set will be used to know the behavior of the proposed approach under real conditions.

Results for 4 different class-balanced problems [3] (labeled as `CIRCLE`, `LINE`, `SINEH` and `SINEV`) will be considered, each containing one single concept drift simulated by varying among two emulated levels of severity (low and high) and two speeds (low and high). This variability results in 4 different types of drift for each data set. It is worth pointing out that, on the one hand, severity represents the amount of changes caused by a new concept, measured by the percentage of the input space which has its target class changed after the drift is complete; on the other hand, speed is the inverse of the time taken for a new concept to completely replace the old one, being measured by the inverse of the number of time steps taken for a new concept to completely replace the old one. In all data sets each example corresponds to one time step in a test-then-train learning scheme, totaling 2000 time steps ($t \in \{1, \ldots, 2000\}$), two continuous input features $\{X_1, X_2\}$ (normalized to the range $[0, 1]$ in all cases) and a binary target class $l \in \{1, 2\}$. Drift occurs at $t = 1000$. The number of samples belonging to class 1 and 2 is always the same, having the effect of changing the unconditional probability distribution function when the drift occurs. We refer to [3] for further details on these data sets.

Additionally, we will employ a different data set [128] with more drifts of different severities, and much more samples in order to evaluate the behavior of DRED hybrids. The SEA artificial data set consists of three attributes, where only the first two attributes are relevant and no class noise is introduced. All three attributes have values between 0 and 10. The class label is determined by $X_1 + X_2 \leq \theta$, where X_1 and X_2 represent the first two attributes and θ is a threshold value. If the condition is satisfied, then the example will be labeled as class 0; otherwise, it will be labelled as class 1. θ is adjusted for new concepts. The data set has $60,000$ time steps, suffering three different drifts at $15,000$, $30,000$, and $45,000$ time step, and showing four different concepts (threshold values 8, 9, 7, and 9.5 for each

concept) producing drifts of different severities.

Finally, the Australian New South Wales Electricity Market [29] (labeled as ELEC2) is the real-world data set used in the experiments with DRED, for which no a priori knowledge about the occurrence and duration of the drift can be assumed. It has been used in several relevant studies of drift detection such as [26], [29], [50], and including studies of online approaches such as [3], [8], [27]. This data set was collected from the Australian New South Wales Electricity Market, where prices are not fixed and become affected by the demand and supply dynamics. It contains 45,312 samples dated from 7 May 1996 to 5 December 1998. Each example of the data set refers to a period of 30 minutes, and has 5 fields (day of week, time stamp, NSW electricity demand, Vic electricity demand, and scheduled electricity transfer between states) and the class label. The class label identifies the change of the price related to a moving average of the last 24 hours. The class level only reflect deviations of the price on a one day average and removes the impact of longer term price trends. This research has considered the first month of the data set to tune the parameters of the drift detector and DRED, whereas the following 6 months have been used for prediction and performance assessment.

3.4.2 Traditional Diversity Generation Techniques

The experiments aim at analyzing the behavior of DRED when hybridized with traditional diversity-based online learning ensemble models against their naive implementation. Three different schemes will be considered:

1. The online bagging method proposed in [3], and already introduced in Section 2.5.2, by which varying level of diversity is imposed to an ensemble by presenting a training example (whenever it is available) Z_k times for every member of the ensemble, where Z_k is drawn from a $Poisson(\lambda)$ distribution. The design idea is to enforce a higher/lower diversity level in the ensemble when λ is set to a lower/higher value: when λ is high, each of the compounding learners is more likely to be fed with at least one copy of the input example. In this case, differences between the trained learners are reduced with respect to lower values of λ, by which classifiers within the ensemble are trained with less copies of the input sample, ultimately

imposing more notable differences among their learned patterns. Algorithm 2 summarizes this method, hereafter labeled as BAGGING(λ).

Algorithm 2: Online Bagging BAGGING(λ)

Input : Ensemble $\{\text{GNB}_k\}_{k=1}^{K}$; training example $\{\mathbf{x}_{tr}, l_{tr}\}$; Poisson rate λ
Output: Updated ensemble $\{\text{GNB}_k^{\star}\}_{k=1}^{K}$

for $k = 1$ *to* K **do**
  Initialize the updated ensemble to the input one: $\text{GNB}_k^{\star} = \text{GNB}_k$
  Draw a realization Z_k from a $Poisson(\lambda)$ distribution
  while $Z_k > 0$ **do**
    Train incrementally the k-th learner in the ensemble with the training example: $\text{GNB}_k^{\star}(\{\mathbf{x}_{tr}, l_{tr}\})$
    Decrease the value of Z_k: $Z_k = Z_k - 1$
  end
end

2. The so-called class-switching technique is a powerful way for training diverse ensembles [111]. This approach hinges on using original training samples with randomized class labels: the class label of each example is switched according to a probabilistic threshold p, which bias the learning process of every learner towards stability (no label change) or plasticity (label change). This scheme is described in Algorithm 3 and will be referred to as SWITCHING(p_{th}).

3. The online learning method proposed in [69], based on the batch boosting algorithm AdaBoost [70], is similar to the bagging one (BAGGING(λ)) but with the difference that when a base learner misclassifies a training example, the Poisson distribution parameter λ associated with that example is increased when presented to the next base learner; otherwise it is decreased. Similarly to AdaBoost, the online boosting algorithm gives those examples wrongly classified by one stage half the total weight in the next stage, whereas the correctly classified examples are given the remaining half of the weight.

Algorithm 3: Class Switching SWITCHING(p_{th})

Input : Ensemble $\{\text{GNB}_k\}_{k=1}^{K}$; training example $\{\mathbf{x}_{tr}, l_{tr}\}$; threshold p_{th}
Output: Updated ensemble $\{\text{GNB}_k^{\star}\}_{k=1}^{K}$

```
for k = 1 to K do
    Draw p uniformly at random from [0, 1]
    if p ≤ p_th then
        Train incrementally the k-th learner in the ensemble
        with the training example with its original label:
        GNB_k({x_tr, l_tr})
    else
        Train incrementally the k-th learner in the ensemble
        with the training example with its changed label:
        GNB_k({x_tr, not l_tr})
    end
end
```

When used for concept drift adaptation, the diversity generation techniques (BAGGING(λ), SWITCHING(p_{th}) and BOOSTING(λ)) operate as follows: before the drift, they use a *low* diversity ensemble (driven by parameters λ^{low} for BAGGING and BOOSTING, and p_{th}^{low} for SWITCHING), whereas after a drift occurs they shift to a *high* diversity ensemble controlled by λ^{high} for BAGGING and BOOSTING, and p_{th}^{high} for SWITCHING during a window of 100 time steps for the four synthetic data sets (CIRCLE, LINE, SINEH, and CIRCLE). This period represents the moment in which *high* diversity ensembles are more likely to outperform *low* diversity models, according to the empirical evidence presented in [25]. In the case of SEA data set, this period is set to 500 time steps. After this window, they return to their *low diversity* mode. However, this procedure implicitly requires a fine tuning of the parameter set $\{\lambda^{low}, \lambda^{high}\}$ (corr. $\{p_{th}^{low}, p_{th}^{high}\}$), as well as a priori estimation of the duration of the transition between concepts so as to adjust properly the number of time steps over which the high diversity ensemble is used. In the case of DRED hybrids (labeled as DRED-OB, DRED-CS, and DRED-BOOST), they only use one ensemble for *low* (stable concept) and *high* diversity phases, but trained in *low* diversity mode (also driven by parameters λ^{low} for DRED-OB and DRED-BOOST, and p_{th}^{low} for DRED-CS) and *high* diversity mode (controlled by λ^{high} for DRED-OB and DRED-BOOST, and p_{th}^{high} for DRED-CS).

Algorithm 4: Online Boosting `BOOSTING`(λ)

Input : Ensemble $\{\text{GNB}_k\}_{k=1}^{K}$; training example $\{\mathbf{x}_{tr}, l_{tr}\}$; Poisson rate for each sample λ_{tr}; $\lambda_m^{sc} = 0$; $\lambda_m^{sw} = 0$

Output: Updated ensemble $\{\text{GNB}_k^{\star}\}_{k=1}^{K}$

for $k = 1$ *to* K **do**
  Initialize the updated ensemble to the input one: $\text{GNB}_k^{\star} = \text{GNB}_k$
  Draw a realization Z_k from a $Poisson(\lambda)$ distribution
  while $Z_k > 0$ **do**
    Train incrementally the k-th learner in the ensemble with the training example: $\text{GNB}_k^{\star}(\{\mathbf{x}_{tr}, l_{tr}\})$
    Decrease the value of Z_k: $Z_k = Z_k - 1$
  end
  if $GNB_k^{\star}(\{\mathbf{x}_{tr}, l_{tr}\})$ is the correct label **then**
    $\lambda_m^{sc} = \lambda_m^{sc} + \lambda_{tr}$
    $\epsilon_m = \lambda_m^{sw} / (\lambda_m^{sc} + \lambda_m^{sw})$
    $\lambda_{tr} = \lambda_{tr} / (2(1 - \epsilon_m))$
  else
    $\lambda_m^{sw} = \lambda_m^{sw} + \lambda_{tr}$
    $\epsilon_m = \lambda_m^{sw} / (\lambda_m^{sc} + \lambda_m^{sw})$
    $\lambda_{tr} = \lambda_{tr} / (2\epsilon_m)$
  end
end

After the window, they use their *low* diversity mode again.

While the *stability* phase entails accumulating knowledge regarding the supposedly stationary phase underlying concept, *plasticity* phase requires forgetting some or all of the old acquired knowledge in order to learn the new upcoming concept. In [21] the authors propose to eliminate the worst learner by picking randomly a learner from the subset of the K' ($K' < K$) worst base learners. They conclude that a subset size of $K' = 1$ in the *stability* phase and $K' = K$ for the *plasticity* phase is the best selection. We adopt this criterion to set the value of K', but removing the less/more diverse learner in the *plasticity/stability* phase instead of using the performance of the learner.

In what refers to the quantification of the predictive performance, discussions will be centered on the so-called *prequential accuracy* metric, first proposed in [129] and frequently used by the online learning

community (e.g. [8]). This score quantifies the average accuracy obtained by the prediction of each test example before its learning in an online test-then-train fashion, and is given by:

$$preACC(t) = \begin{cases} preACC_{ex}(t), & \text{if } t = t_{ref}, \\ preACC_{ex}(t\text{-}1) + \dfrac{preACC_{ex}(t)\text{-}preACC_{ex}(t\text{-}1)}{t\text{-}t_{ref}+1}, & \text{otherwise,} \end{cases} \tag{3.9}$$

where $preACC_{ex}(t) = 0$ if the prediction of the test example at time t before its learning is wrong and 1 if it is correct; and t_{ref} is a reference time that fixes the first time step used in the calculation. This reference time allows isolating the computation of the prequential accuracy before and after a drift has started. Statistics of the obtained results will be extracted based on 30 independent runs for each data set and technique.

3.4.3 Estimation of Parameters

Parameters λ^{high}, p_{th}^{high} and K are inherent to any diversity-based adaptation technique. They may vary depending on the base learner and other factors, yielding different values reported and used the state of the art. In the case of the parameter W, and as it will be further explained in Section 3.5, is inherent to any drift detection technique based on a reservoir of past samples to study the distribution of data, but in the case of DRED serves as a source to generate synthetic samples. Only the parameter h (the bandwidth for the KDE technique), and those for the MOT (size of population, number of generations and number of synthetic samples) are unique to DRED.

In order to find the suitable values for the *high* diversity phase (p_{th}^{low} and λ^{low} are always set to 1.0 as done in [8]), an off-line experiment was carried out previously using a GNB ensemble for each data set:

1. Perform 30 independent runs for every traditional diversity technique using $\lambda^{high} \in \{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$ and $p_{th}^{high} \in \{0.6, 0.7, 0.8, 0.9\}$. In order to encourage different levels of diversity for `BAGGING` and `BOOSTING`, the Poisson distribution can adopt different values depending on the diversity phase. In the case of `SWITCHING`, a probability lower than 0.6 causes that the ensemble is too misfitted due to an excessive number of switched classes for training.

2. Select those values which show a better average prequential accuracy after the drift occurs during a period of 100 (CIRCLE, LINE, SINEH, SINEV), 500 (SEA), and 48 (ELEC2) time steps. The configuration leading to the best average value for the four combinations (severity/speed) of every data set is selected as the result set included in the benchmark (Table 3.1).

In the case of BAGGING and BOOSTING, higher/lower λ values are associated with lower/higher average Bhattacharyya distance (lower/higher diversity), whereas for SWITCHING higher/lower p_{th} values are associated with lower/higher average Bhattacharyya distance (lower/higher diversity). In order to find the suitable number of models in the ensemble (K) for all approaches, and the reservoir size (W) and the bandwidth parameter (h) for DRED hybrids, an experiment with different values over all of the four data sets was carried out:

1. Perform 30 independent runs combining $K \in \{5, 10, 20\}$, $W \in \{10, 20, 40\}$ for CIRCLE, LINE, SINEH, SINEV, and SEA, and $W \in \{96, 144, 192\}$ for ELEC2, and $h \in \{1, 2, 3\}$ for each data set.

2. Determine the prequential accuracy obtained by each run after the drift occurs and during a window of 100 (CIRCLE, LINE, SINEH, SINEV), 500 (SEA), and 48 (ELEC2) time steps, to check the best value of K for traditional approaches, and the best combination of K, W and h for DRED hybrids.

3. Select the best values of K, W and h that showed a better average performance in all data sets. These values are shown in Table 3.1.

Regarding the rest of the parameters setting of DRED, simulations will be run with neighbor-based KDE models with Gaussian kernels. The reservoir size (W of Table 3.1) is the source from which $N^{\boxplus} = 30$ synthetic samples are drawn and fed to the MOT for their optimal labeling, and $N^{\boxtimes} = 100$ samples are equally extracted for estimating the predictive performance within the MOT operation. For the sake of computational efficiency DRED is configured with an off-the-shelf NSGA-II solver with a minimal configuration (population size of $Pop = 10$ individuals and $Gen = 25$ generations), binary tournament selection, uniform crossover with probability $P_c = 0.5$ and random mutation with probability $P_m = 0.1$. These values are in line with typical parameter settings configured for this solver in the literature.

	BAGGING		SWITCHING		BOOSTING		DRED-OB				DRED-CS				DRED-BOOST			
	λ^{high}	K	p_{th}^{high}	K	λ^{high}	K	λ^{high}	K	W	h	p_{th}^{high}	K	W	h	λ^{high}	K	W	h
CIRCLE 1-1	0.5	10	0.6	20	0.005	10	0.5	5	10	2	0.6	5	10	1	0.005	5	10	1
CIRCLE 1-3	0.5	5	0.6	5	0.005	10	0.5	5	20	1	0.6	5	40	1	0.005	5	40	1
CIRCLE 3-1	0.5	5	0.6	10	0.005	5	0.5	20	10	3	0.6	5	20	3	0.005	5	10	2
CIRCLE 3-3	0.5	5	0.6	5	0.005	10	0.5	5	40	1	0.6	5	40	2	0.005	20	40	1
LINE 1-1	0.05	20	0.8	5	0.1	5	0.05	10	20	3	0.8	5	10	1	0.1	5	20	3
LINE 1-3	0.05	10	0.8	5	0.1	10	0.05	5	40	2	0.8	5	40	1	0.1	5	40	1
LINE 3-1	0.05	5	0.8	5	0.1	5	0.05	5	10	2	0.8	5	10	1	0.1	10	10	3
LINE 3-3	0.05	5	0.8	5	0.1	10	0.05	10	10	2	0.8	10	10	1	0.1	5	10	1
SINEH 1-1	0.1	10	0.6	20	0.005	20	0.1	5	10	1	0.6	5	10	1	0.005	5	20	1
SINEH 1-3	0.1	10	0.6	5	0.005	5	0.1	10	10	2	0.6	5	40	2	0.005	10	40	1
SINEH 3-1	0.1	5	0.6	5	0.005	20	0.1	10	40	1	0.6	5	40	1	0.005	5	20	2
SINEH 3-3	0.1	10	0.6	5	0.005	5	0.1	10	10	2	0.6	5	10	2	0.005	10	40	1
SINEV 1-1	0.1	20	0.6	5	0.1	5	0.1	5	10	3	0.6	20	20	1	0.1	5	20	1
SINEV 1-3	0.1	5	0.6	5	0.1	5	0.1	5	20	1	0.6	5	5	1	0.1	5	20	1
SINEV 3-1	0.1	5	0.6	5	0.1	5	0.1	5	10	3	0.6	5	5	1	0.1	5	10	1
SINEV 3-3	0.1	5	0.6	5	0.1	5	0.1	5	40	1	0.6	5	5	2	0.1	5	10	1
SEA	0.0005	10	0.6	10	0.0005	10	0.0005	10	20	3	0.6	10	20	3	0.0005	10	20	3
ELEC2	0.1	5	0.8	20	0.5	5	0.1	5	192	1	0.6	20	192	1	0.1	10	192	1

TABLE 3.1: Parameters of the traditional diversity generation schemes and DRED hybrids for the synthetic data sets. Following the notation in [8], the first number (i.e. circle 3-1) refers to the severity and the second one to the speed, being the combinations as follows: 3 represents the maximum and 1 the minimum severity respectively, whereas 3 represents the minimum and 1 the maximum speed respectively.

3.4.4 Complexity Analysis

The implementation of DRED utilizes NSGA-II as its core MOT, whose complexity is known to be dominated by the non-dominated sorting part of the algorithm [114]. This operation has a complexity per iteration equal to $\mathcal{O}(FPop^2)$, where F is the number of objectives (two in our case) and Pop is the population size. In DRED this parameter is set to $Pop = 10$ individuals (i.e. a small population size), indeed aimed at reducing the impact of the solver on the overall proposed method. Other elements of relevance for the complexity of the MOT are the number of models in the ensemble K and the number of synthetic samples for training $N^{\boxplus}$, which jointly set the number of optimization variables per individual and consequently, the number of times the evolutionary operators (e.g. crossover and mutation) are applied. Finally, the number of class labels L does not impact on the complexity of the NSGA-II solver per iteration, as opposed to the computation of the diversity measure as per Expression (3.5), whose complexity is $\mathcal{O}(LK^2)$.

The analysis on the complexity of DRED is concluded by the

KDE model, whose naive implementation requires $\mathcal{O}(WN^{\boxplus})$ evaluations and $\mathcal{O}(WN^{\boxplus})$ multiplications and additions for $N^{\boxplus}$ test points based on a kernel density estimated from W samples [106]. Scales handled in this chapter for these parameters ($W \leq 40$ samples and $N^{\boxplus} = 30$) are affordable for current computation technologies under demanding online settings.

3.5 Results and Discussion

In this section we analyze the behavior of the traditional diversity generation schemes (`BAGGING`, `SWITCHING`, and `BOOSTING`) in comparison with their versions powered by DRED (`DRED-OB`, `DRED-CS`, and `DRED-BOOST`). Average prequential accuracy results for all data sets and techniques are listed in Appendix A, discussing in what follows those cases from which more relevant conclusions can be drawn.

We start the discussion by analyzing the comparison between the approaches powered by DRED and the traditional diversity generation schemes in the `CIRCLE` data set. A first look on Figures 3.4, 3.5 and 3.6 reveals that DRED hybrids outperforms the traditional ones in almost all cases (stability and plasticity periods). Not only it recovers faster from the drift, but also attains a higher accuracy level in the stability period after the drift. The reason for this dominant behavior stems from several aspects: to begin with, the dynamics of the drift and the data set itself are characterized by circularly shaped regions, shown in Figures 3.7.a and 3.7.b. Such regions are suitable for Gaussian kernel fitting. Furthermore, drift occurs fast, and the new concept is built incrementally from the old one and uniformly across all features, which again favors the generation of synthetic samples using the KDE approach with bandwidth tuning proposed in DRED. Indeed, the effect of h in the learned density of every class is to smooth uniformly the overall kernel aggregation. Since the new concept is modeled as one of the classes expanding its decision region over the feature space, the dynamics of the drift are matched to the effect of h when generating new synthetic samples for diversifying the ensemble. Interesting is also to remark that the significantly better accuracy score attained by DRED long after the drift is a byproduct of the fact that it is run only once after the drift has been detected: besides lowering dramatically the computational cost of this approach, it also allows the ensemble to quickly

capture the new concept, letting the learning process of the new concept itself reduce the diversity gained after DRED as the new data distribution becomes stable.

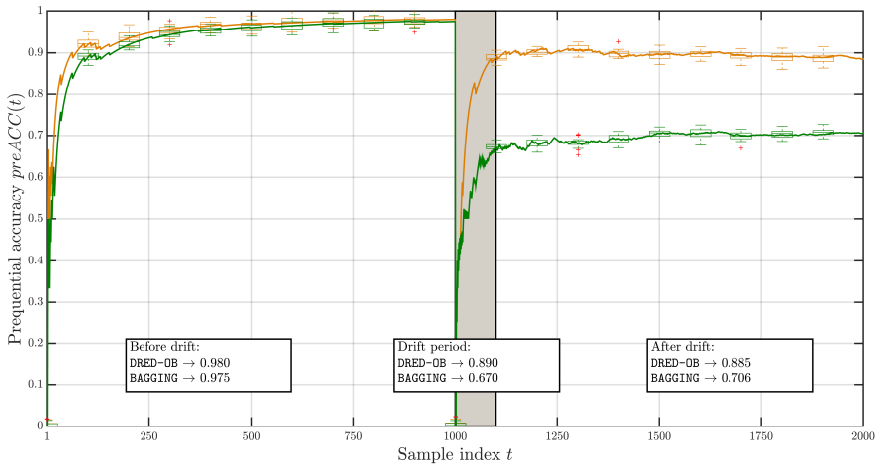


FIGURE 3.4: Prequential accuracy $preACC(t)$ of DRED-OB (—) and BAGGING (—) for the CIRCLE data set under high severity and speed levels.

Regarding the parameters tuning, the results show that a higher bandwidth ($h \in \{2, 3\}$) is needed when high severity and speed conditions are present, which is totally aligned with our previous explanations. Besides, it seems that a bigger window ($W = 40$) is required when the drift is slower, which is expected due to the necessity of more information to generate synthetic samples in order to adapt to the slow change. In general, a smaller ensemble ($K = 5$) than traditional approaches ($K \in \{10, 20\}$) suffices for outperforming their scores.

We follow the discussion by replicating the above analysis with the LINE and SINEV data sets. In these cases, results reveal that the overall behavior of DRED hybrids is as follows: BAGGING and BOOSTING generally outperforms the traditional approaches in the plasticity period when the drift is fast, and when the drift is slower in the case of SWITCHING. In the stability period after the drift only DRED-BOOST outperforms the traditional scheme when the drift occurs fast. In these two data sets, the dynamics of the drift and the data set itself are characterized by linearly-shaped data patterns that are shown in Figure 3.7.c and 3.7.d (LINE), and 3.7.e and 3.7.f (SINEV). Their class distributions before and after the drift make them not so suitable for kernel fitting than the CIRCLE data set; however, DRED

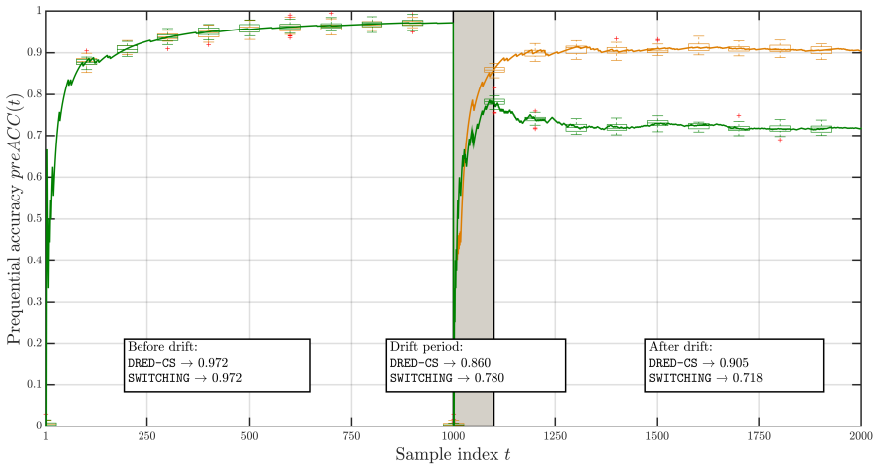


FIGURE 3.5: Prequential accuracy $preACC(t)$ of DRED-CS (—) and SWITCHING (—) for the CIRCLE data set under high severity and speed levels.

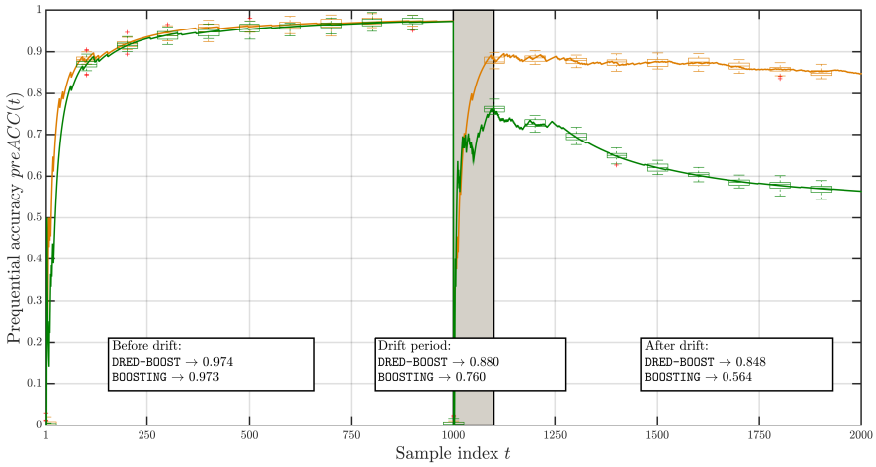
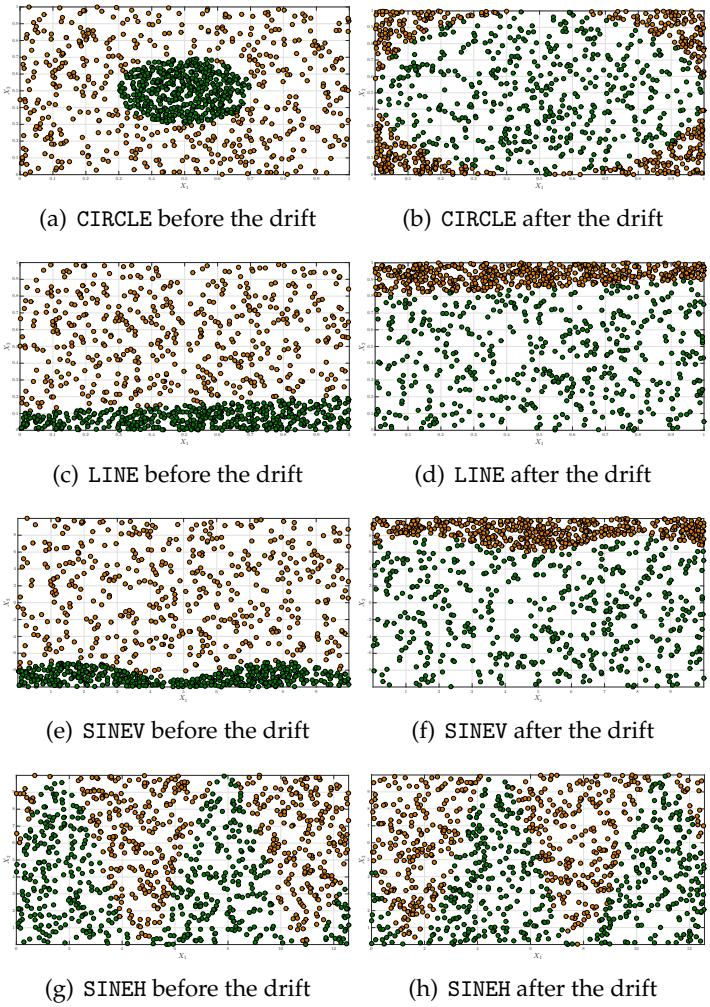


FIGURE 3.6: Prequential accuracy $preACC(t)$ of DRED-BOOST (—) and BOOSTING (—) for the CIRCLE data set under high severity and speed levels.

still performs competitively due to the fact that the new concept again grows incrementally from the previous one, yet only along the vertical feature. All in all, the constituent learners of the ensemble are *pushed* towards the new concept.

In the case of SINEH, only DRED-BOOST outperforms the traditional scheme in the stability periods. This is the data set in which DRED hybrids score worst when compared to the rest of data sets.



(a) CIRCLE before the drift (b) CIRCLE after the drift

(c) LINE before the drift (d) LINE after the drift

(e) SINEV before the drift (f) SINEV after the drift

(g) SINEH before the drift (h) SINEH after the drift

FIGURE 3.7: Feature space evolving after the drift occurs in each data set.

In this regard, Figures 3.7.g and 3.7.h evince that the dynamics of the drift and the data set itself do not grow incrementally with respect to the old concept, nor is the distribution followed by data samples appropriate for Gaussian kernel density estimation. As a result, the synthetic samples generated by DRED are not representative of the new concept and counterproductive when fed to the ensemble. Among all DRED hybrids, DRED-BOOST provides the best overall performance in all data sets.

The SEA data set compiles three drifts with different severities.

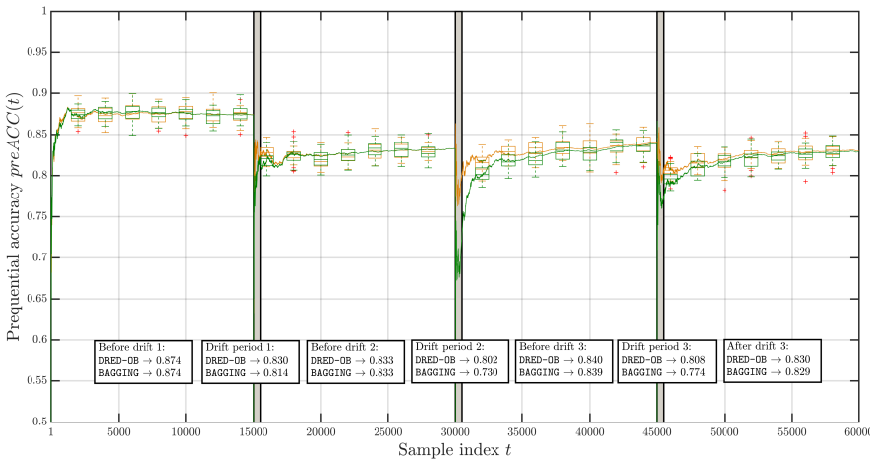


FIGURE 3.8: $preACC(t)$ of DRED-OB (—) and BAGGING (—) for the SEA data set.

In this case, the results reveal that the overall behavior of DRED hybrids is as follows: in the first two drifts, DRED-OB and DRED-BOOST outperform the traditional ones, and only in the case of DRED-CS and DRED-BOOST their performance is lower than traditional approaches in the third drift. DRED-OB is the best hybrid showing the best performance in all the drifts and being as good as the traditional one in the stability periods. The performance of DRED hybrids is depicted in Figures 3.8, 3.9, and 3.10. The behavior of DRED in this data set confirms the fact that the performance of DRED is not punished for the number of drifts in the data stream.

We now delve into the influence of the selected values for W (size of the window) in the performance of DRED hybrids. To this end Figures 3.11.a and 3.11.b unveil that the optimal value for this parameter is tightly liked to the drift dynamics. When the drift occurs fast and implies a severe change of the underlying distribution, the window size should be small enough to capture quickly the new concept and forget the old one. This is indeed what can be seen in Figure 3.11.a, where the performance degrades as W increases. On the contrary, when the drift evolves slowly and the old and new concepts prevail during a longer period, the value of W imposes a trade-off between the reaction time and the quality of the diversity injected by DRED in the ensemble. As shown in Figure 3.11.b, a slow transition between concepts requires tuning W so as to capture and model properly the new concept at an admissible delay in the injection of the generated synthetic samples.

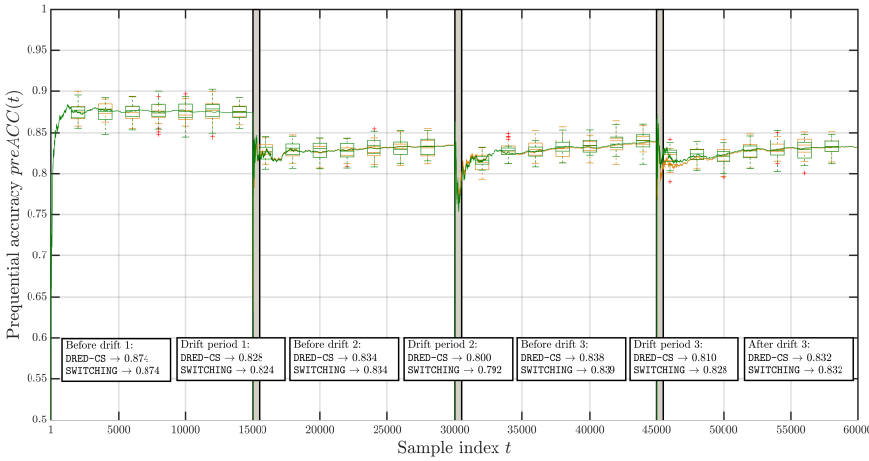


FIGURE 3.9: $preACC(t)$ of DRED-CS (—) and SWITCHING (—) for the SEA data set.

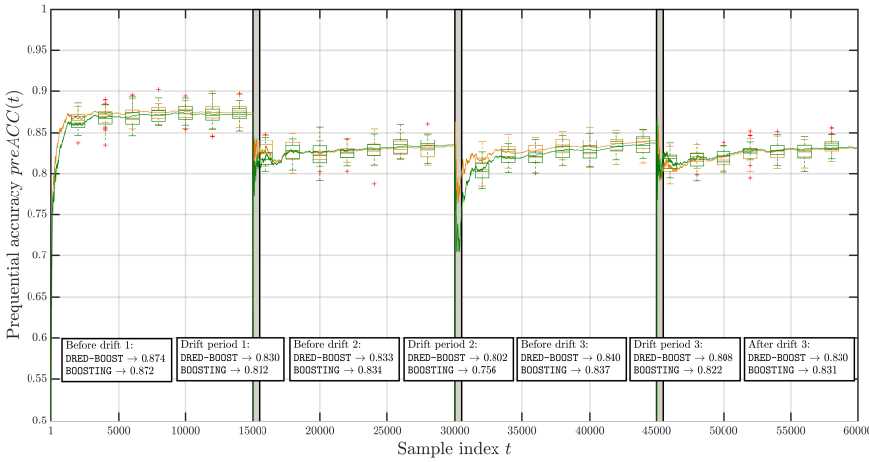
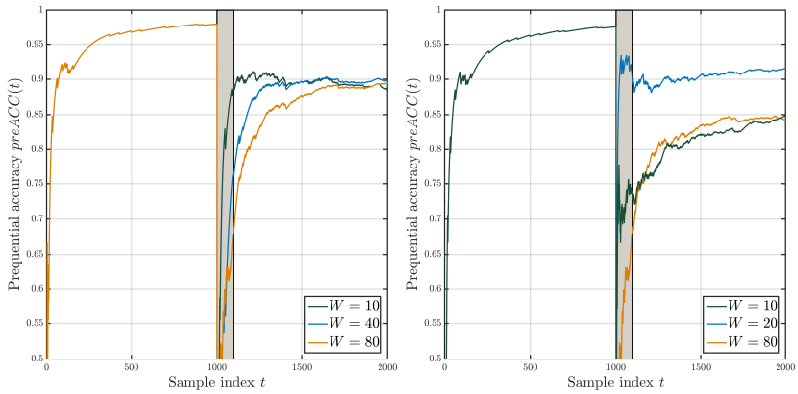


FIGURE 3.10: $preACC(t)$ of DRED-BOOST (—) and BOOSTING (—) for the SEA data set.

We end the discussion highlighting the relevance of the hybridization process of DRED with a drift detector. When dealing with real data sets it is not possible to know a priori the beginning of the drift, its severity, duration or even determine reliably if the observed change is an evidence of a drift start or an isolated outlier. For this reason, firstly the analysis of DRED hybrids has been based on artificial data sets with perfect drift detection. Under these circumstances, for a thorough study of its behavior it is necessary to assume certain information of the drift, which affects the selection of values for the



(a) CIRCLE 3-1 (severe, fast drift) (b) CIRCLE 1-3 (smooth, slow drift)

FIGURE 3.11: Mean prequential accuracy $preACC(t)$ of DRED-OB in CIRCLE 3-1 (a) and CIRCLE 1-3 (b) data sets imposing different values of W. K has been fixed to their best value as per Table 3.1 (i.e. 20 and 5, respectively).

parameters (W, h, duration of the drift, its severity and speed, even the shape of the change in what refers to the features evolution after the change occurs, etc.).

To this end, DRED has been hybridized with the so-called Early Drift Detection Method (EDDM, [50]) and applied to the ELEC2 dataset. A period of 48 time steps that corresponds to one day was assumed to study its behavior during the drift. EDDM has been selected to detect drifts due to its simplicity, its results dealing with abrupt and very slow changes, and its capacity to detect repeatedly occurring concept drifts even with very noisy data. In this case, the first month has been used to tune off-line the values of parameters α (to establish the warning level) and β (to set the drift level), being set to 0.94 and 0.89, respectively. With this configuration, EDDM detects 50 drifts from month 2 to month 7.

Results collected in Table A.6 elucidate that DRED-OB outperforms its naïve counterpart (BAGGING) in the period before the drift and during the drift period, being equally well-performing in the period after the drift. On the other hand, DRED-CS is as good as SWITCHING before and after the drift, but DRED-CS outperforms SWITCHING in the drift period. In the case of DRED-BOOST is as good as BOOSTING in all of the periods. A closer inspection to the obtained results – for example, the performance of DRED-OB and BAGGING in several different drifts detected in this data set, depicted in Figure 3.12 – buttresses

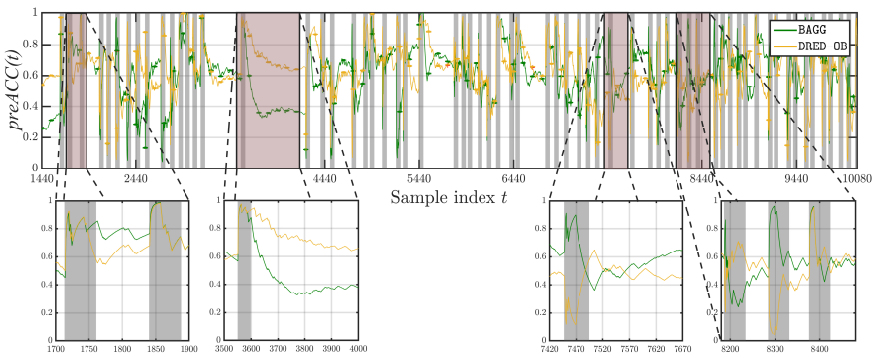


FIGURE 3.12: $preACC(t)$ of DRED-BAGGING (—) and BAGGING (—) for the ELEC2 data set. Average performance scores *during* and *after* specific drifts have been magnified for discussion.

the need for matching the configuration of DRED to the characteristics of the drift at hand.

This insightful conclusion should motivate further research towards deriving drift detectors that could supply DRED with valuable information on the drift. There is a plethora of schemes in the literature such as Drift Detection Method (DDM, [29]) from which DRED could extract information about the size of the window by analyzing the size of the samples that belongs to the warning level prior to the drift occurrence. This warning period, which could be used to set W, can be also inferred by another three detectors: EDIST2 [113], Exponentially Weighted Moving Average (EWMA) [130] and Concept Drift Detection (ECDD). Adaptive Windowing (ADWIN) [27] could also supply DRED with information on the size of the window as it uses sliding windows of variable size, dynamically enlarging the window when there is no apparent change in the context, and shrinking it when a change is detected. Here it is worth mentioning that DRED should delimit the maximum length of the window provided by ADWIN (in turn, the main drawback of this detector). Finally, the output of an adaptive CUSUM [131] could inform DRED on the starting and ending point of the drift, as well as the amplitude (duration) of the drift. All of them, among others, were recently compared in [26] concluding that each drift detection method performs best in data sets with different characteristics.

3.6 Summary

Nowadays fast-arriving information flows lay the basis of many data mining applications. Such data streams are usually affected by non-stationary events that eventually change their distribution (concept drift), causing that predictive models trained over these data become obsolete and do not adapt suitably to the new distribution. Specially in online learning scenarios, there is a pressing need for new algorithms that adapt to this change as fast as possible, while maintaining good performance scores. Recent studies have revealed that a good strategy is to construct highly diverse ensembles towards utilizing them shortly after the drift (independently from the type of drift) to obtain good performance scores. However, the existence of the so-called trade-off between stability (performance over stable data concepts) and plasticity (recovery and adaptation after drift events) implies that the construction of the ensemble model should account simultaneously for these two conflicting objectives. In this regard, this chapter has presented a new approach coined as DRED to artificially generate an optimal diversity level when building prediction ensembles once shortly after a drift occurs. DRED uses a KDE method to generate synthetic data, which are subsequently labeled by means a multi-objective optimization method that allows training each model of the ensemble with a different subset of synthetic samples. Computational experiments have revealed that the proposed approach can be hybridized with other traditional diversity generation approaches, yielding optimized levels of diversity that render an enhanced recovery from drifts.

Chapter 4

Challenge 2: Data Reduction Techniques and New Stream Learning Algorithms

4.1 Introduction

Unfortunately, most off-the-shelf classification models need to be retrained if they are used in a changing environment and fail to scale properly. One of the most promising machine learning techniques in the field that can overcome this noted drawback is the Spiking Neural Network (SNN) [132]. The advent of SNNs was propelled by the need for a better understanding of the information processing skills of the mammalian brain, for which the community committed itself to the development of more complex biologically connectionist systems. SNNs have revealed themselves as one of the most successful approaches to model the behavior and learning potential of the brain, and exploit them to undertake practical learning tasks. In essence SNNs leverage spike information representation so as to construct spike-time learning rules that capture temporal associations between a large number of temporal variables in streaming data. Among other applications (e.g. simulation of space-time systems), such a learned knowledge can be exploited to predict future events. In fact, SNNs must be regarded as a portfolio of models for different computational uses and applications, all inspired by the the same design principles (information encoding and neural processing based on time spikes). One of the successful SNNs is the Evolving Spiking Neural Networks (eSNNs) [133], [134], where the number of spiking neurons evolves incrementally in time to infer temporal patterns from data. First proposed in [135], [136], eSNNs are based on the principles of evolving connectionist systems (ECOS)

and Thorpe's neural model [137]. In SNNs, changes in the input stream data are encoded immediately as binary events - spikes. As we will motivate in forthcoming sections, they use one of the most suitable data encoding strategies for adapting to drifts.

Several works have focused on implementing SNNs for online learning environments. The most early attempt in this regard is SpikeProp, an adaptation of the classical backpropagation algorithm that was the first supervised learning algorithm developed for SNNs [138]. However, it is too slow to be used in an online setting, and prone to get stuck in local minima as a result of its gradient-based learning algorithm. In [139] the authors proposed a derivative-free supervised learning algorithm comprising an evolutionary strategy with a reportedly better performance than SpikeProp, but the training process was extremely time-consuming and hence, not suitable for online learning. ReSuMe [140]–[142] integrated the idea of learning windows with remote supervision; despite this method claimed to be suitable for online learning, the network structure used in this method is fixed and does not adapt to incoming stimuli. In addition, the desired precise output spike timing is crucial to ReSuMe learning [102]. Other studies have addressed the online learning in a more realistic approach. The method proposed in [143] can perform learning in an online mode through synaptic plasticity and adaptive network structure. More recently, the SpikeTemp method proposed in [102] offers an enhanced rank-order-based learning method for SNNs with an adaptive structure where the precise times of incoming spikes are used to determine the required change in synaptic weights. With these few exceptions, there is a lack of efficient and scalable SNN-based algorithms in online learning scenarios.

Interestingly for the scope of this challenge, none of the contributions reviewed above takes into account the requirement of a limited size for the neurons repository, which is of utmost importance to meet the processing requirements of online learning. Should this crucial aspect not be taken into account, the number of neurons in the repository would increase every time step. Therefore, further efforts are still needed to devise new online learning mechanisms for SNNs and increase their applicability to real-world problems. To this end not only the SNN model should learn incrementally from the data stream, but the content of its neurons repository should also be limited in size and adapted when concept drift occurs. To the best of our knowledge, these two issues have not been addressed in the community. The contribution of this thesis with respect to this challenge can be summarized as follows:

1. We develop a new eSNN model that incorporates a set of novel ingredients for efficiently dealing with online learning applications, such as a limitation of the size of the neurons repository and the use of a sliding window. The developed model is able to classify inputs after just one presentation of the training samples, without requiring the entire training set to be available in advance. Regarding the limited size of the neurons repository, the proposed approach will embrace the adoption of Data Reduction Techniques (DRTs) [5], which aim to obtain a representative training set with a lower size when compared to the original one, and with similar or even higher generalization capability when fed to a predictive model. They can be divided into prototype selection (PS) techniques [144], [145], which consist of choosing a subset of the original training data; and prototype generation (PG) techniques [146]–[148], which build new artificial prototypes for a better adjusting of the decision boundaries between classes.

2. We hybridize the proposed eSNNs approach with a drift detector, yielding a solid learning model to be deployed in a realistic online scenario.

3. We analyze the impact of different data reduction techniques in the newly devised eSNN model over a wide range of online learning data sets, with emphasis on the predictive performance of the model after a drift occurs, the data reduction percentage achieved by every DRT, and the implications of the proposed strategy in the future of the online learning field.

4. As a result of this work, we provide an online learning technique based on a single classifier, proven to perform competitively in comparison with other techniques based on ensembles while using less storage capacity. Single classifiers are widely regarded as an attractive solution for mining massive data streams due to their reduced computational cost when compared to their ensemble counterparts [15].

4.2 The Evolving Spiking Neural Networks

An eSNN consists of an encoding part that transforms a real-valued vector into spikes generated over time, a neuron model, and a learning mechanism that calculates the connection weights between the

input and the output neurons. eSNNs are now a part of a comprehensive SNN architecture for spatio-temporal data modeling, NeuCube [149], which can deal with a wide range of applications [150].

4.2.1 Architecture

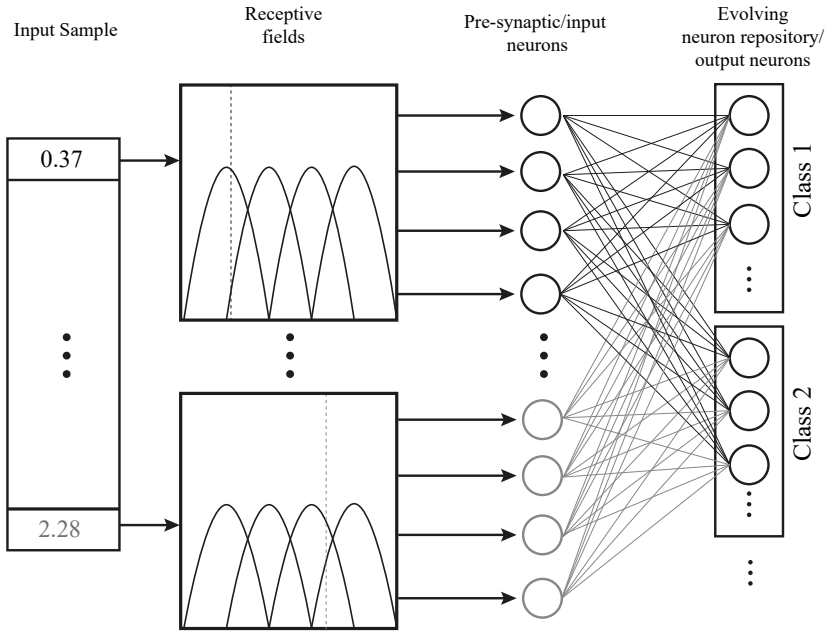


FIGURE 4.1: Architecture of feed-forward eSNN [135].

As depicted in Figure 4.1, the architecture of the eSNN is composed by three layers [135], [136], [143], [151]. The first layer corresponds to the input data. The second layer is for encoding purposes, where the real values of the features of every sample are encoded as trains of spikes, generating the pre-synaptic neurons and each of them having a receptive field. Receptive fields of neighboring neurons overlap with each other by adopting the shape of Gaussian or Logic functions, in all cases covering the whole range of the values of each feature (as explained below). The number of encoding neurons N (or receptive fields) may vary depending on the nature of the data at hand, and must be tuned for achieving a good predictive performance of the overall model. The third layer is the evolving output layer, where a repository of spiking neurons representing

samples (divided in one subrepository per every class in the problem) evolves as new data arrive. Each output neuron is linked to all input neurons through connections whose weights are learned from the data samples fed to the model.

4.2.2 Neural Encoding

In order to learn from real-valued data, each sample is encoded to a sequence of spikes over time (spike trains) by using a neural encoding technique, e.g. rank order population [152], [153] or any other encoding approach alike. The rank order population scheme works on the basis of the order of the spikes across all the synapses connected to the particular neuron. It creates the priority in the input spikes depending on the order of spike arrival to the neuron, which provides extra information to the network regarding the order of the spike [137]. In this chapter we adopt the Gaussian Receptive Field (GRF) population encoding scheme, where the input can be distributed over several neurons with overlapping and graded sensitivity profiles by using Gaussian activation functions. Each encoding neuron is fired only once during the time coding interval T. As a result, each input sample is translated into a spatio-temporal spike pattern. Specifically, the center C_j and the width W_j of each GRF of pre-synaptic neuron j are computed as:

$$C_j = I^n_{min} + \frac{2j-3}{2}\left(\frac{I^n_{max} - I^n_{min}}{N-2}\right), \tag{4.1}$$

and:

$$W_j = \frac{1}{\beta}\left(\frac{I^n_{max} - I^n_{min}}{N-2}\right), \tag{4.2}$$

where N is the number of receptive fields, whose value impacts on the amplitude of the input neuron and must be optimized for the problem. The range of the n-th input variable is assumed to be $\mathbb{R}[I^n_{min}, I^n_{max}]$. Parameter $\beta \in \mathbb{R}[1,2]$ (also referred to as overlap factor) establishes the width of receptive fields, thereby their amount of overlapping and ultimately, the firing time of the pre-synaptic neuron j. The output of neuron j is defined as:

$$output_j = \exp\left(-\frac{(x - C_j)^2}{2W_j^2}\right), \tag{4.3}$$

where x is the input value. The firing time of each pre-synaptic neuron j is defined as:

$$T_j = \lfloor T(1 - output_j) \rfloor \tag{4.4}$$

where T is the simulation or spike interval. Figure 4.2 exemplifies the GRF encoding process for the feature of any given sample.

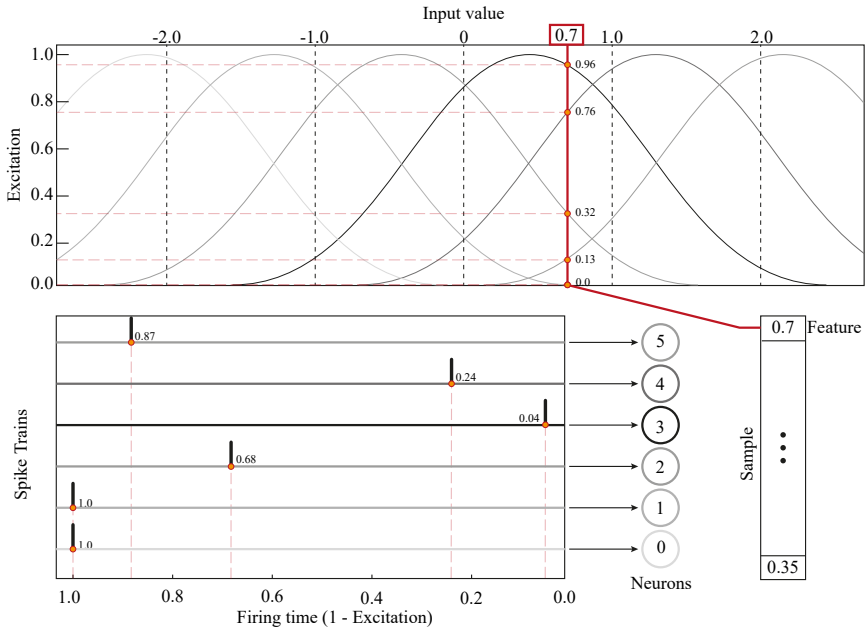


FIGURE 4.2: Example of population encoding based on 6 GRFs. For an input value of 0.7 (bold straight line) the intersection points with each GRF are computed $(0.96, 0.76, 0.32, 0.13, 0.0, 0.0)$, which are in turn translated into firing times $(0.04, 0.24, 0.68, 0.87, 1.0, 1.0)$.

4.2.3 Neural Model

A simplified Leaky Integrate-and-Fire (LIF) model was formally proposed in [137], but the idea can be tracked to publications as early as 1990. LIF states that the spike response of a neuron depends only on the arrival time of pre-synaptic spikes, that is, the earlier the spike arrives to a neuron, the stronger its weight will be when compared to a later spike. Each neuron in this model can spike at most once, and a neuron fires when its PSP reaches its threshold value. The PSP

of a neuron i is defined as:

$$PSP_i = \begin{cases} 0, & \text{if fired,} \\ \sum_j w_{ji} \cdot mod^{order(j)}, & \text{otherwise,} \end{cases} \tag{4.5}$$

where $w_{j,i}$ represents the weight of the synaptic connection between pre-synaptic neuron j to output neuron i; mod is the modulation factor with a range $\mathbb{R}[0,1]$; and $order(j)$ defines the rank of the pre-synaptic neurons spike. The first rank is assigned as 0 and subsequently, rank is increased by 1 based on firing time of each pre-synaptic neuron.

4.2.4 Supervised Learning

When utilized in a supervised learning setting, the aim of the eSNN learning method is to produce and update a repository of output neurons, each of them labeled with a certain class label.

In this classification context, the eSNN training algorithm is algorithmically described in Algorithm 5. First, the model creates a repository of output neurons for the training patterns. For each pattern that belongs to a same given class, a new output neuron is created and connected to all pre-synaptic neurons in the previous layer through weights w_{ji} (see Figure **??**). The value of $w_{j,i}$ is calculated based on the spike order through a synapse j as $w_{j,i} = mod^{order(j)}$, where j is the pre-synaptic neuron of the output neuron i (line 7). A numerical threshold γ_i is set for the newly created output neuron as the fraction $C \in \mathbb{R}(0,1)$ of its maximum post-synaptic potential $PSP_{max,i)}$, i.e. $\gamma_i = PSP_{max,i} \cdot C$. The weight vector of a newly created output neuron is then compared with the already trained output neurons in the repository. If the Euclidean distance between the newly created output neuron weight vector and that of anyone of the already trained output neurons is smaller than a similarity parameter (SIM), they are considered to be similar. As a result, their thresholds and weight vectors are merged according to:

$$w_{j,i} = \frac{w_{new} + (w_{j,i} \cdot M)}{M+1}, \tag{4.6}$$

and:

$$\gamma_i = \frac{\gamma_{new} + (\gamma_i \cdot M)}{M+1}, \tag{4.7}$$

where M is the number of previous merges of similar neurons during the learning history of the eSNN. After merging, the weight vector of the newly created output neuron is discarded, and the new pattern is presented to the model. If none of the already trained neurons in the repository is found to be similar (as per the *SIM* parameter) to the newly produced output neuron, then it is added to the repository.

Algorithm 5: eSNN algorithm

```
Initialize neurons repository, NR = {}
Set eSNN parameter mod = [0,1], C = [0,1], SIM = [0,1]
Set eSNN encoding parameters β, T, N
Calculate I_max, I_min for the training data set
for every sample s belonging to class c do
    Calculate C_j and W_j for encoding s into firing time of
     multiple pre-synaptic neurons j
    Create a new output neuron i and the connection weights
     as w_{j,i} = mod^{order(j)}
    Calculate PSP_{max(i)} = Σ_j w_{j,i} · mod^{order(j)}
    Compute PSP threshold value γ_i = PSP_{max(i)} · C
    if min distance(Newly output neuron weight vector,Neurons
     repository weight vectors in in NR)) ≤ SIM then
        Update the weight vector and threshold of the most
         similar neuron w_{j,i} = (w_new + (w_{j,i} · M)) / (M+1) and γ_i = (γ_new + (γ_i · M)) / (M+1)
        Set M = M + 1
    else
        Add the weight vector and threshold of the new
         output neuron to NR
    end
end
Go to 1 and repeat for all target classes
```

The testing phase propagates the spikes that encode the test sample to all trained output neurons. The class label for the test sample is assigned according to the class label of the output neuron which has fired first after reaching its threshold value γ_i.

4.2.5 eSNN in Online Learning

The neural model of eSNN models allows for a very fast real-time simulation of large networks and a low computational cost. These properties make eSNNs a very suitable candidate for online learning scenarios, where stringent restrictions on computational cost and processing time prevail. Besides, the evolving nature of the network makes it possible to accumulate knowledge as data become available, without the requirement of storing and retraining the model with past samples. We will later evince how this evolving nature is also useful for adapting the model to eventual drifts along the stream.

Several approaches have been developed so far in order to adapt eSNNs to online learning setups [136], [142], [154], [155]. However, most of them are unable to predict inputs after just one presentation of the training samples, hence requiring the entire training set to be available in advance. The online learning field has not certainly been as thoroughly addressed in the eSNN literature as its offline (batch) counterpart. The reason for this lack of research lies on the aforementioned computational restrictions, which require adapting the original eSNN learning algorithm to meet these design constraints.

In online learning scenarios tailored predictive scores metric are often proposed to shed light not only on the net accuracy of online learning models, but also to quantitatively analyze its reaction capability against changes in the data streams. Discussions in this chapter will follow the common practice in this research area by embracing the so-called *prequential accuracy* metric, already defined in Equation (3.9) and widely used by the community [8]. This metric quantifies the sample-by-sample progression of the average accuracy obtained by a learning model in a test-then-train basis (i.e. null verification latency).

4.3 Data Reduction Techniques

In eSNN, if the Euclidean distance between the newly created output neuron weight vector and that of anyone of the already trained output neurons is smaller than *SIM*, they are considered to be close and they are merged. Closeness is decided based on the similarity between the weight vectors $\{w_{j,i}\}$ of every output neuron i to the weight vector produced by the newly input sample. This distance-based prediction strategy can be regarded as that of the k-Nearest Neighbors (kNN), one of the most utilized algorithms in machine

learning due to its simplicity and effectiveness. However, kNN models are known to suffer from several drawbacks [156]:

- The need for a high storage capacity to retain the set of training samples so as to perform the decision rule;
- the computational burden associated to the search for the closest example, due to multiple similarity computations between the test sample and the training examples; and
- the low tolerance of to noisy samples within the training data.

The literature is rich in methods proposed to tackle the above drawbacks. In this chapter we will embrace DRTs to simultaneously face all such issues in an online setting. DRTs aim to obtain a representative training set with a lower size compared to the original one, yet with similar or even higher classification accuracy for new incoming data. DRTs can be classified depending on whether the method at hand is based on filtering and selecting samples from the training set (prototype selection, PS) or, alternatively, on synthesizing representative examples therefrom (prototype generation techniques, PG). Next we overview both categories and show how DRTs can be applied to reduce the number of output neurons in the repository providing a useful mechanism to limit the size of the repository.

4.3.1 Prototype Selection Techniques

PS techniques select a subset of the original training data for constructing the model, hence discarding the remaining data samples. The main advantage of these techniques is their capacity to discriminate relevant examples without synthesizing artificial data. A widely used categorization of PS techniques include edition, condensation, and hybrid methods [157]. Edition methods remove noisy samples in order to increase the classification performance. Condensation methods remove superfluous samples that do not affect the classification performance. Hybrid methods are based on combining edition and condensation methods to yield a PS technique leveraging specific computational and/or performance aspects of both approaches.

The exhaustive study of PS techniques presented in [157] shows the advantages and disadvantages of all DRT methods falling in this category. Indeed this work empirically proved that the choice of a certain method depends on diverse factors, in essence a multi-criteria decision that becomes even more crucial when dealing with

online learning in the presence of concept drift. For example, an edition method usually outperforms a naive kNN in the presence of noise, but only a few samples will be removed [157]. However, although they allow for a high data reduction rate while preserving the model accuracy, edition-based PS techniques are usually the slowest ones due to their greedy search procedure. On the contrary, condensation PS approaches are fast and achieve high data reduction rates, but they usually render classification performance scores lower than those of naive kNN schemes [157].

Figures 4.3 and 4.4 show the behavior of 4 different PS techniques on a synthetic data set, illustrating the nature of decision boundaries after applying sample reduction techniques.

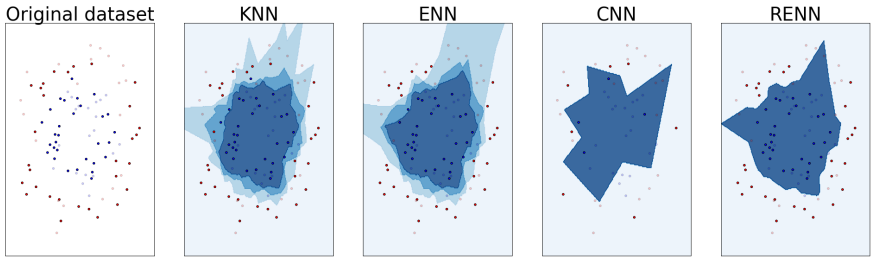


FIGURE 4.3: A comparison of the mean accuracies (95.00%, 82.50%, 87.50%, 77.50%) and data reduction percentages (0.00%, 11.67%, 78.33%, 16.67%) of kNN, ENN, CNN, and RENN techniques respectively on a synthetic data set. The figure shows training points in solid colors and testing points semi-transparent.

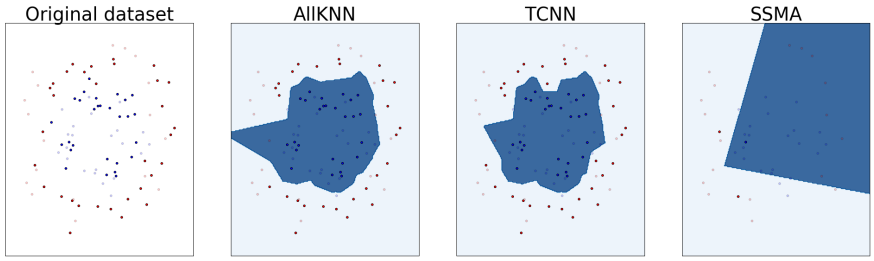


FIGURE 4.4: A comparison of the mean accuracies (92.50%, 90.00%, 85.00%) and data reduction percentages (18.33%, 10.00%, 91.67%) of AllKNN, TCNN, and SSMA techniques respectively on a synthetic data set. The figure shows training points in solid colors and testing points semi-transparent.

4.3.2 Prototype Generation Techniques

Prototype generation techniques build new artificial prototypes to better adjust the decision boundaries between classes in kNN classifiers. To this end, PG methods produce and replace training data samples with new artificial data filling regions in the domain of the problem lacking representative samples in the original data set. The thorough categorization and empirical assessment of PG techniques reported [146] drew similar conclusions to those obtained for PS schemes in [157]: a categorical claim cannot be made in regards to the comparative performance of different PG techniques: the choice of one approach or another will roughly depend on the problem under consideration.

Figure 4.5 shows the behavior of 3 different PG techniques on a synthetic data set, illustrating the nature of decision boundaries after applying the sample reduction techniques.

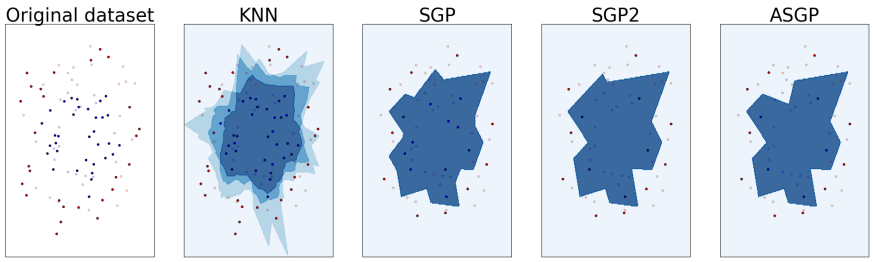


FIGURE 4.5: A comparison of the mean accuracies (95.00%, 92.50%, 85.00%, 90.00%) and data reduction percentages (0.00%, 65.00%, 81.67%, 75.00%) of kNN, SGP, SGP2 and ASGP techniques respectively on a synthetic data set. The figure shows training points in solid colors and testing points semi-transparent.

As it will be further explained, the reduction power of DRTs at the same time that their accuracy remains competitive will be used in favor of our proposed approaches.

4.4 Proposed Approach: Online Evolving Spiking Neural Networks (OeSNN)

As in other online learning algorithms, the proposed online version of eSNN (OeSNN) stores a reduced amount of samples taken over a $\mathcal{W}$-sized sliding window. In our case, however, such samples are

not used for statistical tests [27], but rather for performing the neural encoding procedure described in Section 4.2.2. Every time a new sample arrives in the stream, the neural encoding is performed for the samples falling in $\mathcal{W}$. This encoding is also used for the prediction of the test sample, following a *test-then-train* scheme [158]. The value of $\mathcal{W}$ can be 1) set fixed, e.g. whenever a new sample arrives it is stored in the memory and the oldest one is discarded; or 2) adjusted over time depending on e.g. the information provided by a drift detector analyzing the statistical characteristics of the stream. A fixed size is often adopted as a baseline scheme when evaluating new online learning algorithms [10]; the study will follow this common practice from the related literature.

As explained in previous sections, the learning procedure of the eSNN relies mostly on the neurons repository. Therefore, the size of this repository should be upper bounded in order to meet the restrictive storage constraints imposed in an online learning scenario. The neurons repository collects all the available knowledge in the form of output neurons, so that the more knowledge (neurons) is stored in this model stage, the more likely it will be to find an output neuron similar to the one under test, and ultimately the better accuracy will be achieved. Consequently, the proposed OeSNN approach utilizes a fixed size of the neurons repository NR_size: the new output neuron produced by the test sample through the eSNN structure is stored whenever there is room in the repository, i.e. its current occupation CNR_size is below the net capacity of the reservoir NR_size; if there is no free space (corr. $CNR\_size = NR\_size$), the oldest output neuron will be replaced by the new output neuron. Intuitively, this repository updating strategy addresses two different constraints in online learning under concept drift: the need for a limited size reservoir of output neurons, and an inherent forgetting mechanism to discard outdated concepts when data streams are non-stationary.

Other related studies usually divide the data set into training and testing phases, applying the online learning to the test part after once a well-trained eSNN classifier has been attained [102], [103], [134]. However, in fast streaming scenarios the algorithm must update itself one sample at a time from the beginning of the data streaming process. This is accomplished by adopting incremental methods for warm-start model training while predicting test samples in parallel. This will be the scheme adopted in this chapter.

Algorithm 6 reflects the adaptations made to the original eSNN in Algorithm 5 in order to deploy it in online learning scenarios

by fulfilling with the restrictions described before. The aforementioned $\mathcal{W}$-sized sliding window yields a group of recent samples from which the encoding parameters are computed (lines 4, 8, and 9). The neurons repository is limited to a fixed size (line 5), which is checked in order to decide whether the recent sample can be stored directly or instead, replaces the oldest output neuron in the repository.

At this point it is important to underline that every time a merging process is performed, only two neurons are involved at most. Therefore, it is likely that the output neurons repository stores redundant information when processing a stream, with emphasis during those periods where the data distribution remains stable, i.e. before the drift occurs and long after the drift event. Here lies the rationale for further optimizing the information stored within the OeSNN neurons repository by applying data reduction techniques, which is exposed in the next section.

4.4.1 Data Reduction for OeSNN Models: OeSNN-PS and OeSNN-PG

The proposed OeSNN approach sketched in Algorithm 6 is the basis for the OeSNN models hybridized with DRTs schemes (OeSNN-PS and OeSNN-PG), in which PS and PG techniques are applied respectively on the neurons repository. For this purpose we have selected a wide portfolio of DRTs:

- OeSNN-PS, all based on a majority voting between the k most similar samples to a given unseen observation (the value of k has to be defined beforehand):
  - Edited Nearest Neighbor (ENN) [159], which is a modified editing version of the kNN rule, applies a NN algorithm and *edits* the data set by removing samples which do not agree *enough* with their neighborhood. For each sample in the class to be undersampled, the set of nearest neighbors are computed; if the selection criterion is not fulfilled, the sample is removed.
  - Repeated Edited Nearest Neighbor (RENN) [160] extends ENN by repeating the algorithm multiple times so that more data samples are deleted.
  - Condensed Nearest Neighbor (CNN) [161] was suggested as a rule which retains the basic approach of the NN rule,

Algorithm 6: Proposed OeSNN algorithm

Initialize neuron repository, $NR = \{\}$
Set eSNN parameter $mod = [0,1], C = [0,1], SIM = [0,1]$
Set eSNN encoding parameters β, T, N
Set sliding window size $\mathcal{W}$
Set neuron repository of size NR_size
Set current neuron repository size $CNR\_size = 0$
for *every sample s belonging to the class c* **do**
    Update sliding window with sample s
    Calculate I_{max}, I_{min} for the $\mathcal{W}$ samples in the sliding window
    Calculate C_j and W_j over the sliding window for encoding s into firing time of multiple pre-synaptic neurons j
    Create a new output neuron i and the connection weights as $w_{ji} = mod^{order(j)}$
    Calculate $PSP_{max(i)} = \Sigma_j w_{ji} \cdot mod^{order(j)}$
    Get PSP threshold value $\gamma_i = PSP_{max(i)} \cdot C$
    if min *distance(Newly output neuron weight vector,Neurons repository weight vectors in in NR))* $\leq SIM$ **then**
        Update the weight vector and threshold of the most similar neuron $w_{j,i} = \frac{w_{new}+(w_{j,i} \cdot M)}{M+1}$ and $\gamma_i = \frac{\gamma_{new}+(\gamma_i \cdot M)}{M+1}$
        Set $M = M + 1$
    else
        if $CNR\_size < NR\_size$ **then**
            Add the weight vector and threshold of the new output neuron to NR
            $CNR\_size = CNR\_size + 1$
        else
            Remove the oldest weight vector and its threshold, and put the new ones into NR
        end
    end
end
Repeat above for all target classes

but without imposing its stringent storage requirements. CNN picks out points near the boundary between the classes, achieving an important reduction of the sample size while maintaining the underlying distribution. It uses

a 1-NN rule to iteratively decide if a sample should be removed or not. It is important to note that it is sensitive to noise and will add noisy samples.
  - AllKNN [162] differs from RENN in the fact that the number of neighbors of the kNN algorithm is increased at each iteration so as to yield a smoother decision region.
  - Tomek Condensed Nearest Neighbor (TCNN) [163] removes every pair of samples $\mathbf{x}$ and $\mathbf{x}'$ of different class that form a Tomek link, i.e. whenever there is no other sample $\mathbf{z}$ such that $d(\mathbf{x}, \mathbf{z}) < d(\mathbf{x}, \mathbf{x}')$ or $d(\mathbf{x}', \mathbf{z}) < d(\mathbf{x}, \mathbf{x}')$, where $d(\cdot, \cdot)$ is the distance measure of the problem at hand.
  - Steady-State Memetic Algorithm (SSMA) [164], which is an evolutionary prototype selection algorithm that uses a memetic algorithm in order to perform a local search. In this case, an additional parameter *max_it* sets the maximum number of iterations performed by the search algorithm.

- OeSNN-PG, all controlled by parameters *min_size_cluster* and *error_tol*, which determine the minimum size of the cluster and the error tolerance before splitting a group, respectively:
  - Self-Generating Prototypes (SGP) [165], [166], which is a centroid-based prototype generation algorithm that uses a space splitting mechanism to generate prototypes in the center of every cluster in which data can be grouped;
  - Self-Generating Prototypes 2 (SGP2) [165], [166] is the second version of the SGP algorithm. It has a higher generalization power, including merge and pruning procedures; and
  - Adaptive Self-Generating Prototypes (ASGP) [165], [166], which has been specially designed to cope with imbalanced data sets.

We propose two different strategies for the resulting hybrid approaches, hereafter labeled as OeSNN-*DRT*, where

$$DRT \in \{\text{ENN}, \text{RENN}, \text{CNN}, \text{AllKNN}, \text{SSMA}, \text{SGP}, \text{SGP2}, \text{ASGP}\}.$$

In the first strategy data reduction is carried out every time the neurons repository is full (Algorithm 7), whereas in the second approach

a drift detector notifies when the data reduction process needs to be triggered (Algorithm 8). As it will be further explained, the first one (passive strategy) will be used for those experiments with synthetic data where the drift moment is known beforehand, whereas the second one (active strategy) will be adopted for the experiments with real data, where the drift moment is unknown.

One of the differences between the proposed OeSNN approach (Algorithm 6) and the approaches hybridized with DRTs schemes (Algorithms 7 and 8) is that the neurons repository size is limited and the merging process is hence unnecessary. The main goal of the DRTs is to summarize the underlying characteristics of the neurons repository over long periods of time, such that every newly included neuron has relevant information from two perspectives: 1) timeliness, as it replaces old neurons when the knowledge base in the neuron repository has no free space; and 2) predictive representativeness, because the new neuron will be fused with other neurons in the repository if it provides no further information on the prevailing concepts along the stream. This process can be regarded as a merging strategy of the whole neurons repository rather than a fusion between any two output neurons. Furthermore, the knowledge stored in the repository becomes more optimally assigned and hence, leads to a more suitable model design for online learning environments.

Algorithms 7 (passive strategy) and 8 (active strategy) evince that the operation of the OeSNN-PS and OeSNN-PG variants is similar to that of the OeSNN baseline in Algorithm 6. In essence, when the neurons repository is full DRTs are used to summarize the content of the neurons repository (line 18 in Algorithms 7 and 8). In the case of an active strategy, a drift detector is used (lines 16 and 17) to infer the moment at which DRTs should be applied: while no drift is detected and the neurons repository is not full, produced output neurons are always stored in the repository. When the repository saturates, the oldest output neuron is removed and the new one is stored instead. When a drift is detected, a DRT is applied to the neurons repository so as to make more room to store samples of the newly arriving concept.

The proposed OeSNN approach and those hybridized with DRTs schemes (OeSNN-PS and OeSNN-PG in both passive and active operation modes schematically depicted in Figure 4.6) have been specially devised for online learning purposes: on the one hand, windowing strategies are usually preferred for sudden drifts, while on the other hand, samples selection strategies are instead adopted to

handle gradual drifts and reoccurring contexts [11]. Both characteristics have been included in the proposed approaches by using a sliding window and neurons repository composed of prototypes.

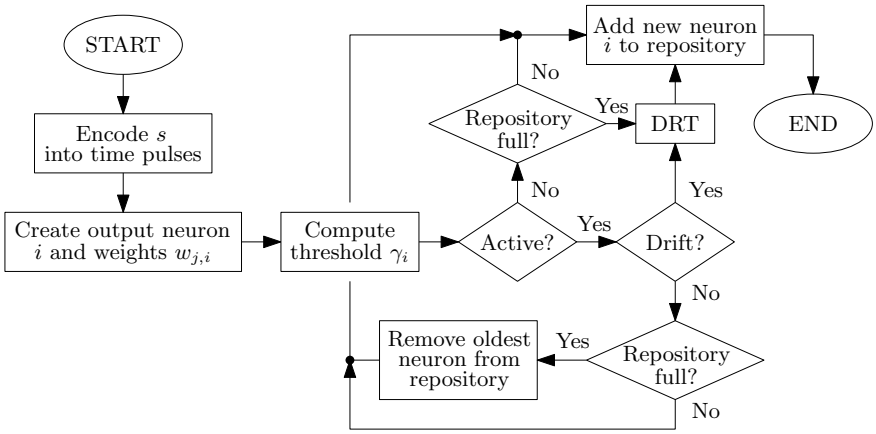


FIGURE 4.6: Scheme of the proposed OeSNN-DRT schemes with passive and active strategies.

Algorithm 7: OeSNN algorithm with DRTs: passive approach

Initialize neuron repository, $NR = \{\}$
Set eSNN parameter $mod = [0,1], C = [0,1], SIM = [0,1]$
Set eSNN encoding parameters β, T, N
Set sliding window size $\mathcal{W}$
Set *Neuron repository* of size NR_size
Set current neuron repository size $CNR\_size = 0$
for *every sample s belonging to the class c* **do**
    Update sliding window with sample s
    Calculate I_{max}, I_{min} for the $\mathcal{W}$ samples in the sliding window
    Calculate C_j and W_j over the sliding window for their encoding into firing time of multiple pre-synaptic neurons j
    Create a new output neuron i and the connection weights as $w_{ji} = mod^{order(j)}$
    Calculate $PSP_{max(i)} = \Sigma_j w_{ji} \cdot mod^{order(j)}$
    Get PSP threshold value $\gamma_i = PSP_{max(i)} \cdot C$
    if $CNR\_size < NR\_size$ **then**
        Add the weight vector and threshold of the new output neuron to NR
        $CNR\_size = CNR\_size + 1$
    else
        Apply a DRT over the neurons repository (PS or PG)
        Add the weight vector and threshold of the new output neuron to NR
        Update CNR_size
    end
end
Repeat above for all target classes

Algorithm 8: OeSNN algorithm with DRTs: active approach

Initialize *DriftDetector()*
Initialize neuron repository, $NR = \{\}$
Set eSNN parameter $mod = [0,1], C = [0,1], SIM = [0,1]$
Set eSNN encoding parameters β, T, N
Set sliding window size $\mathcal{W}$
Set *Neuron repository* of size NR_size
Set current neuron repository size $CNR\_size = 0$
Set $drift\_detection = False$
for *every sample s belonging to the class c* **do**
    Update sliding window with sample s
    Calculate I_{max}, I_{min} for the $\mathcal{W}$ samples in the sliding window
    Calculate C_j and W_j over the sliding window for their encoding into firing time of multiple pre-synaptic neurons j
    Create a new output neuron i and the connection weights as $w_{ji} = mod^{order(j)}$
    Calculate $PSP_{max(i)} = \Sigma_j w_{ji} \cdot mod^{order(j)}$
    Get PSP threshold value $\gamma_i = PSP_{max(i)} \cdot C$
    $drift\_detection = DriftDetector()$
    if $drift\_detection = True$ **then**
        Apply a DRT over the neurons repository (PS or PG)
        Add the weight vector and threshold of the new output neuron to NR
        Update CNR_size
        Set $drift\_detection = False$
    else
        if $CNR\_size < NR\_size$ **then**
            Add the weight vector and threshold of the new output neuron to NR
            $CNR\_size = CNR\_size + 1$
        else
            Remove the oldest weight vector and its threshold, and insert the new one into NR
        end
    end
end
Repeat above for all target classes

4.5 Computer Experiments

An extensive experimental benchmark has been designed to shed light on the performance of the proposed schemes over synthetic and real streaming data sets. Such experiments are divided into two main blocks:

- In the first set of experiments, the naive OeSNN approach (Algorithm 6) will be compared to *passive* OeSNN-PS and OeSNN-PG techniques (Algorithm 7) when they are applied over synthetic data sets, assuming that the drift moment is known beforehand.
- In the second set of experiments, the naive OeSNN will be compared to *active* OeSNN-PS and OeSNN-PG approaches over real data streams, where drifts are unknown and therefore motivates the use of a drift detector.

As a result of this experimental design, we will first analyze the performance of OeSNN-PS and OeSNN-PG approaches with synthetic data sets, and assess their benefits during the plasticity period (i.e. shortly after the drift). The main advantage of OeSNN-PS and OeSNN-PG is that while the neurons repository is not full, it can accept more neurons (the latest ones) inside. But when it is full, all the information is condensed in a reduced number of neurons (prototypes), letting more free space to accept more neurons inside until the neurons repository is full again. Therefore, we should expect OeSNN-PS and OeSNN-PG to react better (faster) to sudden drifts. Next, a set of experiments with real data sets and a drift detector are planned to confirm this benefit in a realistic setting. To numerically quantify the predictive performance of the proposed schemes during the stability and plasticity periods, the prequential accuracy will be measured at three different points in time: right before the drift occurs (BD), during the drifting period (D), and after the drift occurs (AD). The exact time ticks at which this score is computed will be set explicitly for every data set in the benchmark, which are described in the following section.

4.5.1 Data sets

As was argued in Section 3.5, working with real data sets, it is not possible to know exactly when a drift occurs, which type of drift

arises when a drift is detected, or even if there is any drift. Consequently, it is not possible to perform a detailed analysis of the behavior of different algorithms in the presence of concept drift by using only real-world data sets. In order to analyze the effect of DRTs for facing concept drift and to complement the analysis of our proposed approaches, we first use the renowned set of synthetic data sets described in [3].

Results for 4 different problems (see Section 3.4.1) will be considered as synthetic data. As for real drifting scenarios we resort to three different data sets. The first two data sets are well-known in the online learning community, since they have been used in several relevant studies of online approaches [8], [27], [167]. The third one was published in Kaggle[1], a platform for predictive modeling and analytics competitions and challenges. It is a brand new data set, recently used for the first time in a work on evolving data stream classification [168]. More details on these three data sets are next provided:

- The Australian New South Wales Electricity Market, already used in Section 3.5.
- The National Oceanic and Atmospheric Administration of the United States Department of Commerce[2] (USDC) has built a database (labeled as `NOAA`) with 18,154 daily weather measurements (50 years) from over 7,000 weather stations all around the world. Data samples include 8 features, such as temperature, dew point, sea level pressure, visibility, average wind speed, and other weather related predictors alike. These variables are used to infer whether each day was rainy or not.
- The Give Me Some Credit[3] data set (labeled as `GMSC`) is a credit scoring data set aimed at deciding whether a loan should be granted. This is a core decision for banks due to the risk of unexpected expenses and future lawsuits. The data set comprises supervised historical data of 150,000 borrowers described by 10 features.

[1] https://www.kaggle.com

[2] Available at: ftp.ncdc.noaa.gov/pub/data/gsod. Last access in March 20th, 2018.

[3] Available at: https://www.kaggle.com/c/GiveMeSomeCredit. Last access in March 20th, 2018.

4.5.2 Drift Detection

When dealing with real-world streaming data sets, drifts moments are unknown. As a consequence, a drift detector is required for those active approaches that need to know this information as soon as possible in order to trigger their adaptation mechanisms. This is the case of our proposed approaches: to this aim, they have been hybridized with the so-called Early Drift Detection Method (EDDM) [50]. Its simplicity and capacity to detect repeatedly occurring concept drifts even with very noisy data [50] has motivated the selection of EDDM for the experimental benchmark discussed in what follows. But there is the possibility of hybridizing with other drift detection methods in the literature.

4.5.3 Parameters Configuration

Parameters configuration for synthetic data sets `CIRCLE`, `LINE`, `SINEH` and `SINEV` data sets are presented in Table 4.1. Empirical experiments have been carried out to find the values.

	$\mathcal{W}$	T	β	N	*MOD*	C	*SIM*	k	*max_loop*	*min_size_cluster*	*error_tol*
OeSNN	100	20	1.5	10	0.85	0.75	0.15	-	-	-	-
OeSNN-TCNN	100	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-SSMA	100	20	1.5	10	0.85	0.75	-	3	50	-	-
OeSNN-ENN	100	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-RENN	100	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-AllKNN	100	20	1.5	10	0.85	0.75	-	5	-	-	-
OeSNN-CNN	100	20	1.5	10	0.85	0.75	-	1	-	-	-
OeSNN-SGP	100	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3
OeSNN-SGP2	100	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3
OeSNN-ASGP	100	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3

TABLE 4.1: Parameter values set for OeSNN when applied over `CIRCLE`, `LINE`, `SINEH` and `SINEV`.

In a real scenario, there is no a priori knowledge about the streaming data that the algorithm will predict, thus we cannot assume any configuration of parameters. Given this issue, an effective yet unrealistic workaround is to isolate a representative portion of the data set to make offline assumptions about the distribution of the data and to assign a suitable parametric configuration through a heuristic wrapper. We embrace this strategy to initialize the algorithm with a realistic configuration. Remarkably, the recent literature has widely acknowledged that the parametric optimization of predictive models for data streams while in operation still remains an open research problem [18], a paradigm that has been also noted in the research trends exposed in this thesis (see Section 2.7.1).

Regarding the ELEC2 data set, we have used the first 12 months (38%) of the data set to tune the parameters of the algorithms, whereas the remaining months have been used for prediction and performance assessment. A period of 48 time steps that corresponds to one day was assumed to study the behavior of our approaches during the drift (D). The sliding window size $\mathcal{W}$ was set to 96 samples. As for the NOAA data set we proceed similarly: the first 5 years (10%) of data were used for parameter tuning, whereas the following years were used for testing purposes. A period of 2 time steps that corresponds to two days was assumed to study the behavior of our approaches during the drift (D). The value of $\mathcal{W}$ was set to 25 samples. Finally, in the GMSC data set the first 20000 samples (16%) of the data set were used for model configuration, and the rest for performance assessment. The behavior of our approaches during the drift (D) was studied over 50 time steps. In this case, $\mathcal{W}$ was established to 25 samples.

	T	β	N	MOD	C	SIM	k	max_loop	$min\_size\_cluster$	$error\_tol$
OeSNN	20	1.5	10	0.85	0.75	0.15	-	-	-	-
OeSNN-TCNN	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-SSMA	20	1.5	10	0.85	0.75	-	1	50	-	-
OeSNN-ENN	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-RENN	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-AllKNN	20	1.5	10	0.85	0.75	-	3	-	-	-
OeSNN-CNN	20	1.5	10	0.85	0.75	-	2	-	-	-
OeSNN-SGP	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3
OeSNN-SGP2	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3
OeSNN-ASGP	20	1.5	10	0.85	0.75	-	-	-	0.2	0.3

TABLE 4.2: Parameter values utilized for the real data sets ELEC2, NOAA and GMSC.

Table 4.2 summarizes the parameter values for experiments with the real data sets. The warning level α and the drift level threshold β of the EDDM detector were set to 0.95 and 0.90, respectively. In order to inspect the impact of the DRTs on the neurons repository, three storage capacities $NR\_size \in \{50, 100, 150\}$ have been simulated for all data sets (either synthetic or real).

4.6 Results and Discussion

This section analyzes the behavior of the proposed OeSNN when hybridized with OeSNN-PS and OeSNN-PG schemes, in which PS and PG data reduction techniques are applied respectively to the neurons repository. The analysis focuses not only on the accuracy

of the approaches during the stability (BD and AD) and plasticity periods (D), but also on the data reduction percentage they achieve.

4.6.1 Impact of the Neurons Repository with Synthetic Data

To begin with, Table 4.3 shows the prequential accuracies of the naive OeSNN (i.e. the OeSNN without DRTs that has been detailed in Algorithm 6) when the neurons repository has a storage capacity of 50, 100, and 150, measured at points BD, D, and AD for all synthetic data sets. The table is complemented by Figure 4.7, which exemplifies the occupancy level of the neurons repository along the stream. Little variations of the occupancy correspond to those time instants at which the incoming sample is found to be similar to another one in the neurons repository, thus triggering the merging process. As evinced in this plot, this procedure occurs frequently depending on the *SIM* parameter. However, the main drawback is that this process only involves two neurons at every time, so there is no real contribution of this merging process to the optimized management of the neurons repository that stream processing clearly demands.

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	0.884/0.889/0.888	0.790/0.750/0.760	0.851/0.880/0.873	0.884/0.889/0.888	0.940/0.960/0.950	0.876/0.909/0.904
LINE	0.926/0.935/0.948	0.900/0.910/0.880	0.932/0.950/0.948	0.926/0.935/0.948	0.890/0.940/0.960	0.912/0.922/0.930
SINE	0.907/0.928/0.942	0.880/0.900/0.880	0.921/0.950/0.944	0.907/0.928/0.942	0.940/0.990/0.970	0.929/0.937/0.936
SINEH	0.792/0.811/0.827	0.760/0.770/0.670	0.753/0.795/0.794	0.792/0.811/0.827	0.840/0.840/0.830	0.736/0.781/0.797
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	0.884/0.889/0.888	0.810/0.730/0.740	0.831/0.845/0.834	0.884/0.889/0.888	0.880/0.890/0.910	0.791/0.834/0.823
LINE	0.930/0.940/0.949	0.850/0.690/0.650	0.912/0.901/0.893	0.930/0.940/0.949	0.890/0.940/0.950	0.879/0.882/0.866
SINE	0.927/0.947/0.956	0.820/0.680/0.610	0.917/0.918/0.880	0.927/0.947/0.956	0.890/0.910/0.920	0.878/0.884/0.865
SINEH	0.792/0.811/0.827	0.670/0.460/0.460	0.773/0.766/0.744	0.792/0.811/0.827	0.830/0.840/0.840	0.699/0.738/0.757

TABLE 4.3: Prequential accuracies of the proposed naive OeSNN model with neurons repository sizes 50, 100, and 150 for CIRCLE, LINE, SINEH and SINEV data sets.

We now turn the focus on the OeSNN incorporating DTRs in a passive strategy. Tables 4.4 and 4.5 summarize the results obtained for the proposed model hybridized with selective DRTs (OeSNN-PS): OeSNN-TCNN, OeSNN-SSMA and OeSNN-ENN (Table 4.4), and OeSNN-RENN, OeSNN-AllKNN and OeSNN-CNN (Table 4.5) when the neurons repository has a storage capacity of 50, 100, and 150 neurons, measured in the points BD, D, and AD over synthetic data sets. Figures 4.8 and 4.9 depict the evolution of the repository occupancy over the stream for all the aforementioned OeSNN schemes. Finally, the same set of results are shown in Table 4.6 and

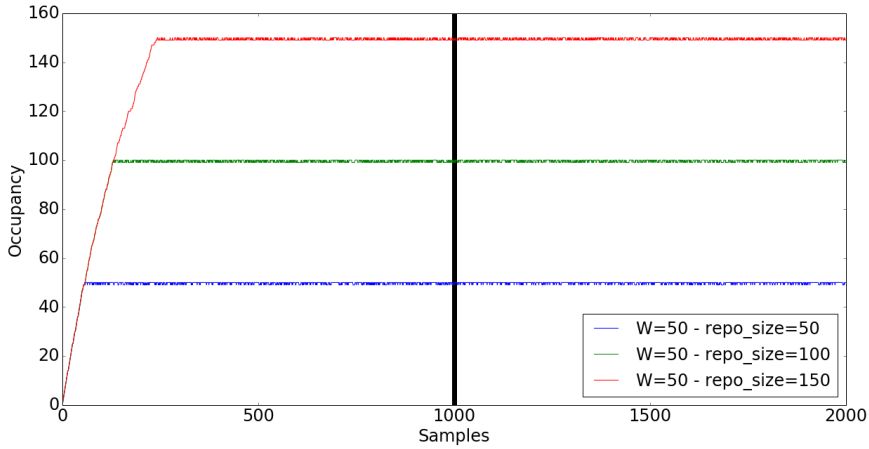
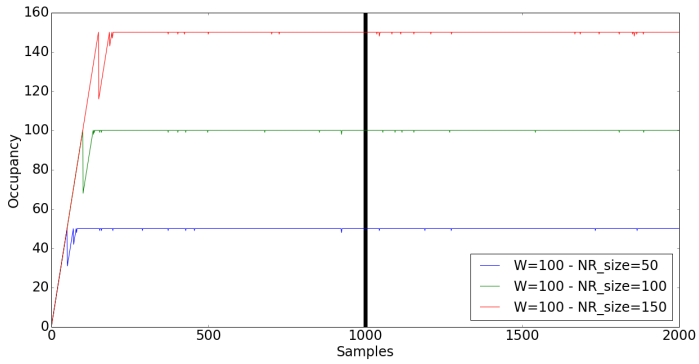


FIGURE 4.7: Evolution of the occupancy of the neurons repository for the proposed OeSNN in the CIRCLE data set under low severity and high speed conditions. The averaged occupancy is 98.08% (50 neurons), 96.56% (100 neurons), and 94.57% (150 neurons).

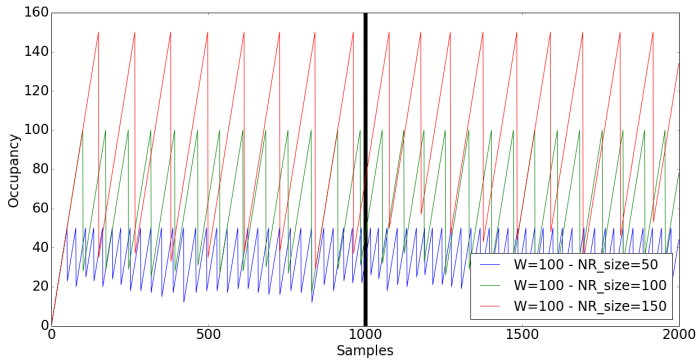
Figure 4.10 for OeSNNs based on generative data reduction techniques (OeSNN-PG).

4.6.2 Impact of the Neurons Repository with Real Data

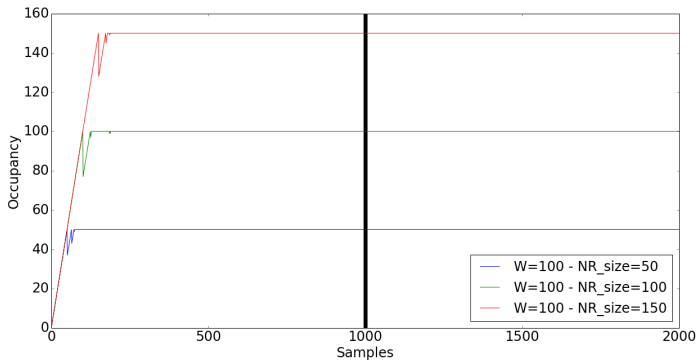
Once OeSNN-PS and OeSNN-PG approaches have been simulated with synthetic data sets, a second set of experiments has been performed with real data sets. The OeSNN-DRT hybridized with a drift detector (Algorithm 8) has been used in these simulations to be compared with the naive version of the proposed OeSNN because no a priori information about the drift moments is known. The drift detector serves as an active strategy to notify when it is necessary to trigger the DRT at hand. In this second experimental benchmark, averaged prequential accuracy scores for all cases (OeSNN, active OeSNN-PS and active OeSNN-PG) are jointly summarized in Table 4.7 for all real data sets, repository sizes (50, 100, and 150 neurons), and measured at points BD, D, and AD. Statistics during the same periods for the drift detector are also provided. The averaged prequential accuracies have been calculated over all BD, D, and AD periods that occur in all drift detections.



(a) OeSNN-AllKNN: 98.47%, 97.19%, 95.96%

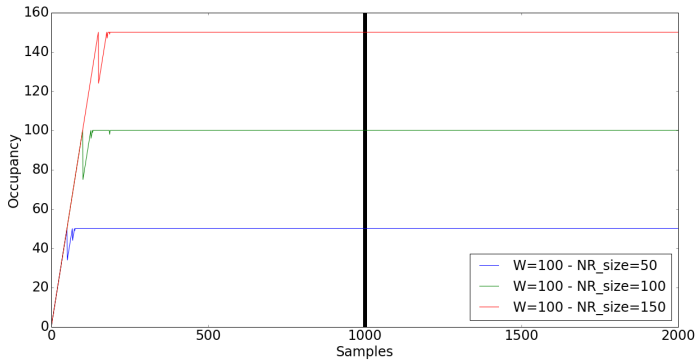


(b) OeSNN-CNN: 69.93%, 67.60%, 64.71%

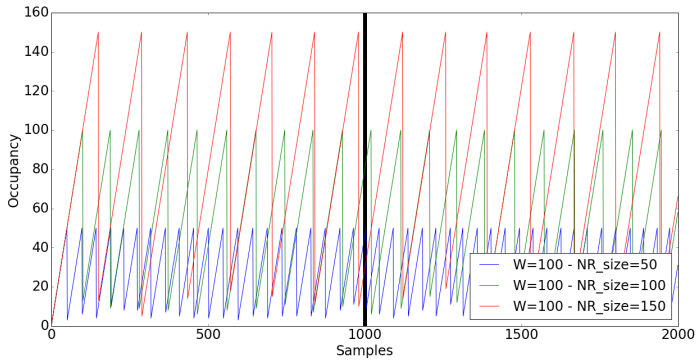


(c) OeSNN-ENN: 98.60%, 97.32%, 96.08%

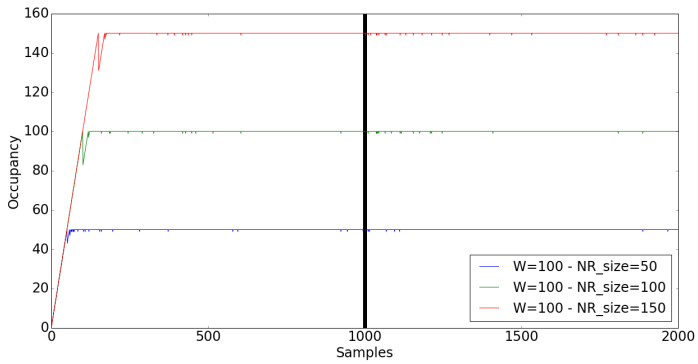
FIGURE 4.8: Occupancy of the neurons repository for the OeSNN-PS approaches (OeSNN-AllKNN, OeSNN-CNN, OeSNN-ENN) in the CIRCLE data set under high severity and speed conditions. Occupancy (in %) for repository sizes equal to 50, 100 and 150 neurons is displayed.



(a) OeSNN-RENN: 98.57%, 97.26%, 95.99%



(b) OeSNN-SSMA: 55.83%, 54.97%, 53.40%



(c) OeSNN-TCNN: 98.64%, 97.39%, 96.16%

FIGURE 4.9: Occupancy of the neurons repository for the OeSNN-PS approaches (OeSNN-RENN, OeSNN-SSMA, OeSNN-TCNN) in the `CIRCLE` data set under high severity and speed conditions. Occupancy (in %) for repository sizes equal to 50, 100 and 150 neurons is displayed.

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.848*/0.882/0.895	0.780/**0.770**/**0.894**	0.857/0.885/**0.915**	*0.848*/0.882/0.895	*0.870*/*0.940*/0.940	*0.845*/0.916/**0.922**
LINE	*0.911*/0.932/0.946	**0.920**/**0.930**/**0.900**	*0.910*/*0.927*/*0.925*	*0.911*/0.932/0.946	**0.920**/*0.910*/0.960	*0.888*/*0.910*/*0.913*
SINE	*0.892*/0.933/0.938	**0.900**/0.900/0.870	*0.888*/*0.915*/*0.901*	*0.892*/0.933/0.938	0.940/*0.940*/0.960	*0.893*/*0.910*/*0.921*
SINEH	*0.764*/0.802/0.836	*0.720*/*0.740*/**0.710**	0.751/0.801/*0.797*	*0.764*/0.802/0.836	*0.780*/**0.850**/0.850	0.746/**0.811**/**0.810**
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.848*/0.882/0.895	**0.840**/**0.750**/0.730	*0.777*/*0.780*/*0.769*	*0.848*/0.882/0.895	0.890/**0.920**/0.900	*0.764*/*0.772*/*0.752*
LINE	0.932/0.937/0.951	*0.650*/*0.620*/*0.610*	*0.755*/*0.709*/*0.697*	0.932/0.937/0.951	**0.910**/*0.910*/0.950	*0.780*/*0.759*/*0.748*
SINE	0.922/0.943/*0.942*	*0.680*/*0.630*/**0.630**	*0.710*/*0.707*/*0.701*	0.922/0.943/*0.942*	**0.910**/0.920/0.920	*0.779*/*0.763*/*0.748*
SINEH	*0.764*/0.802/0.836	*0.560*/0.460/*0.440*	*0.664*/*0.658*/*0.656*	*0.764*/0.802/0.836	*0.760*/0.840/0.840	*0.661*/*0.653*/*0.617*

(a) OeSNN-TCNN

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.832*/*0.861*/**0.888**	*0.770*/**0.780**/**0.850**	*0.803*/*0.836*/**0.888**	*0.812*/*0.864*/**0.882**	*0.890*/*0.890*/**0.960**	*0.824*/*0.879*/**0.895**
LINE	*0.902*/*0.922*/**0.944**	**0.890**/**0.900**/**0.890**	*0.889*/*0.932*/*0.932*	*0.898*/*0.918*/*0.932*	*0.860*/*0.870*/*0.910*	*0.866*/*0.900*/*0.912*
SINE	*0.886*/*0.904*/*0.917*	*0.840*/*0.860*/**0.870**	*0.892*/*0.922*/*0.924*	*0.870*/*0.904*/*0.916*	*0.880*/*0.950*/*0.940*	*0.871*/*0.909*/**0.928**
SINEH	*0.768*/*0.778*/*0.795*	**0.770**/**0.800**/**0.710**	*0.727*/**0.785**/**0.785**	*0.753*/*0.770*/*0.789*	*0.760*/*0.810*/**0.840**	**0.727**/*0.747*/*0.759*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.817*/*0.870*/**0.898**	*0.750*/**0.770**/**0.740**	*0.785*/*0.819*/**0.833**	*0.828*/*0.866*/*0.872*	**0.870**/**0.880**/*0.890*	*0.740*/*0.772*/**0.813**
LINE	*0.890*/*0.919*/*0.927*	**0.840**/**0.800**/**0.870**	*0.874*/**0.915/0.916**	*0.886*/*0.912*/*0.929*	*0.840*/*0.880*/*0.910*	*0.828*/**0.876**/**0.880**
SINE	*0.875*/*0.914*/*0.930*	**0.830**/**0.780**/**0.810**	*0.872*/**0.910**/**0.912**	*0.873*/*0.918*/*0.931*	*0.870*/*0.890*/*0.900*	*0.858*/*0.861*/**0.891**
SINEH	*0.731*/*0.766*/*0.784*	**0.750**/**0.710**/**0.660**	*0.724*/**0.776**/**0.773**	*0.743*/*0.766*/*0.784*	*0.700*/**0.840**/**0.830**	*0.670*/*0.709*/*0.716*

(b) OeSNN-SSMA

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.727*/*0.839*/*0.848*	*0.700*/**0.780**/**0.780**	*0.710*/*0.850*/0.871	*0.727*/*0.839*/*0.848*	*0.700*/*0.850*/*0.910*	*0.716*/*0.870*/*0.889*
LINE	*0.890*/**0.943**/0.938	0.900/0.900/**0.910**	*0.903*/*0.891*/*0.904*	*0.890*/**0.943**/0.938	**0.910**/0.940/*0.900*	*0.891*/*0.901*/*0.901*
SINE	*0.859*/*0.908*/0.932	*0.840*/*0.880*/**0.910**	*0.837*/*0.881*/*0.907*	*0.859*/*0.908*/0.932	*0.900*/*0.900*/*0.950*	*0.850*/*0.882*/*0.914*
SINEH	*0.724*/*0.784*/*0.807*	*0.650*/*0.740*/**0.720**	*0.691*/*0.749*/*0.735*	*0.724*/*0.784*/*0.807*	*0.750*/*0.790*/*0.810*	*0.723*/0.775/*0.751*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.727*/*0.839*/*0.848*	*0.630*/0.730/0.730	*0.677*/*0.717*/*0.713*	*0.727*/*0.839*/*0.848*	*0.740*/*0.830*/*0.880*	*0.672*/*0.711*/*0.717*
LINE	*0.884*/0.930/0.941	*0.660*/*0.640*/*0.630*	*0.656*/*0.652*/*0.659*	*0.884*/0.930/0.941	*0.850*/*0.840*/*0.920*	*0.698*/*0.696*/*0.718*
SINE	*0.885*/*0.907*/*0.922*	*0.630*/*0.640*/**0.640**	*0.672*/*0.666*/*0.658*	*0.885*/*0.907*/*0.922*	0.880/*0.890*/*0.900*	*0.736*/*0.739*/*0.736*
SINEH	*0.724*/*0.784*/*0.807*	*0.400*/*0.350*/*0.350*	*0.400*/*0.410*/*0.380*	*0.724*/*0.784*/*0.807*	*0.760*/*0.770*/*0.790*	*0.475*/*0.493*/*0.474*

(c) OeSNN-ENN

TABLE 4.4: Prequential accuracies obtained by (a) OeSNN-TCNN, (b) OeSNN-SSMA and (c) OeSNN-ENN working with repository sizes of 50, 100, and 150 neurons over the CIRCLE, LINE, SINEH and SINEV data sets. Prequential accuracies in bold denote an improvement greater than 0.01 in comparison with the traditional OeSNN for the same scenario setup. On the contrary, prequential accuracies in italics are declared to be worse than the traditional OeSNN if they are at least 0.01 below the accuracy scored by the latter. Prequential accuracies in regular text stand for performance gaps less than 0.01 in absolute value.

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.723/0.825/0.833*	*0.690/**0.770**/**0.810***	*0.718/0.842/0.876*	*0.723/0.825/0.833*	*0.720/0.840/0.880*	*0.727/0.851/0.881*
LINE	*0.869*/0.943/0.941	*0.780*/0.900/***0.910***	*0.835/0.891/0.916*	*0.869*/0.943/0.941	***0.910***/0.940/*0.910*	*0.847/0.901*/0.926
SINE	*0.859/0.908/0.924*	*0.840/0.880*/0.890	*0.837/0.881/0.894*	*0.859/0.908/0.924*	*0.900/0.900/0.930*	*0.850/0.882/0.907*
SINEH	*0.724/0.783/0.806*	*0.650/0.740/**0.720***	*0.691/0.749/0.736*	*0.724/0.783/0.806*	*0.750/0.790/0.810*	*0.723*/0.772/*0.752*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.723/0.825/0.833*	*0.640/**0.750***/0.750	*0.681/0.734/0.736*	*0.723/0.825/0.833*	*0.740/0.830/0.860*	*0.669/0.724/0.730*
LINE	*0.884*/0.930/0.940	*0.660/0.640/0.630*	*0.656/0.652/0.659*	*0.884*/0.930/0.940	*0.850/0.840/0.920*	*0.698/0.696/0.718*
SINE	*0.885/0.907/0.919*	*0.630/0.640/**0.640***	*0.672/0.666/0.659*	*0.885/0.907/0.919*	0.880/*0.890/0.900*	*0.736/0.739/0.735*
SINEH	*0.724/0.783/0.806*	*0.400/0.350/0.350*	*0.400/0.407/0.380*	*0.724/0.783/0.806*	*0.760/0.770/0.790*	*0.475/0.491/0.474*

(a) OeSNN-RENN

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.794/0.828/0.852*	*0.770/**0.800**/**0.820***	*0.747/0.856/0.876*	*0.794/0.828/0.852*	*0.770/0.830/0.880*	*0.803/0.850/0.886*
LINE	*0.870/0.894/0.929*	*0.880/0.860/0.860*	*0.882/0.891/0.911*	*0.870/0.894/0.929*	0.880/*0.890/0.900*	*0.855/0.874/0.892*
SINE	*0.879/0.906/0.918*	*0.780/0.840*/0.880	*0.839/0.873/0.904*	*0.879/0.906/0.918*	0.930/*0.900/0.950*	*0.847/0.890/0.915*
SINEH	*0.719/0.793/0.790*	*0.660*/0.780/0.680	*0.638*/0.785/*0.728*	*0.719/0.793/0.790*	*0.730/0.800/0.810*	*0.657/**0.801***/*0.757*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.794/0.828/0.852*	*0.730/**0.800**/**0.780***	*0.649/0.781/0.787*	*0.794/0.828/0.852*	*0.800/0.820/0.860*	*0.655/0.743/0.763*
LINE	0.922/0.930/0.940	*0.700/0.640*/0.650	*0.809/0.782/0.779*	0.922/0.930/0.940	*0.870/0.900/0.910*	*0.845/0.811/0.805*
SINE	*0.849/0.906/0.918*	*0.710/0.660/**0.660***	*0.783/0.800/0.782*	*0.849/0.906/0.918*	*0.810/0.880/0.890*	*0.825/0.808/0.816*
SINEH	*0.719/0.793/0.790*	*0.460/0.440/**0.500***	*0.488/0.600/0.613*	*0.719/0.793/0.790*	*0.720/0.820/0.810*	*0.565/0.617/0.615*

(b) OeSNN-AllKNN

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.795/0.841/0.840*	*0.740/**0.770**/**0.750***	*0.807/0.823/0.810*	*0.837/0.846/0.862*	***0.930***/*0.890/0.870*	*0.825/0.840/0.826*
LINE	*0.903/0.924/**0.938***	*0.860/0.860/**0.910***	*0.913/0.909/**0.937***	***0.916**/**0.929**/**0.945***	*0.860/0.880/0.910*	*0.851/0.871/0.879*
SINE	*0.889/0.916/0.928*	*0.840/0.860/0.860*	*0.902/0.918/0.908*	*0.870/0.902/0.927*	*0.900/0.960/**0.960***	*0.887/0.916/0.907*
SINEH	*0.734/0.772/0.804*	***0.770***/*0.680/**0.720***	***0.744***/*0.750/0.762*	*0.709/0.756/0.782*	*0.780/0.730/**0.830***	*0.712/0.727/0.740*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.847/0.841/0.848*	*0.730/0.670/0.680*	*0.776/0.758/0.760*	*0.861/0.846/0.849*	*0.850/0.790/0.780*	*0.759/0.745/0.731*
LINE	*0.891/0.911/0.927*	*0.750/**0.730**/**0.700***	*0.863/0.829/0.818*	*0.899/0.922/0.925*	***0.910***/*0.920/0.850*	*0.795/0.801/0.804*
SINE	*0.856/0.913/0.931*	*0.750/**0.740**/**0.700***	*0.841/0.853/0.852*	*0.884/0.908/0.923*	*0.840/**0.900**/**0.920***	*0.799/0.762/0.793*
SINEH	*0.726/0.773/0.809*	*0.550/**0.540**/**0.560***	*0.676/0.693/0.688*	*0.733/0.787/0.793*	*0.690/0.780/0.820*	*0.651/0.662/0.631*

(c) OeSNN-CNN

TABLE 4.5: Prequential accuracies obtained by (a) OeSNN-RENN, (b) OeSNN-AllKNN and (c) OeSNN-CNN working with repository sizes of 50, 100, and 150 neurons over the CIRCLE, LINE, SINEH and SINEV data sets. The same notational criteria hold in terms of statistical significance between these schemes and the naive OeSNN scheme.

4.6.3 Comparison to Other Methods

In Table 4.8 a comparison with some of the most recent and well-known online learning methods in the presence of concept drift is presented, namely, Hoeffding Trees or also known as Very Fast Decision Trees (HTs, [58]), Random Forests (RFs, [169]), and Hoeffding

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.860/0.876/0.859*	*0.750/***0.810/0.760**	*0.823/0.855/0.859*	*0.860/0.876/***0.882**	*0.860/0.920/***0.950**	*0.843/0.878/0.888*
LINE	**0.918/0.928**/*0.932*	**0.910/0.940/0.920**	**0.927/0.943/0.951**	**0.918/0.928**/*0.932*	**0.880**/*0.900/0.910*	*0.899/***0.916/0.923**
SINE	**0.911/0.922**/*0.918*	**0.910/0.890**/*0.850*	**0.923**/*0.930/***0.934**	**0.911/0.922**/*0.918*	**0.950**/*0.960/0.950*	*0.917/0.923/***0.933**
SINEH	*0.766/0.780/0.785*	**0.760/0.770/0.760**	**0.754**/*0.766/***0.787**	*0.766/0.780/0.785*	**0.830**/*0.800/***0.830**	**0.742**/*0.757/0.762*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.860/0.876/***0.882**	*0.770/***0.750/0.740**	*0.800/0.812/0.820*	*0.860/0.876/***0.882**	*0.850/0.870/0.860*	*0.754/0.797/0.799*
LINE	**0.925**/*0.924/0.929*	*0.820/***0.850/0.770**	*0.897/***0.912/0.904**	**0.925**/*0.924/0.929*	*0.870/0.880/0.870*	*0.858/0.859/***0.870**
SINE	*0.900/0.916/0.928*	**0.880/0.810/0.760**	**0.912/0.911/0.912**	*0.900/0.916/0.928*	**0.890/0.900**/*0.890*	**0.873/0.878/0.883**
SINEH	*0.766/0.780/0.785*	**0.690/0.710/0.690**	*0.741/***0.960/0.767**	*0.766/0.780/0.785*	**0.850**/*0.770/***0.830**	**0.707**/*0.712/0.734*

(a) OeSNN-SGP, OeSNN-SGP2

	Low drift severity, high drift speed			Low drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.860/0.876/***0.882**	*0.750/***0.810/0.760**	*0.822/0.855/0.859*	*0.860/0.876/***0.882**	*0.870/0.920/***0.950**	*0.834/0.878/0.888*
LINE	**0.918/0.928**/*0.932*	**0.910/0.940/0.920**	**0.927/0.943/0.951**	**0.918/0.928**/*0.932*	**0.880**/*0.900/0.910*	*0.889/***0.916/0.923**
SINE	**0.911/0.922**/*0.918*	**0.910/0.890**/*0.850*	**0.923**/*0.930/***0.934**	**0.911/0.922**/*0.918*	**0.950**/*0.960/0.950*	*0.917/0.923/***0.933**
SINEH	*0.766/0.780/0.785*	**0.760/0.770/0.760**	**0.754**/*0.766/***0.787**	*0.766/0.780/0.785*	**0.830**/*0.800/***0.830**	**0.742**/*0.757/0.762*
	High drift severity, high drift speed			High drift severity, low drift speed		
	BD	D	AD	BD	D	AD
CIRCLE	*0.860/0.876/***0.882**	*0.770/***0.750/0.740**	*0.807/0.821/0.823*	*0.860/0.876/***0.882**	*0.860/0.870/0.860*	*0.752/0.780/0.798*
LINE	**0.925**/*0.924/0.929*	*0.820/***0.850/0.770**	*0.897/***0.912/0.904**	**0.925**/*0.924/0.929*	*0.870/0.880/0.870*	*0.858/0.859/***0.870**
SINE	*0.900/0.916/0.928*	**0.880/0.810/0.760**	**0.912/0.911/0.912**	*0.900/0.916/0.928*	**0.890/0.900**/*0.890*	**0.873/0.878/0.883**
SINEH	*0.766/0.780/0.785*	**0.690/0.710/0.690**	*0.741/***0.760/0.767**	*0.766/0.780/0.785*	**0.850**/*0.770/***0.830**	**0.707**/*0.712/0.734*

(b) OeSNN-ASGP

TABLE 4.6: Prequential accuracies of the (a) OeSNN-SGP/OeSNN-SGP2 and (b) OeSNN-ASGP approach working with the neurons repository sizes of 50, 100, and 150 respectively for CIRCLE, LINE, SINEH and SINEV data sets. Where prequential accuracies are in bold, there is a notable improvement in comparison with the traditional OeSNN, whereas prequential accuracies in italics are worse than the traditional OeSNN. When prequential accuracies are in regular text means that there is no significant difference.

Naive Bayes Tree ensembles (HNBTs, [168]). Active implementations of some of the above detectors are also included in this chapter, encompassing drift detectors such as DDM [29], EDDM [50], ADWIN [27] or EDIST2 [113]. For the sake of fairness and comparability the table only reports experimental results of schemes published in the literature by other authors. Unfortunately, to the best of our knowledge no prior work for NOAA data set has been carried out under similar conditions and evaluation criteria, hence this data set has not been considered in this last experimental phase.

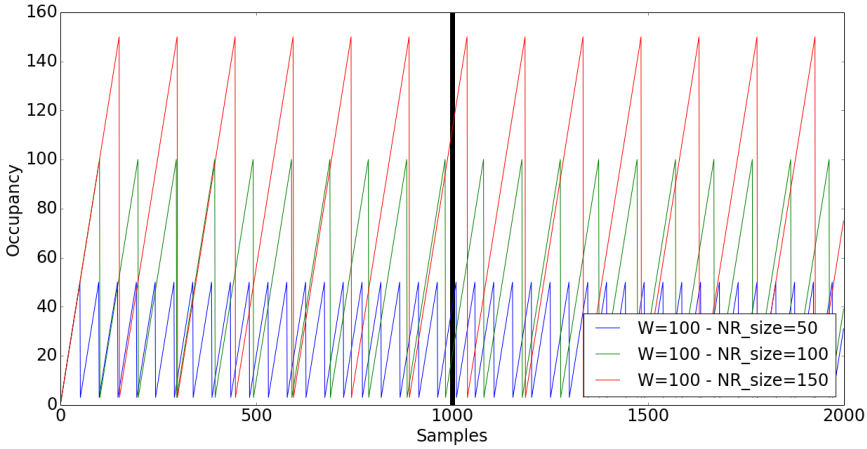


FIGURE 4.10: Averaged neurons repository occupancies (52.64%, 50.85%, 50.01%) of the all OeSNN-PG approaches for the sizes 50, 100, and 150 respectively with the CircleG data set under high severity and high speed conditions.

A first look at the results in this table reveals that the performance of several OeSNN-DRT approaches occurs to be very competitive with respect to the state of the art, above all after considering that our OeSNN-DRT approaches are based on a single model and the rest are based on ensemble models composed of 10 [170] or 100 [168] base learners. Nevertheless, the next section will elaborate on this comparison in depth.

4.6.4 Discussion

A first look in Figures 4.8, 4.8, and 4.10 reveals that DRTs are able to retain the knowledge with a reduced number of output neurons, but not all DRTs achieve a good classification performance and a relevant data reduction percentage at the same time. This is a key point in online learning scenarios where the storage capacity is limited.

We start the discussion by analyzing the results of the synthetic data, which are presented in Tables 4.3 to 4.6. The obtained scores for OeSNN-PS show that in general, OeSNN-TCNN, OeSNN-ENN, OeSNN-RENN and OeSNN-AllKNN render a degraded performance when compared to the naive OeSNN approach. Some exceptions deserve further attention at this point: the OeSNN-TCNN approach has a general better classification performance in the plasticity period D, mostly in those data sets with low severity and high speed.

		BD	D	AD	# drifts
OeSNN approach					
OeSNN	ELEC2	0.817/0.808/0.806	0.802/0.793/0.784	0.810/0.802/0.796	n.a.
	NOAA	0.700/0.675/0.685	0.692/0.640/0.674	0.702/0.630/0.673	n.a.
	GMSC	0.907/0.919/0.908	0.901/0.916/0.898	0.912/0.923/0.905	n.a.
Active OeSNN-PS approaches					
OeSNN-TCNN	ELEC2	0.827/0.806/0.801	**0.822**/0.792/*0.763*	**0.834**/0.802/*0.776*	13/117/23
	NOAA	0.697/**0.689**/0.692	*0.604*/**0.689**/*0.623*	*0.669*/**0.686**/*0.646*	50/61/12
	GMSC	**0.907/0.911/0.907**	*0.888*/**0.908/0.906**	*0.896/0.912*/**0.906**	44/33/50
OeSNN-SSMA	ELEC2	**0.819/0.807/0.807**	**0.792/0.804/0.800**	**0.802/0.810/0.805**	5/106/66
	NOAA	*0.684*/**0.706/0.675**	*0.647/0.567*/**0.715**	*0.652/0.540*/**0.705**	85/4/22
	GMSC	**0.902/0.917/0.908**	**0.891/0.932/0.896**	*0.889*/**0.921/0.907**	45/7/19
OeSNN-ENN	ELEC2	0.817/0.804/0.800	0.800/0.784/0.774	0.805/0.801/0.792	106/111/99
	NOAA	0.697/**0.699**/0.695	*0.679*/**0.710**/*0.646*	0.705/**0.726/0.687**	56/17/57
	GMSC	**0.908/0.913/0.915**	**0.907/0.919/0.918**	**0.913/0.923/0.923**	51/49/47
OeSNN-RENN	ELEC2	0.811/0.804/0.804	*0.740*/0.784/0.788	*0.768*/0.801/0.800	5/111/100
	NOAA	0.693/0.684/0.681	*0.659*/**0.664**/*0.583*	*0.639*/**0.675**/*0.621*	56/71/15
	GMSC	**0.908/0.913/0.915**	**0.907/0.923/0.912**	**0.913/0.924/0.921**	51/49/46
OeSNN-AllKNN	ELEC2	0.815/0.811/0.802	0.792/0.784/0.785	0.808/0.801/0.803	106/100/126
	NOAA	**0.719/0.696/0.728**	**0.783/0.674/0.669**	**0.742/0.704/0.716**	12/19/8
	GMSC	**0.908/0.912/0.914**	**0.911/0.921/0.925**	**0.912/0.919/0.925**	51/46/45
OeSNN-CNN	ELEC2	**0.816/0.806/0.800**	**0.798/0.791/0.787**	**0.806/0.814/0.802**	60/26/114
	NOAA	*0.684*/**0.763/0.716**	**0.702/0.919**/*0.571*	*0.650*/**0.951**/*0.642*	55/1/3
	GMSC	**0.915/0.911**/*0.865*	*0.846*/**0.915**/*0.841*	*0.845/0.886/0.820*	5/6/21
Active OeSNN-PG approaches					
OeSNN-SGP	ELEC2	**0.817/0.813/0.813**	**0.815/0.807/0.809**	**0.814/0.810/0.811**	112/113/104
	NOAA	**0.695/0.674/0.683**	*0.657*/**0.739/0.720**	**0.698/0.623/0.698**	51/21/28
	GMSC	**0.911/0.920/0.908**	**0.914/0.931/0.911**	**0.913/0.935/0.912**	45/6/30
OeSNN-SGP2	ELEC2	**0.819/0.811/0.810**	**0.815/0.805/0.802**	**0.819/0.810/0.812**	113/112/98
	NOAA	**0.695/0.682/0.688**	*0.657*/**0.662/0.754**	**0.698/0.680/0.741**	51/59/39
	GMSC	**0.911/0.920/0.908**	**0.914/0.931/0.911**	**0.916/0.935/0.912**	45/6/30
OeSNN-ASGP	ELEC2	**0.816/0.812/0.805**	**0.811/0.806/0.797**	**0.815/0.809/0.802**	117/117/52
	NOAA	**0.697/0.678/0.687**	**0.713/0.638/0.693**	**0.696/0.658/0.686**	48/38/69
	GMSC	**0.912**/*0.904/0.896*	*0.875/0.893/0.873*	*0.882/0.880/0.880*	18/13/7

TABLE 4.7: Averaged prequential accuracies and number of detected drifts of the OeSNN-DRT approaches working with the neurons repository sizes 50, 100, and 150 for the real data sets. Where prequential accuracies are in bold, there is a notable improvement in comparison with the traditional OeSNN, whereas prequential accuracies in italics are worse than the traditional OeSNN. When prequential accuracies are in regular text means that there is no significant difference.

Similarly, OeSNN-ENN and OeSNN-RENN yield a better performance in the plasticity period D in data sets with low severity and high speed when the size of the repository is large (150 neurons). The OeSNN-AllKNN approach performs best in the plasticity period D over data sets with high severity and high speed, again for large repository sizes. This interesting result advocates for one of the postulated hypothesis: the larger the neuron repository is, the more important an optimized management of its contents is in an online learning setup; this fact becomes even more noticeable when

TECHNIQUES	TYPE	ELEC2	GMSC
OeSNN-DRT approaches			
OeSNN-TCNN	Single	0.822/0.792/0.763	0.888/0.908/0.906
OeSNN-SSMA	Single	0.792/0.804/0.800	0.891/0.932/0.896
OeSNN-ENN	Single	0.800/0.784/0.774	0.907/0.919/0.918
OeSNN-RENN	Single	0.740/0.784/0.788	0.907/0.923/0.912
OeSNN-AllKNN	Single	0.792/0.784/0.785	0.911/0.921/0.925
OeSNN-CNN	Single	0.798/0.791/0.787	0.846/0.915/0.841
OeSNN-SGP	Single	0.815/0.807/0.809	0.914/0.931/0.911
OeSNN-SGP2	Single	0.815/0.805/0.802	0.914/0.931/0.911
OeSNN-ASGP	Single	0.811/0.806/0.797	0.875/0.893/0.873
[170]			
HTs ensemble + EDIST2	Ensemble (10)	0.848	-
[168]			
Adaptive RFs	Ensemble (100)	0.885	0.935
Online Bagging (HNBTs)	Ensemble (100)	0.825	0.935
Online Accuracy Updated Ensemble (HNBTs)	Ensemble (100)	0.863	0.935
Online Boosting (HNBTs)	Ensemble (100)	0.901	0.926
Online Smooth-boost (HNBTs)	Ensemble (100)	0.875	0.925
Leveraging Bagging (HNBTs)	Ensemble (100)	0.885	0.935

TABLE 4.8: Comparison during the drifting phase of the average prequential accuracies of OeSNN-DRTs and some of the most relevant ensemble-based techniques in the literature. The same evaluation criteria was used in all works: a *test-then-train* scheme and average of prequential accuracies during the drifting phase as the performance metric.

the drift imprints deep changes on the stream data (high severity, high speed drifts), as the repository needs to be refreshed quickly so as to grasp and learn the newly evolving concept.

On the contrary, OeSNN-SSMA and OeSNN-CNN (those with higher data reduction percentages) offer further performance improvements in other periods over the simulated streams. OeSNN-SSMA has in general better classification performance in 1) the plasticity period D in data sets characterized by fast drifts, and 2) in the stable period AD for data sets with high severity and high speed. Besides, for stable periods BD and AD in many other cases, OeSNN-SSMA outperforms the proposed OeSNN, remarkably when the size of the repository is the largest one (150). On the other hand, the OeSNN-CNN approach produces in general better accuracy scores in the plasticity period D over data sets with high speed, and occasionally outperforms the naive OeSNN in other specific conditions. But it is the OeSNN-SMMA technique which provides better prequential accuracies and a more efficient use of the neurons repository capacity (more data reduction percentage) at the same time. Unfortunately, this comes along with a computation penalty, since the SMMA encompasses a heuristic search demanding for memory and processing resources that could eventually clash with stringent

computational constraints of the online problem in question.

Regarding OeSNN-PG approaches whose results are compiled in Table 4.6, OeSNN-SGP and OeSNN-SGP2 approaches rendered the same results and show a general better classification performance than the OeSNN approach in the plasticity period D for high speed data sets. They were found to be also competitive in low speed data sets, even in stable periods. The OeSNN-ASGP approach has similar results than OeSNN-SGP and OeSNN-SGP2, but it performs better in the CIRCLE data set for low severity and high speed drifts in the stable period BD when the size of the neurons repository is 150. It also shows a general better classification performance than the OeSNN approach in the plasticity period D for high speed data sets, and it is also a competitive technique in the low speed data sets, even in stable periods.

Once analyzed the impact of applying DRTs to OeSNN, we proceed by analyzing the set of experiments with real data sets leveraging the use of a drift detector to trigger the application of the data reduction technique. Regarding OeSNN-PS approaches, in Table 4.7 one can observe that OeSNN-TCNN, OeSNN-ENN, and OeSNN-RENN perform competitively in the stability and plasticity periods for several data sets. However, OeSNN-SSMA, OeSNN-AllKNN and OeSNN-CNN are the approaches rendering the best overall accuracy scores, with OeSNN-AllKNN dominating all OeSNN-PS techniques specially in the NOAA and GMSC data sets. As for OeSNN-PG approaches, Table 4.7 elucidates that the three approaches under this category perform better than the naive OeSNN in a wider spectrum of data sets and regions than their OeSNN-PS counterparts. Nonetheless, OeSNN-SGP and OeSNN-SGP2 are the techniques producing better results in all real data sets.

All in all, from the above experiments insightful conclusions can be drawn: OeSNN-SMMA, OeSNN-CNN, and all OeSNN-PG approaches (OeSNN-SGP, OeSNN-SGP2, and OeSNN-ASGP) achieve very high data reduction ratios and a competitive classification performance in comparison with the naive OeSNN approach in plasticity periods (D). They allow for more space for newly produced output neurons to enter the repository, thus favoring a quicker adaptation to the drift. Besides, these approaches decrease the processing time when the newly output neuron is compared to the rest of the neurons in the repository in the training phase: the less neurons in the repository, the less the number of pairwise comparisons will be needed to find the most similar neuron (*SIM* parameter for the merging process), and ultimately the less processing time and less storing

space will be required. Interestingly, OeSNN-PG approaches have revealed themselves as the most suitable models to be applied right after a drift occurs.

Finally, the results in Section 4.6.3 certify that OeSNN-DRT schemes perform very competitively in comparison with other methods from the state of the art, getting comparable accuracy scores to avant-garde schemes over the ELEC2 and GMSC data sets. At this point the reader should not overlook the fact that most online learning methods are based on ensemble classifiers rather than single models. Ensemble classifier models are widely acknowledged to be more accurate due to their robustness to error variance. Furthermore, they are more flexible to assimilate new available data into their learning algorithm, and they provide more straightforward mechanisms to forget irrelevant knowledge once a drift has been detected (e.g. by simply discarding the oldest classifier from the ensemble). In contrast, single model approaches generally trade lower accuracy results for a reduced computational cost, reason for which they are often regarded as an attractive solution for massive data streams [15]. Bearing the previous observations and considering key performance factors such as the size of the window or the number of base learners utilized in the compared ensembles (10 in the case of HTs ensemble + EDIST2 [170] or 100 in the case of ARFs and HNBTs approaches in [168]), we can conclude that OeSNN-DRTs are along with the state of the art related to online learning and concept drift in terms of accuracy and computational efficiency (since, as already argued in the introduction, single classifiers are considered more suitable for online scenarios with stringent timing constraints that jeopardize the adoption of ensemble-based approaches). This statement stimulates further research around different extensions of the proposed family of online classifiers, as will be next outlined.

4.7 Summary

Nowadays huge volumes of data affected by non-stationary phenomena are produced in the form of fast streams. The resulting lack of stationarity in the distribution of the produced data calls for efficient and scalable algorithms for online analysis capable of adapting such changes. The online learning field has lately turned its focus on this challenging scenario, by designing incremental learning algorithms that avoid becoming obsolete after a concept drift occurs. Despite the noted activity in the literature, a need for new efficient

and scalable algorithms that adapt to the drift still prevails as a research topic deserving further efforts. Surprisingly, eSNN, one of the major exponents of the third generation of artificial neural networks, have not been thoroughly studied as an online learning approach, even though they are naturally suited to easily and quickly adapt to changing environments. This chapter covers this research gap by adapting eSNN to meet the processing requirements that online learning scenarios impose. In particular, this chapter focuses on limiting the size of the neuron repository and making the most of this limited size by resorting to DRTs. Experiments with synthetic and real data have been discussed, leading to the empirically validated assertion that, by virtue of a tailored exploitation of the neuron repository, eSNNs adapt better to drifts, obtaining higher accuracy scores than naive versions of eSNN for online learning environments.

Chapter 5

Challenge 3: Drift Detection based on eSNN

5.1 Introduction

Several model approaches have been presented in the literature to deal with concept drift [26], [36]. Some of them base their strategy on adapting the internal structure of their classifiers [8], [37], while others utilize an ensemble of models [38] to which techniques to induce diversity are applied [39]. When focused on drift detection approaches, they can be used as a module inside other classifiers to adapt either the classifier's internal structure or the number of classifiers within the ensemble. All of them usually use a specific classifier called base learner, and analyze its performance score (in general, accuracy or error rate) to indicate whether a drift has occurred. This base learner is trained on the current samples, an incremental process repeated for each incoming sample from the stream.

This challenge proposes a novel drift detection method coined as `eSNN-DD` (Evolving Spiking Neural Network for Drift Detection), which hinges on the OeSNN proposed in Chapter 2. Such learning models leverage the representation of information as temporal spikes, based on which a number of spike-time association are learned to that capture temporal associations between a large number of variables in streaming data. One of the most promising flavors of SNNs is the eSNN [133], [134], where the number of spiking neurons evolves incrementally over time to unveil temporal patterns from data. The proposed `eSNN-DD` embraces eSNNs to derive a drift detection mechanism inspired by how spiking neurons evolve along time, which is an inner architectural part of the eSNN learning method. Indeed, as opposed to the majority of drift detection mechanisms, `eSNN-DD` does not require any additional *base learner* to analyze its performance metrics towards indicating when a drift

has occurred. Similarly to other drift detection methods, eSNN-DD is independent of the learning/adapting algorithm and can therefore be seamlessly combined with any algorithmic choice in this regard. We assess the performance of the proposed drift detection method by comparing it to that of a number of consolidated methods from the state of the art, in terms of true positives, false alarms (false positives), missed detection (false negatives) and distance to the drift point. As will be shown by the results of experiments with synthetic data sets, eSNN-DD performs competitively as a drift detector yet it does not need to compare distributions or monitor performance statistics to detect the drift, thus making it easier and computationally lighter to detect drifts.

5.2 Proposed Approach

A detailed description of the eSNN has been already presented in Section 4.2, thus we turn now to elaborate the eSNN-DD approach. The proposed eSNN-DD method is based on the number of merges that occur in a sliding window $\mathbf{W}$ with size $|\mathcal{W}| = W$. While the data distribution remains stable, it is more likely that any of the output neurons in the repository is more similar to the incoming one, and thus there will be more merging processes. In contrast, when a drift occurs the data distribution changes and the resulting neurons start to be different from those in the output neuron repository, correspondingly decreasing the number of merging processes. With this intuition in mind, the proposed detection mechanism considers that a change has taken place when there has not been any merging process W time ticks after the last fusion of neurons in repository $\mathcal{R}$.

Figure 5.1 depicts the differences of the merging processes before and after the drift (sample 1000) for CIRCLE, one of the synthetic data sets compounding the landmark repository contributed in [3] for concept drift and used in previous chapters. This data set contains a single drift starting at sample 1000. During the first 100 samples the input data distribution is frequently new for the eSNN and the merging process is carried out only occasionally as a result of the transient to the stable period. From the first 100 samples onwards, and until drift occurs, the merging process is more usual because the input data distribution remains stable. When the drift occurs, the input data distribution does not resemble the previous concept any longer, and the merging process is not triggered again until the eSNN captures the new distribution (concept). This new knowledge

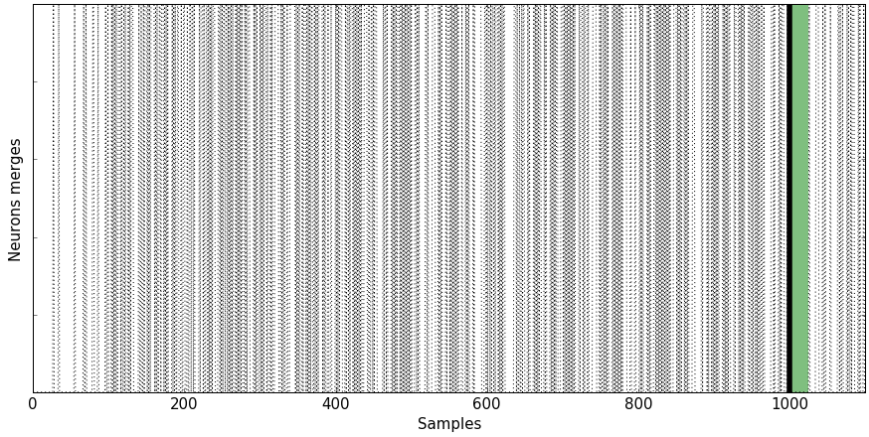


FIGURE 5.1: Time instants at which output neurons are merged for the CIRCLE synthetic data set with high severity and high speed. Before drift occurs at instant 1000, merging processes are held recurrently, whereas right after the drift (area shaded in gray) merges do not occur until the new concept is captured by the eSNN learning algorithm.

is acquired by storing output neurons that represent the new data distribution.

It should be noted that this detection mechanism overrides any need for further processing elements in the model pipeline, as it exploits changes in the structural change of the eSNN in response to newly arriving concepts. Algorithm 9 describes the drift detection mechanism along with the learning procedure of eSNNs. If at least one neuron has been merged in the previous W time instants or the repository size limit $|\mathcal{R}|_{max}$ has not been reached, no drift is declared. In case of a full repository and when a neuron must be merged, the oldest output neuron is discarded yielding one free slot for the newly produced output neuron. When no neuron has been merged over W time instants, then a drift is declared.

Some of the eSNN parameters impact significantly on the performance of `eSNN-DD`. The size W of $\mathcal{W}$ is a key point due to the fact that the method declares a drift when there are no merges during W consecutive time instants. In the case of the repository size $|\mathcal{R}|$, the higher this parameter is, the more likely a similar neuron will be found, thus the number of merges is also affected by this parameter. Finally, *SIM* defines the threshold under which two neurons are regarded as similar and by which the merging process is triggered, hence it drives directly the detection behavior of `eSNN-DD`.

Algorithm 9: Proposed `eSNN-DD` drift detection algorithm

Set an empty repository: $\mathcal{R} = \emptyset$
Set $mod \in \mathbb{R}[0,1]$, $C \in \mathbb{R}[0,1]$, $SIM \in \mathbb{R}[0,1]$
Set eSNN encoding parameters β, T, N and
Set sliding window size W
Set neuron repository maximum size $|\mathcal{R}|_{max}$
Set number of merging operations as $\#_{merges} = W$
foreach *input sample* $\mathbf{x}_i$ *from the stream* **do**
    Update $\mathcal{W}$ with $\mathbf{x}_i$ by discarding the oldest sample in the window
    Calculate C_j and $w_{j,i}$ with the samples in $\mathcal{W}$ to encode $\mathbf{x}$ into firing time of multiple pre-synaptic neurons j
    Create a new output neuron i and the connection weight vectors
    Calculate $PSP_{max,i}$ and $\gamma_i = C \cdot PSP_{max,i}$
    if $\min_{r \in \mathcal{R}} EuclideanDistance(i, r) \leq SIM$ **then**
        Update weight vector & threshold of most similar neuron in $\mathcal{R}$
        $\#_{merges} = \#_{merges} + 1$
    else
        if $|\mathcal{R}| < |\mathcal{R}|_{max}$ **then**
            Add the new neuron to the repository: $\mathcal{R} = \mathcal{R} \cup \{i\}$
        else
            Remove the oldest neuron in $\mathcal{R}$
            Insert the new neuron: $\mathcal{R} = \mathcal{R} \cup \{i\}$
        end
        $\#_{merges} = \#_{merges} + 0$ (no merging performed)
    end
    if $\#_{merges} = 0$ **then**
        Issue a drift alarm
        Set $\#_{merges} = W$
    end
end

5.3 Computer Experiments

A set of experiments has been designed to analyze the behavior of `eSNN-DD` over several synthetic data sets. To this end, we will first find suitable values for the eSNN parameters, and then we will discuss on its detection performance in comparison with drift detectors

from the literature. Before proceeding forward, details on the considered data sets and drift detectors are next given.

When dealing with real data it is not possible to determine exactly when a drift occurs, its type, or if it is actually a drift. Therefore, it is not possible to carry out an analysis of the behavior of a new drift detector by focusing on real-world data sets. For this rationale, 4 synthetic data sets have been chosen, already described in Section 3.4.1.

Drift detection methods for the benchmark were selected based on their impact on the field (e.g. high number of citations), and their thorough description in the literature, which eased their implementation and minimized misunderstanding. All of them resort to a Naive Bayes classifier as their *base learner* [26], [27] due to the efficiency and simplicity of this class of models. As such, the first considered method is `DDM` [29], which departs from the intuitive fact that, when the concept changes, the base learner will incorrectly predict the incoming samples that are shaped on a different data distribution. Therefore, when the error rate increases `DDM` asumes that a new concept emerges from the stream. `EDDM` [50] is partly based on `DDM`, but differs in the monitored performance metric (i.e. instead of error rate, `DDM` uses the so-called distance-error-rate). `ADWIN` [27] operates with two sub-windows of a variable-sized window, and it makes a comparison to find differences between them; in such a case it will consider that a drift has occurred. Finally, drift detection methods based on the Hoeffding's bound (`HDDM`, [171]) monitor the mean of the base learner's performance over time, but without assuming any distribution for the incoming data. we will refer as `HDDM_A` to the HDDM approach that uses a window and a moving average as its estimator. Likewise, `HDDM_W` does not use a window and uses EWMA as its estimator, giving a higher weight to the most recent samples.

When it comes to the evaluation of the obtained results, drift detection metrics vary considerably. Nevertheless, we have embraced the recommendations provided in [26]; therefore, comparisons are carried out in terms of FP, TP, FN, DT. FP indicate the number of drifts identified at time instants when no drift was present in the data set whatsoever: therefore, lower values mean that the method does not detect false drift events. Likewise, a high TP value indicates that the method has identified true drifts during a specific period of time, which in this research is given by the drifting time of every data set. FN indicates the number of missed drift detections, which can easily reach 100% because the methods are based

on *test-then-train*. For this reason TP are computed during the drifting time. Finally, DT indicates the average distance to the true drift whenever TP > 1. The accuracy metric is not analyzed because in our experiments eSNN-DD is not hybridized with any base learner so as to focus exclusive on its performance as a drift detector. Only if hybridized with an adaptation method a comparison analysis on accuracy scores could be undertaken.

5.3.1 Configuration of Parameters

In order to find suitable values for the parameters related to DD-eSNN encoding (T, N and β) and learning (*mod* and C), a set of offline experiments was designed to empirically find the configuration best balancing between all detection performance metrics, which resulted to be $T = 20$, $N = 10$, $\beta = 1.5$, $mod = 0.85$ and $C = 0.95$. For those parameters with high impact on the detection (W, $|\mathcal{R}|$ and SIM), different configurations were tested for each of the 16 data sets: $W \in \{20, 25, 30\}$, $|\mathcal{R}| = \{20, 25, 30\}$ and $SIM = \{0.20, 0.25, 0.30\}$. Comparing their results in terms of TP, FP, FN, and DT, the more suitable configuration was $W = 20$, $|\mathcal{R}| = 30$ and $SIM = 0.25$. For the rest of drift detectors, their configuration was retrieved from the literature using each technique. In the case of DDM, the minimum number of samples before the detection of a drift is permitted (*min_samples*) was set to 20, 25 and 30 [29]. As for EDDM, after occurring 30 classification errors this detector uses thresholds $\alpha = 0.95$ and $\beta = 0.9$ to detect a concept drift, and as in DDM *min_samples* equals 30 as suggested in [50]. Regarding ADWIN, the confidence level is $\delta = 0.1$ and the minimum frequency of samples needed for the window size to be reduced is set to $f = 16$ [26]. For HDDM_W the confidence interval for the drift detection is $cid = 0.001$, the confidence interval for the warning is $cid = 0.001$, and the parameter controlling the weight given to the most recent data with respect to older data is $\lambda = 0.05$. Finally, for HDDM_A, $cid = 0.001$, $cid = 0.001$, and the window sizes are set to 5, 20, and 50, which is included in the label as HDDM_A-ω (with $\omega \in \{5, 20, 50\}$).

5.4 Results and Discussion

A first look at the obtained statistics presented in Table 5.1 reveals that eSNN-DD is very competitive in comparison with the rest of drift detectors. Indeed, eSNN-DD shows good results in terms of detection

statistics for both abrupt (data sets with high speed drift dynamics) and gradual drift dynamics (correspondingly, those with low values of drift speed). Specifically, the proposed method never suffers from a non-detection (ND) and FN, a fact that evinces its robustness to missed detections. It can be observed that `eSNN-DD` yields a higher number of false positives than the rest of approaches, but in return it detects the drift earlier, mostly in abrupt data sets. This noted fact suggests the existence of a trade-off between the number of FP and the distance to the drift (detection delay), a particularity that was already revealed in [172], [173] for other detectors. Finally, it is worth mentioning that although `eSNN-DD` operates with a smaller window ($W = 30$) than HDDM_A_50 (respectively, $W = 50$), the proposed detector outperforms this latter counterpart in what regards to the distance to the drift over many data sets, such as `CIRCLE`, `SINEH`, `LINE` (high severity, high speed), `SINEV` (low severity, high speed) and `SINEV` (low severity, low speed).

5.5 Summary

Drift detection in changing environments is a key factor for those active adaptive methods which require trigger mechanisms for drift adaptation. Most approaches are relied on a base learner that provides accuracies or error rates to be analyzed by an algorithm. In this chapter, we have proposed the use of eSNN as a new form of drift detection, which resorts to the own architectural changes of this particular class of models to estimate the drift location without requiring any external base learner. By virtue of its inherent simplicity and lower computational cost, this embedded approach can be suitable for its adoption in online learning scenarios with severe resource constraints. Experiments with synthetic data sets have shown that the proposed technique is very competitive when compared to other drift detection techniques.

CIRCLE																
Methods	Low severity, high speed				Low severity, low speed				High severity, high speed				High severity, low speed			
	DT	FP	TP	FN	DT	FP	TP	FN	DT	FP	TP	FN	DT	FP	TP	FN
ADWIN	20 ± 13.17	26	11	0	273.89 ± 103.92	19	28	0	48 ± 1.41	27	5	0	220.36 ± 93.50	29	25	0
DDM-20	49 ± 0.0	0	1	0	265 ± 0.0	0	1	0	23 ± 0.0	0	1	0	227 ± 0.0	0	1	0
DDM-25	49 ± 0.0	0	1	0	265 ± 0.0	1	1	0	23 ± 0.0	0	1	0	227 ± 0.0	0	1	0
DDM-30	49 ± 0.0	0	1	0	265 ± 0.0	1	1	0	23 ± 0.0	0	1	0	227 ± 0.0	0	1	0
EDDM	29 ± 0.0	0	1	0	339 ± 116	2	2	0	11 ± 0.0	6	1	0	314.33 ± 138.88	4	3	0
HDDM_A-5	17.66 ± 15.86	38	3	0	275.9 ± 138.82	23	10	0	25 ± 24	36	2	0	272.23 ± 126.56	28	13	0
HDDM_A-20	11 ± 0.0	10	1	0	310.5 ± 127.76	4	4	0	5 ± 0.0	5	1	0	255.8 ± 142.37	5	5	0
HDDM_A-50	9 ± 0.0	2	1	0	271 ± 114	2	2	0	25 ± 22	3	2	0	209 ± 123.28	3	3	0
HDDM_W	*ND*	2	0	1	*ND*	3	0	1	23 ± 8	25	2	0	420.66 ± 55.40	18	6	0
eSNN-DD	$\mathbf{3 \pm 0.0}$	21	1	0	$\mathbf{191.80 \pm 107.35}$	19	5	0	*22.5 ± 19.5*	*16*	2	0	$\mathbf{163 \pm 184.11}$	*14*	4	0
LINE																
Methods	Low severity, high speed				Low severity, low speed				High severity, high speed				High severity, low speed			
	DT	FP	TP	FN	DT	FP	TP	FN	DT	FP	TP	FN	DT	FP	TP	FN
ADWIN	20.81 ± 8.36	24	11	0	240.92 ± 152.32	26	14	0	35.66 ± 3.74	36	12	0	260.25 ± 115.33	29	24	0
DDM-20	*ND*	1	0	1	325.5 ± 33.5	0	2	0	21 ± 0.0	0	1	0	156.5 ± 13.5	0	2	0
DDM-25	*ND*	1	0	1	325.5 ± 33.6	0	2	0	21 ± 0.0	0	1	0	164 ± 21	0	2	0
DDM-30	*ND*	1	0	1	325.5 ± 33.6	0	2	0	21 ± 0.0	0	1	0	143 ± 0.0	0	1	0
EDDM	*ND*	3	0	1	206 ± 0.0	1	1	0	9 ± 0.0	0	1	0	258.66 ± 144.98	1	3	0
HDDM_A-5	19 ± 10	32	2	0	216.25 ± 136.34	23	8	0	15 ± 14	42	2	0	235.84 ± 140.90	30	13	0
HDDM_A-20	21 ± 0.0	7	1	0	232 ± 108.07	6	4	0	9 ± 0.0	8	1	0	250.14 ± 119.37	6	7	0
HDDM_A-50	27 ± 0.0	2	1	0	193 ± 0.0	2	1	0	24 ± 19	2	2	0	245.66 ± 156.06	2	3	0
HDDM_W	*ND*	0	0	0	*ND*	0	0	0	39 ± 0.0	17	1	0	439 ± 20	12	2	0
eSNN-DD	27 ± 10	30	2	0	*230.92 ± 134.58*	26	12	0	$\mathbf{9 \pm 0.0}$	*7*	1	0	271.14 ± 130.39	*7*	7	0
SINEH																
Methods	Low severity, high speed				Low severity, low speed				High severity, high speed				High severity, low speed			
	DT	FP	TP	FN	DT	FP	TP	FN	DT	FP	TP	FN	DT	FP	TP	FN
ADWIN	*ND*	48	0	1	*ND*	32	0	1	*ND*	49	0	1	438.91 ± 24.18	35	12	0
DDM-20	*ND*	0	0	0	*ND*	0	0	0	*ND*	1	0	1	*ND*	1	0	1
DDM-25	*ND*	0	0	0	*ND*	0	0	0	*ND*	1	0	1	*ND*	1	0	1
DDM-30	*ND*	0	0	0	*ND*	0	0	0	*ND*	1	0	1	*ND*	1	0	1
EDDM	*ND*	0	0	0	*ND*	0	0	0	*ND*	10	0	1	*ND*	3	0	1
HDDM_A-5	33.33 ± 10.20	67	3	0	271.95 ± 133.97	56	20	0	42 ± 6	71	2	0	315.66 ± 131.99	57	15	0
HDDM_A-20	34 ± 0.0	18	1	0	247.5 ± 120.32	12	6	0	*ND*	20	0	1	291.4 ± 113.99	15	5	0
HDDM_A-50	34 ± 0.0	8	1	0	375 ± 0.0	7	1	0	*ND*	9	1	0	245.33 ± 91.22	6	3	0
HDDM_W	7 ± 0.0	32	1	0	190.4 ± 81.18	26	5	0	25.5 ± 19.5	36	2	0	272.84 ± 128.39	28	13	0
eSNN-DD	$\mathbf{7 \pm 0.0}$	33	1	0	*244.08 ± 119.38*	27	13	0	$\mathbf{7 \pm 0.0}$	**36**	1	0	$\mathbf{242.86 \pm 130.97}$	26	14	0
SINEV																
Methods	Low severity, high speed				Low severity, low speed				High severity, high speed				High severity, low speed			
	DT	FP	TP	FN	DT	FP	TP	FN	DT	FP	TP	FN	DT	FP	TP	FN
ADWIN	40 ± 4.96	37	3	0	347.93 ± 97.49	23	16	0	31.78 ± 9.11	47	14	0	278.89 ± 142.38	27	29	0
DDM-20	*ND*	2	0	1	495 ± 0.0	1	1	0	19 ± 0.0	0	1	0	174.5 ± 12.5	0	2	0
DDM-25	*ND*	2	0	1	495 ± 0.0	1	1	0	19 ± 0.0	0	1	0	186.5 ± 24.5	0	2	0
DDM-30	*ND*	2	0	1	495 ± 0.0	0	1	0	19 ± 0.0	0	1	0	187 ± 25	0	2	0
EDDM	31 ± 0.0	2	1	0	349 ± 0.0	0	1	0	*ND*	5	0	1	357.33 ± 106.55	4	3	0
HDDM_A-5	19 ± 10	38	2	0	244.3 ± 139.61	26	10	0	*ND*	39	0	1	256.07 ± 130.20	28	13	0
HDDM_A-20	17 ± 0.0	9	1	0	273 ± 82	5	2	0	1 ± 0.0	10	1	0	254.66 ± 140.66	*7*	6	0
HDDM_A-50	*ND*	1	0	1	359 ± 0.0	0	1	0	22 ± 21	3	2	0	269.33 ± 159.70	3	3	0
HDDM_W	*ND*	0	0	0	*ND*	0	0	0	41 ± 0.0	20	1	0	382.33 ± 84.21	11	6	0
eSNN-DD	$\mathbf{5 \pm 0.0}$	*28*	1	0	*264.55 ± 158.74*	*22*	11	0	*24.50 ± 11.50*	*11*	2	0	*310.40 ± 91.79*	*5*	5	0

TABLE 5.1: Drift detection statistics for the synthetic data sets. Those statistics in which eSNN-DD shows better results are in bold, and in italics when the results are just competitive. *ND* refers to a simulation where no drift is detected within the drifting period.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis has provided an overview of the current state of the art for the *online learning* and *concept drift* fields. After that, it has identified a set of open challenges, and we have selected some of them to be part of this thesis. The addressed challenges stand out among others for their high relevance, being highlighted in recent publications by prominent authors of the field. By addressing these challenges, we have provided new perspectives and methods that take a step further beyond the current state of the art.

Regarding the first addressed challenge of this thesis work, Chapter 3 has elaborated on a new approach for generating diversity in ensembles for online learning scenarios, considering the need for achieving a good balance between predictive performance in stable data distributions and adaptability to concept drifts. Coined as DRED, this approach has proven to be a suitable additive for the traditional diversity-induction methods such as `BAGGING`, `SWITCHING`, and `BOOSTING`, improving their performance in those cases where the dynamics of the drift and the data set itself are characterized by a shape that does fit well with the Gaussian kernel. Under this circumstance, the generation of synthetic samples produced by DRED manages to promote the diversity artificially and prepares the algorithm for the drift period. These artificial data is subsequently optimally labeled by a multi-objective optimization technique, allowing to find a range of Pareto-optimal labels with different levels of diversity in order to train the ensemble. The Bhattacharyya distance has been utilized throughout the study, demonstrating its capability to serve as diversity measure to build diverse ensembles, being

the basis of an optimization process focused on keeping the balance between stability and plasticity during the adaptation.

Computer experiments have been carried out over artificial data sets of common use in the related literature. Interestingly, DRED hybrids have been found to outperform traditional approaches more remarkably when the new concept grows incrementally upon the prior distribution of classes over the feature space. While this particularity does pose a challenge for the proposed diversity generation scheme, it is also a niche of opportunity towards refining the synthetic sample generation process so as to accommodate non-incremental drift dynamics. This motivates further research dealing with different approximations to incorporate such dynamics during the sample generation process.

Finally, DRED can be hybridized with any drift detector of the literature, especially interesting with those that provide useful information about the dynamics of the drift, such as the duration of the drift, its severity and speed, the size of the window to store past data, even the speed and shape of the evolution of the feature space after the change occurs. At this stage, more drift detectors are needed to infer information about the type of drift (abrupt or gradual) and even predict how the feature space will evolve over time.

Continuing with the second addressed challenge, Chapter 4 has presented a portfolio of new adaptations of eSNNs for online learning scenarios under concept drift. Firstly, the traditional eSNN technique has been adapted to be used over online data streams by limiting the size of the neurons repository, yielding the so-called OeSNN. Secondly, we have embraced the use of selective and generative data reduction techniques (DRTs) to optimize the contents of the neurons repository so as to achieve a better adaptability of the model to changing concepts over the processed data stream. Both passive and active strategies have been defined to incorporate DRTs into the OeSNN learning procedure: the active comprises a drift detector that detects changes along the data stream and triggers the application of the DRT at hand to the neurons repository. Two different families of OeSNN models have been proposed: OeSNN-PS (using prototype-selection DRTs) and OeSNN-PG (using prototype-generation DRTs), both capable of operating in passive and active modes when processing data streams.

An extensive set of computer experiments over synthetic and real streaming data sets has been designed and discussed to find performance differences between the above approaches. Part of the

OeSNN-PS variants (OeSNN-SMMA and OeSNN-CNN) and all OeSNN-PG approaches (i.e. OeSNN-SGP, OeSNN-SGP2, and OeSNN-ASGP) have been proven to attain better predictive scores during plasticity periods than the naive version of the OeSNN (i.e. the proposed OeSNN with no DRT included in its learning algorithm). The application of DRTs to the proposed OeSNN model also allows reducing the required space to store output neurons, thus decreasing the processing time needed to train with newly samples: the less neurons in the repository, the less similarity computations during the learning phase. This alleviation of the computational resources demanded by the model is of utmost importance in online learning, where processing times and storage should be kept as low as possible to process high stream data rates.

Those approaches that use DRTs achieving the lowest occupancy of the neurons repository retain the old concept by generating or selecting prototypes, and at the same time they adapt better to the new concept by storing more information (output neurons) of the new concept during the plasticity period after a drift occurs. This is possible due to the fact that high data reduction ratios lead to more available space in the neurons repository where to store neurons with more recent information. Thus, the adaptation to the new concept is better (a higher accuracy is achieved) and faster (less similarity computations in the training phase as argued previously). Although the selection of a data reduction technique is not straightforward (it depends on the characteristics of the data set), empirical evidences indicate that OeSNN-PG approaches feature a better adaptability right after the drift occurs, while performing very competitively in stability periods.

In summary, this challenge brings to light the natural capability of the proposed family of OeSNN-DRTs to:

1. simultaneously learn the new incoming concept while retaining the older one without incorporating any forgetting mechanisms ($\mathcal{W}$ is used just to compute encoding parameters, there is no windowed adaptation whatsoever as Section 4.2.2 clearly shows);

2. update the model without requiring a retraining mechanism;

3. use less capacity storage by reducing the amount of required

neurons for the overall model to achieve a better balance between stability (performance over stationary data distributions) and plasticity (reaction against drift events in the processed data stream);

4. be faster than the traditional OeSNN approach by carrying out less similarity computations (less output neurons) in the training phase;

5. be competitive in terms of their balance between predictive performance and complexity in comparison with ensembles of classifiers, which makes OeSNN-DRTs a better match for scenarios under severe computational restrictions; and

6. to provide a realistic solution to be hybridized with a drift detector.

Finally, the third challenge addressed in this thesis (Chapter 5) has been elaborated on a novel drift detection technique coined as `eSNN-DD`, which does not rely on the performance of any base learner as opposed to other approaches from the literature utilized for drift detection. Instead, the proposed approach exploits the structural changes (neural merges) of an eSNN predictive model to identify the drift point, hence it can be regarded as an *embedded* drift detection approach that stems as a byproduct of the particular learning algorithm of this class of neural networks. Contributions dealing with drift detection often build upon the hybridization of a classifier and a detection criteria. The embedded strategy suggested in this challenge can be useful to simplify the drift detection process in those applications with stringent computational constraints. Nevertheless, `eSNN-DD` can also work in collaboration with any other adaptive technique, roughly like most of the other detection approaches. A set of computer experiments with synthetic data sets has shed light on the competitive behavior of `eSNN-DD` when compared to other well-known drift detectors.

6.2 Future Work

6.2.1 Addressed Challenges

Regarding the first addressed challenge of this thesis, future efforts will be invested on several research directions. On the one hand, it is essential to address the aforementioned need for incorporating

a priori inferred information on the dynamics of the drift during the sample generation process. This would allow steering newly produced data towards regions in the feature space where the new concept is evolving. It is also interesting to study further the performance of this technique when new input variables appear *on the fly*, or even when new class labels emerge in the data stream. Regarding the Bhattacharyya distance,it will be also interesting to analyze the utility of this measure in comparison with other traditional diversity measures from the field. Additionally, the practical adaptability of DRED when complementing drift detectors in real data sets could be analyzed, with emphasis on its resiliency against false detections. Finally, it is worth mentioning that, when the choice of the optimal kernel bandwidth is computationally more intensive than the online requirements impose by the system or application, it has been recently presented a solution called fastKDE [174], which is more computationally efficient. A new version of DRED considering fastKDE could be a good solution for those constricted scenarios.

Continuing with the second addressed challenge, and as mentioned before for the first challenge, the incorporation of information of dynamics of the drift (concretely severity and velocity), which can be estimated in practice when drifts occur in a recurrent fashion, could be very useful to choose the suitable data reduction technique for the plasticity period. Besides, it would be also interesting to explore the posibility of building ensembles of OeSNNs and study their behaviour in comparison with other ensemble approaches.

In the third addressed challenge, the extensive offline search for suitable `eSNN-DD` parameters suggests that in the near future research efforts should be invested towards the use of online optimization techniques to tailor the parametric configuration of the detector while in operation. A first portion of the data set could be used to find a proper initial configuration, after which parameters could be tuned slightly during the stream mining process. Besides this research path, it would be interesting to find out whether the learning procedure of eSNN models can unveil more characteristics of the drift than its occurrence (e.g. duration, severity). Finally, it is necessary to assess the performance of `eSNN-DD` over more diverse data sets, encompassing not only more complex concepts (number of features and/or number of classes), but also different types of drifts (such as recurring or incremental). Additionally, further research towards quantifying the diversity of the output neurons could be helpful in order to develop new innovative approaches for drift adaptation and detection.

Finally, the fusion of eSNN with DRED could be very interesting as a new drift adaptation approach. For this purpose, it is necessary first to undertake a deeper study of the eSNNs parameters configuration, and how the optimization of this process could be performed in a stream learning process. Besides, it also becomes necessary to delve further into encoding techniques that are used in eSNNs.

6.2.2 Others

Other research interests for the future that are also related to the addressed challenges of this thesis are next outlined.

Diversity and Self-Configuring Ensembles

Not much attention has been given to self-configuring ensembles problem in the state of the art, as the author of [175] revealed, and more recently in [18]. Many efforts have been devoted to the self-configuring field using evolutionary computation in traditional machine learning problems (e.g. hyper-parameter tuning), but not to online learning scenarios from a diversity point of view. Nevertheless, the work in [176] carried out extensive experiments to study different adaptive forms: the first concerns the self-organizing nature of the base learners in the ensemble; the second is about the weighted contribution of the base learners when incorporated into an incremental ensemble, and the third is about the incremental and dynamic update over time of the structure of the ensemble.

Most online and batch learning approaches use models with parameters that require individual tuning or they use preset values fixed for the whole analysis process. However, after a change occurs in the data stream, these preset values-fixed parameters may no longer be good for the task at hand (especially in case of parameter-sensitive techniques such as SVM or Neural Networks). Therefore, researching on a new methodology for self-tuning streaming ensemble systems may lead to improved predictive power. Besides, parameter tuning for single classifiers should contemplate that they are members within the ensemble. Thus, more global update methods that can lead to obtain more complementary models seems to be worth exploring in line with recent developments contributed in [177].

eSNN as a General-Purpose Lifelong Learning Algorithm

Nowadays, we find two relevant limitations of the Artificial Intelligence (AI). On the one hand, AI is only as smart as the data sets served, which means that any inaccuracies in the data will be reflected in the AI results. On the other hand, AI is one-purpose designed, it is trained to do a clearly defined task, that is, an AI model that plays chess cannot play solitaire or poker. This latter limitation is more restricting than the former one, due to the fact that, despite nowadays we have enough different high quality data sets to train an AI to perform different tasks, we cannot do it because of the second limitation. In this regard, this caveat has lately stimulated one big area of research typically referred to as *transfer learning* [178], which is based on the idea of how to take models or learnings or insights gained from one task and extrapolate them to another task. Despite progress in *transfer learning*, it is actually one of the hardest problems to solve.

Recently, AlphaGo Zero [179], the latest evolution of AlphaGo (the first computer program to defeat a world champion at the ancient Chinese game of Go) is able to play three different games but has just a generalized structure of games. It is able to do this by using a novel form of *reinforcement learning* [180], but even that is limited in the sense that it is still limited to games that take a certain form. *Reinforcement learning* is a type of unsupervised machine learning where an AI learns to interact with an environment (e.g. a game) entirely through trial and error.

With machine learning, given enough quality example data, the same neural network algorithm, or at least a suite of similar network models, can be trained to solve both face detection and speech recognition problems. By virtue of recent advancements in *deep learning* [181], machines are beginning to see, hear, and understand the world around them in ways and volumes that humans simply cannot (*multimodal learning*) [182]. Innovations in Convolutional Neural Networks [183] have enjoyed widespread use in batch processing the billions of images that are uploaded to social networks, extracting information that allows them to identify people and contextualize their surroundings to build up a profile of the people that use their services.

Recent implementations of Recurrent Neural Networks (RNNs) [184], particularly those using Long Short Term Memory (LSTM)

processing units as neurons are becoming increasingly able to synthesize sequential patterns like text, speech, and even music. Towards this end, perhaps one of the most promising advancements in machine learning over the past few years are Generative Adversarial Networks (GANs) [185]. GANs are sophisticated generative models that are able to generate stunningly accurate synthesized images of objects, people, and places among other things. Research into GANs is very new and the results are already overwhelmingly promising.

All these subfields (*transfer learning*, *reinforcement learning*, *online learning*, *multi-task learning*, *knowledge representation and maintenance*) are usually framed within the terms Lifelong Machine Learning, or Continuous Learning, or Continual Learning. It is now a fast emerging paradigm in AI, which focuses on developing versatile systems that accumulate and refine their knowledge over time [12], [13].

LML is built on the idea of learning continuously and adaptively about the external world and enabling the autonomous incremental development of ever more complex skills and knowledge. In the context of machine learning it means being able to smoothly update the prediction model to take into account different tasks and data distributions but still being able to reuse and retain useful knowledge and skills during time. Hence, LML is a paradigm which force us to deal with an higher and realistic time-scale where data (and tasks) become available only during time, we have no access to previous perceived information and it is imperative to build on top of previously learned knowledge. In [186], it has been recently explored the connecting link between LML and Neural Networks.

As Pedro Domingos (the author of The Master Algorithm [187]) explains, we have used the same basic paradigms since the 1950s, thereby, we are going to eventually need some new ideas. In this line of reasoning, we have seen how eSNN have revealed themselves as one of the most successful approaches to model the behavior and learning potential of the brain, and exploit them to undertake practical learning tasks. Future work should be invested in developing an eSNN capable of learning in an online manner from different data sets at the same time, exploit their relationships, and study how the eSNN evolves and which are its limits as a bigger size of neuron repository is available and new data sets are added to the data stream. This would provide a new approach to be considered under the LML paradigm.

Appendix A

Challenge 1

A.1 Experimental Results for All Data Sets

Note: Those results in which one of the approaches shows a better performance have been highlighted in bold. The statistical significance of the obtained performance gaps has been assessed via a non-parametric Wilcoxon rank-sum test with a confidence interval of 95%. BD, D and AD correspond to Before Drift, Drift and After Drift, respectively.

	Sev.	Sp.	Region	DRED-OB	BAGGING	DRED-CS	SWITCHING	DRED-BOOST	BOOSTING
CIRCLE	1	1	BD	0.976 ± 0.001	0.974 ± 0.008	0.972 ± 0.009	0.972 ± 0.001	0.974 ± 0.004	0.972 ± 0.004
	1	1	D	$\mathbf{0.950 \pm 0.003}$	0.780 ± 0.001	0.970 ± 0.001	0.970 ± 0.002	$\mathbf{0.960 \pm 0.001}$	0.890 ± 0.009
	1	1	AD	$\mathbf{0.978 \pm 0.003}$	0.852 ± 0.002	$\mathbf{0.976 \pm 0.002}$	0.870 ± 0.011	$\mathbf{0.973 \pm 0.002}$	0.852 ± 0.003
	1	3	BD	0.977 ± 0.005	0.975 ± 0.005	0.972 ± 0.007	0.972 ± 0.001	0.974 ± 0.005	0.972 ± 0.001
	1	3	D	0.970 ± 0.004	0.970 ± 0.001	$\mathbf{0.970 \pm 0.005}$	0.950 ± 0.013	$\mathbf{0.960 \pm 0.011}$	0.920 ± 0.006
	1	3	AD	$\mathbf{0.939 \pm 0.002}$	0.864 ± 0.003	$\mathbf{0.915 \pm 0.001}$	$0.866 \pm .009$	$\mathbf{0.962 \pm 0.002}$	0.870 ± 0.005
	3	1	BD	0.980 ± 0.007	0.975 ± 0.006	0.972 ± 0.008	0.972 ± 0.008	0.974 ± 0.007	0.973 ± 0.008
	3	1	D	$\mathbf{0.890 \pm 0.007}$	0.670 ± 0.008	$\mathbf{0.860 \pm 0.006}$	0.780 ± 0.008	$\mathbf{0.880 \pm 0.008}$	0.760 ± 0.010
	3	1	AD	$\mathbf{0.885 \pm 0.007}$	0.706 ± 0.008	$\mathbf{0.874 \pm 0.008}$	0.714 ± 0.007	$\mathbf{0.848 \pm 0.009}$	0.564 ± 0.009
	3	3	BD	0.972 ± 0.003	0.975 ± 0.005	0.972 ± 0.006	0.972 ± 0.005	0.970 ± 0.008	0.972 ± 0.007
	3	3	D	0.940 ± 0.002	0.940 ± 0.004	$\mathbf{0.940 \pm 0.004}$	0.920 ± 0.001	$\mathbf{0.940 \pm 0.005}$	0.910 ± 0.001
	3	3	AD	0.738 ± 0.006	0.740 ± 0.007	$\mathbf{0.759 \pm 0.008}$	0.739 ± 0.009	$\mathbf{0.766 \pm 0.006}$	0.741 ± 0.002

TABLE A.1: Average prequential accuracy results (mean $\pm$ standard deviation) obtained by the diversity generation schemes in the benchmark for the CIRCLE data set with several severity and speed levels.

	Sev.	Sp.	Region	DRED-OB	BAGGING	DRED-CS	SWITCHING	DRED-BOOST	BOOSTING
LINE	1	1	BD	0.980 ± 0.005	0.978 ± 0.003	0.979 ± 0.009	0.979 ± 0.003	0.977 ± 0.006	0.978 ± 0.008
	1	1	D	$\mathbf{0.890 \pm 0.003}$	0.870 ± 0.003	0.880 ± 0.004	0.880 ± 0.004	$\mathbf{0.900 \pm 0.001}$	0.880 ± 0.004
	1	1	AD	0.914 ± 0.007	0.909 ± 0.005	0.916 ± 0.002	0.910 ± 0.004	$\mathbf{0.918 \pm 0.003}$	0.905 ± 0.005
	1	3	BD	0.975 ± 0.006	0.978 ± 0.006	0.979 ± 0.006	0.979 ± 0.005	0.970 ± 0.003	0.969 ± 0.006
	1	3	D	0.950 ± 0.007	$\mathbf{0.960 \pm 0.009}$	$\mathbf{0.950 \pm 0.005}$	0.940 ± 0.006	0.950 ± 0.005	$\mathbf{0.970 \pm 0.002}$
	1	3	AD	0.926 ± 0.006	0.925 ± 0.005	0.925 ± 0.008	0.922 ± 0.002	0.929 ± 0.010	0.922 ± 0.002
	3	1	BD	0.973 ± 0.002	0.975 ± 0.002	0.974 ± 0.004	0.974 ± 0.003	0.971 ± 0.001	0.973 ± 0.003
	3	1	D	$\mathbf{0.650 \pm 0.001}$	0.630 ± 0.005	0.650 ± 0.005	0.650 ± 0.001	$\mathbf{0.650 \pm 0.005}$	0.640 ± 0.004
	3	1	AD	0.746 ± 0.004	0.748 ± 0.004	0.754 ± 0.007	0.748 ± 0.007	$\mathbf{0.758 \pm 0.005}$	0.738 ± 0.008
	3	3	BD	0.971 ± 0.004	0.966 ± 0.006	0.974 ± 0.007	0.974 ± 0.004	0.974 ± 0.004	0.972 ± 0.002
	3	3	D	0.880 ± 0.008	$\mathbf{0.930 \pm 0.003}$	$\mathbf{0.890 \pm 0.008}$	0.840 ± 0.005	0.910 ± 0.005	$\mathbf{0.930 \pm 0.002}$
	3	3	AD	0.767 ± 0.007	0.767 ± 0.007	0.767 ± 0.007	0.760 ± 0.006	0.770 ± 0.007	0.764 ± 0.004

TABLE A.2: Average prequential accuracy results (mean $\pm$ standard deviation) obtained by the diversity generation schemes in the benchmark for the LINE data set with several severity and speed levels.

	Sev.	Sp.	Region	DRED-OB	BAGGING	DRED-CS	SWITCHING	DRED-BOOST	BOOSTING
SINEV	1	1	BD	0.970 ± 0.001	0.965 ± 0.005	0.965 ± 0.005	0.965 ± 0.005	0.966 ± 0.007	0.965 ± 0.005
	1	1	D	$\mathbf{0.860 \pm 0.006}$	0.850 ± 0.004	0.870 ± 0.007	0.870 ± 0.006	$\mathbf{0.880 \pm 0.006}$	0.850 ± 0.006
	1	1	AD	0.918 ± 0.002	0.909 ± 0.003	0.920 ± 0.002	0.920 ± 0.002	$\mathbf{0.925 \pm 0.005}$	0.910 ± 0.009
	1	3	BD	0.967 ± 0.006	0.974 ± 0.004	0.965 ± 0.005	0.965 ± 0.005	0.966 ± 0.002	0.966 ± 0.006
	1	3	D	$\mathbf{0.980 \pm 0.001}$	0.970 ± 0.004	$\mathbf{0.970 \pm 0.007}$	0.940 ± 0.004	0.970 ± 0.007	0.970 ± 0.005
	1	3	AD	0.938 ± 0.006	0.927 ± 0.007	0.937 ± 0.006	0.928 ± 0.008	$\mathbf{0.940 \pm 0.005}$	0.920 ± 0.002
	3	1	BD	0.967 ± 0.007	0.969 ± 0.006	0.968 ± 0.006	0.968 ± 0.003	0.969 ± 0.004	0.966 ± 0.005
	3	1	D	$\mathbf{0.650 \pm 0.005}$	0.620 ± 0.002	0.670 ± 0.007	$\mathbf{0.740 \pm 0.007}$	0.660 ± 0.001	0.660 ± 0.004
	3	1	AD	0.731 ± 0.001	0.724 ± 0.004	0.740 ± 0.004	0.740 ± 0.002	0.731 ± 0.004	0.724 ± 0.004
	3	3	BD	0.970 ± 0.007	0.969 ± 0.007	0.968 ± 0.008	0.968 ± 0.008	0.969 ± 0.009	0.964 ± 0.006
	3	3	D	0.900 ± 0.005	$\mathbf{0.920 \pm 0.002}$	$\mathbf{0.900 \pm 0.004}$	0.840 ± 0.006	0.910 ± 0.002	$\mathbf{0.920 \pm 0.001}$
	3	3	AD	0.778 ± 0.008	0.778 ± 0.008	$\mathbf{0.784 \pm 0.004}$	0.770 ± 0.002	0.778 ± 0.005	0.778 ± 0.008

TABLE A.3: Average prequential accuracy results (mean $\pm$ standard deviation) obtained by the diversity generation schemes in the benchmark for the SINEV data set with several severity and speed levels.

	Sev.	Sp.	Region	DRED-OB	BAGGING	DRED-CS	SWITCHING	DRED-BOOST	BOOSTING
SINEH	1	1	BD	0.731 ± 0.001	0.731 ± 0.003	0.728 ± 0.001	0.728 ± 0.008	$\mathbf{0.728 \pm 0.003}$	0.716 ± 0.006
	1	1	D	0.630 ± 0.003	$\mathbf{0.650 \pm 0.005}$	0.630 ± 0.006	$\mathbf{0.650 \pm 0.005}$	$\mathbf{0.670 \pm 0.007}$	0.650 ± 0.005
	1	1	AD	0.713 ± 0.005	0.716 ± 0.004	0.712 ± 0.002	0.719 ± 0.009	0.715 ± 0.005	0.713 ± 0.003
	1	3	BD	0.731 ± 0.004	0.739 ± 0.009	0.728 ± 0.002	0.728 ± 0.004	$\mathbf{0.726 \pm 0.007}$	0.716 ± 0.003
	1	3	D	0.760 ± 0.006	$\mathbf{0.790 \pm 0.007}$	0.760 ± 0.006	$\mathbf{0.770 \pm 0.006}$	0.780 ± 0.008	0.780 ± 0.006
	1	3	AD	0.733 ± 0.003	0.733 ± 0.008	0.728 ± 0.007	0.732 ± 0.002	0.730 ± 0.003	0.733 ± 0.001
	3	1	BD	0.732 ± 0.002	0.729 ± 0.002	0.728 ± 0.008	0.728 ± 0.004	$\mathbf{0.728 \pm 0.002}$	0.712 ± 0.002
	3	1	D	0.590 ± 0.009	$\mathbf{0.640 \pm 0.006}$	0.580 ± 0.005	$\mathbf{0.620 \pm 0.006}$	0.590 ± 0.009	$\mathbf{0.640 \pm 0.003}$
	3	1	AD	0.627 ± 0.007	0.634 ± 0.003	0.624 ± 0.004	$\mathbf{0.634 \pm 0.004}$	$\mathbf{0.627 \pm 0.007}$	0.612 ± 0.006
	3	3	BD	0.731 ± 0.005	0.731 ± 0.001	0.728 ± 0.001	0.728 ± 0.008	$\mathbf{0.726 \pm 0.006}$	0.716 ± 0.004
	3	3	D	0.780 ± 0.006	$\mathbf{0.800 \pm 0.007}$	0.770 ± 0.007	$\mathbf{0.780 \pm 0.007}$	0.800 ± 0.004	0.800 ± 0.008
	3	3	AD	0.664 ± 0.004	0.667 ± 0.007	0.663 ± 0.003	0.664 ± 0.004	0.665 ± 0.003	0.664 ± 0.005

TABLE A.4: Average prequential accuracy results (mean $\pm$ standard deviation) obtained by the diversity generation schemes in the benchmark for the SINEH data set with several severity and speed levels.

	Region	DRED-OB	BAGGING	DRED-CS	SWITCHING	DRED-BOOST	BOOSTING
SEA	BD1	0.874 ± 0.004	0.874 ± 0.008	0.874 ± 0.006	0.874 ± 0.004	0.874 ± 0.005	0.872 ± 0.006
	D1	$\mathbf{0.830 \pm 0.007}$	0.814 ± 0.006	0.828 ± 0.008	0.824 ± 0.009	$\mathbf{0.830 \pm 0.007}$	0.812 ± 0.004
	AD1-BD2	0.833 ± 0.006	0.833 ± 0.008	0.834 ± 0.007	0.834 ± 0.006	0.833 ± 0.006	0.834 ± 0.007
	D2	$\mathbf{0.802 \pm 0.007}$	0.730 ± 0.008	0.800 ± 0.008	0.792 ± 0.006	$\mathbf{0.802 \pm 0.008}$	0.756 ± 0.009
	AD2-BD3	0.840 ± 0.009	0.839 ± 0.006	0.838 ± 0.006	0.839 ± 0.007	0.840 ± 0.008	0.837 ± 0.007
	D3	$\mathbf{0.808 \pm 0.008}$	0.774 ± 0.009	0.810 ± 0.006	$\mathbf{0.828 \pm 0.007}$	0.808 ± 0.008	$\mathbf{0.822 \pm 0.006}$
	AD3	0.830 ± 0.008	0.829 ± 0.007	0.832 ± 0.007	0.832 ± 0.009	0.830 ± 0.006	0.831 ± 0.006

TABLE A.5: Average prequential accuracy results (mean $\pm$ standard deviation) obtained by the diversity generation schemes in the benchmark for the SEA data set. BD, D and AD correspond to Before Drift, Drift, and After Drift, respectively. As such, BD-1 denotes the period *before drift 1* present in the SEA data set.

	Region	DRED-OB	BAGGING	DRED-CS	SWITCHING	DRED-BOOST	BOOSTING
ELEC2	BD	$\mathbf{0.593 \pm 0.002}$	0.582 ± 0.007	0.596 ± 0.006	0.590 ± 0.003	0.565 ± 0.005	0.565 ± 0.007
	D	$\mathbf{0.652 \pm 0.005}$	0.630 ± 0.007	$\mathbf{0.662 \pm 0.007}$	0.617 ± 0.008	0.625 ± 0.006	0.625 ± 0.004
	AD	0.588 ± 0.005	0.585 ± 0.006	0.594 ± 0.007	0.591 ± 0.006	0.558 ± 0.005	0.558 ± 0.008

TABLE A.6: Average prequential accuracy results (mean $\pm$ standard deviation) obtained by the diversity generation schemes in the benchmark for the ELEC2 data set. BD, D and AD correspond to average values Before Drift, Drift, and After Drift, respectively, measured over the 50 periods between the detected drifts.

Bibliography

[1] Z.-H. Zhou, N. V. Chawla, Y. Jin, and G. J. Williams, "Big data opportunities and challenges: Discussions from data analytics perspectives [discussion forum]", *IEEE Computational Intelligence Magazine*, vol. 9, no. 4, pp. 62–74, 2014.

[2] C. Alippi, *Intelligence for embedded systems*. Springer, 2014.

[3] L. L. Minku, A. P. 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*, vol. 22, no. 5, pp. 730–742, 2010.

[4] P. Domingos and G. Hulten, "A general framework for mining massive data streams", *Journal of Computational and Graphical Statistics*, vol. 12, no. 4, pp. 945–949, 2003.

[5] I. Triguero, S. García, and F. Herrera, "Ipade: Iterative prototype adjustment for nearest neighbor classification", *IEEE Transactions on Neural Networks*, vol. 21, no. 12, pp. 1984–1990, 2010.

[6] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Discussion and review on evolving data streams and concept drift adapting", *Evolving Systems*, vol. 9, no. 1, pp. 1–23, 2018.

[7] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification", *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, p. 23, 2017.

[8] L. L. Minku and X. Yao, "Ddd: A new ensemble approach for dealing with concept drift", *IEEE transactions on knowledge and data engineering*, vol. 24, no. 4, pp. 619–633, 2012.

[9] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions", *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation", *ACM computing surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[11] I. Žliobaitė, "Learning under concept drift: An overview", *ArXiv preprint arXiv:1010.4784*, 2010.

[12] D. L. Silver, Q. Yang, and L. Li, "Lifelong machine learning systems: Beyond learning algorithms.", in *AAAI Spring Symposium: Lifelong Machine Learning*, vol. 13, 2013, p. 05.

[13] Z. Chen and B. Liu, "Lifelong machine learning", *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 10, no. 3, pp. 1–145, 2016.

[14] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer, "A survey on feature drift adaptation: Definition, benchmark, challenges and future directions", *Journal of Systems and Software*, vol. 127, pp. 278–294, 2017.

[15] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey", *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[16] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift", *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[17] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications", in *Big Data Analysis: New Algorithms for a New Society*, Springer, 2016, pp. 91–114.

[18] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey", *Information Fusion*, vol. 37, pp. 132–156, 2017.

[19] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift", *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–20, 2018.

[20] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures", *Neural Networks*, vol. 1, no. 1, pp. – 17–61, 1988.

[21] G. Jaber, A. Cornuéjols, and P. Tarroux, "Online learning:-searching for the best forgetting strategy under concept drift", in *International Conference on Neural Information Processing*, Springer, 2013, pp. 400–408.

[22] E. K. Tang, P. N. Suganthan, and X. Yao, "An analysis of diversity measures", *Machine learning*, vol. 65, no. 1, pp. 247–271, 2006.

[23] N. M. Nasrabadi, "Pattern recognition and machine learning", *Journal of electronic imaging*, vol. 16, no. 4, p. 049 901, 2007.

[24] O. Chapelle, B. Scholkopf, and A. Zien, "Semi supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]", *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[25] L. L. Minku, "Online ensemble learning in the presence of concept drift", PhD thesis, University of Birmingham, 2011.

[26] P. M. Gonçalves Jr., S. G. de Carvalho Santos, R. S. Barros, and D. C. Vieira, "A comparative study on concept drift detectors", *Expert Systems with Applications*, vol. 41, no. 18, pp. 8144–8156, 2014.

[27] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing", in *SIAM International Conference on Data Mining*, 2007, pp. 443–448.

[28] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers–part ii: Designing the classifier", *IEEE Transactions on Neural Networks*, vol. 19, no. 12, pp. 2053–2064, 2008.

[29] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection", in *Brazilian Symposium on Artificial Intelligence*, Springer, 2004, pp. 286–295.

[30] A. Bifet and R. Gavalda, "Kalman filters and adaptive windows for learning in data streams", in *International Conference on Discovery Science*, Springer, 2006, pp. 29–40.

[31] E. Cohen and M. Strauss, "Maintaining time-decaying stream aggregates", in *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ACM, 2003, pp. 223–233.

[32] R. Klinkenberg, "Learning drifting concepts: Example selection vs. example weighting", *Intelligent data analysis*, vol. 8, no. 3, pp. 281–300, 2004.

[33] J. S. Vitter, "Random sampling with a reservoir", *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[34] C. C. Aggarwal, "On biased reservoir sampling in the presence of stream evolution", in *Proceedings of the 32nd international conference on Very large data bases*, VLDB Endowment, 2006, pp. 607–618.

[35] W. Ng and M. Dash, "A test paradigm for detecting changes in transactional data streams", in *International Conference on Database Systems for Advanced Applications*, Springer, 2008, pp. – 204–219.

[36] J. Demšar and Z. Bosnić, "Detecting concept drift in data streams using model explanation", *Expert Systems with Applications*, vol. 92, pp. 546–559, 2018.

[37] P. M. Gonçalves Jr. and R. S. M. De Barros, "Rcd: A recurring concept drift framework", *Pattern Recognition Letters*, vol. 34, no. 9, pp. 1018–1025, 2013.

[38] M. Dehghan, H. Beigy, and P. ZareMoodi, "A novel concept drift detection method in data streams using ensemble classifiers", *Intelligent Data Analysis*, vol. 20, no. 6, pp. 1329–1350, 2016.

[39] D. Brzezinski and J. Stefanowski, "Ensemble diversity in evolving data streams", in *International Conference on Discovery Science*, Springer, 2016, pp. 229–244.

[40] C. Alippi, G. Boracchi, and M. Roveri, "Hierarchical change detection tests", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 2, pp. 246–258, 2017.

[41] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines.", in *ICML*, 2000, pp. 487–494.

[42] A. Wald, *Sequential analysis*. Courier Corporation, 1973.

[43] E. S. Page, "Continuous inspection schemes", *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[44] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift", *Pattern recognition letters*, vol. 33, no. 2, pp. 191–198, 2012.

[45] N. Lu, G. Zhang, and J. Lu, "Concept drift detection via competence models", *Artificial Intelligence*, vol. 209, pp. 11–28, 2014.

[46] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer, "Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances", *Information Sciences*, vol. 355, pp. 127–151, 2016.

[47] A. Dries and U. Rückert, "Adaptive concept drift detection", *Analysis and Data Mining: The ASA Data Science Journal*, vol. 2, no. 5-6, pp. 311–327, 2009.

[48] J. H. Friedman and L. C. Rafsky, "Multivariate generalizations of the wald-wolfowitz and smirnov two-sample tests", *The Annals of Statistics*, pp. 697–717, 1979.

[49] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. CRC Press, 2003.

[50] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, "Early drift detection method", in *International Workshop on Knowledge Discovery from Data Streams*, 2006, pp. 77–86.

[51] S. H. Bach and M. A. Maloof, "Paired learners for concept drift", in *IEEE International Conference on Data Mining (ICDM)*, IEEE, 2008, pp. 23–32.

[52] P. Sobhani and H. Beigy, "New drift detection method for data streams", in *Adaptive and Intelligent Systems*, Springer, 2011, pp. 88–97.

[53] K. Nishida and K. Yamauchi, "Detecting concept drift using statistical testing", in *International Conference on Discovery Science*, Springer, 2007, pp. 264–269.

[54] R. S. Barros, D. R. Cabral, P. M. Gonçalves Jr., and S. G. Santos, "Rddm: Reactive drift detection method", *Expert Systems with Applications*, vol. 90, pp. 344–355, 2017.

[55] T. Escovedo, A. Koshiyama, A. A. da Cruz, and M. Vellasco, "Detecta: Abrupt concept drift detection in non-stationary environments", *Applied Soft Computing*, vol. 62, pp. 119–133, 2018.

[56] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2010, pp. 135–150.

[57] N. C. Oza, "Online bagging and boosting", in *IEEE International Conference on Systems, Man and Cybernetics*, Ieee, vol. 3, 2005, pp. 2340–2345.

[58] P. Domingos and G. Hulten, "Mining high speed data streams", in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2000, pp. 71–80.

[59] D. H. Wolpert, "The supervised learning no-free-lunch theorems", in *Soft Computing and Industry*, Springer, 2002, pp. 25–42.

[60] D. V. Oliveira, G. D. Cavalcanti, and R. Sabourin, "Online pruning of base classifiers for dynamic ensemble selection", *Pattern Recognition*, vol. 72, pp. 44–58, 2017.

[61] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning", in *Advances in Neural Information Processing Systems*, 1995, pp. 231–238.

[62] A. Fern and R. Givan, "Online ensemble learning: An empirical study", *Machine Learning*, vol. 53, no. 1-2, pp. 71–109, 2003.

[63] L. Rokach, "Ensemble-based classifiers", *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.

[64] H. M. Gomes, J. P. Barddal, and F. Enembreck, "Pairwise combination of classifiers for ensemble learning on data streams", in *ACM Symposium on Applied Computing*, ACM, 2015, pp. 941–946.

[65] B. Krawczyk and M. Woźniak, "One-class classifiers with incremental learning and forgetting for data streams with concept drift", *Soft Computing*, vol. 19, no. 12, pp. 3387–3400, 2015.

[66] B. Krawczyk and A. Cano, "Online ensemble learning with abstaining classifiers for drifting and noisy data streams", *Applied Soft Computing*, vol. 68, pp. 677–692, 2018.

[67] L. Pietruczuk, L. Rutkowski, M. Jaworski, and P. Duda, "How to adjust an ensemble size in stream data mining?", *Information Sciences*, vol. 381, pp. 46–54, 2017.

[68] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy", *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

[69] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting", in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2001, pp. 359–364.

[70] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[71] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams", in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2009, pp. 139–148.

[72] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams", in *International Symposium on Intelligent Data Analysis*, Springer, 2009, pp. 249–260.

[73] G. D. F. Morales and A. Bifet, "Samoa: Scalable advanced massive online analysis.", *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 149–153, 2015.

[74] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution", in *International Conference on Machine Learning (ICML)*, 2003, pp. 856–863.

[75] L. Rokach, "Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography", *Computational Statistics and Data Analysis*, vol. 53, no. 12, 4 bibrangedash 046–4072, 2009.

[76] S. G. Soares and R. Araújo, "An adaptive ensemble of on-line extreme learning machines with variable forgetting factor for dynamic system prediction", *Neurocomputing*, vol. 171, pp. 693–707, 2016.

[77] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms", *Machine Learning*, vol. 90, no. 3, pp. 317–346, 2013.

[78] G. U. Yule, "On the association of attributes in statistics: With illustrations from the material of the childhood society, 8c", *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 194, no. 252-261, pp. 257–319, 1900.

[79] P. H. Sneath, R. R. Sokal, *et al.*, *Numerical taxonomy. The principles and practice of numerical classification*. Taylor and Francis, 1973.

[80] T. K. Ho, "The random subspace method for constructing decision forests", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[81] D. B. Skalak *et al.*, "The sources of increased accuracy for two proposed boosting algorithms", in *American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*, vol. 1129, 1996, p. 1133.

[82] G. Giacinto and F. Roli, "Design of effective neural network ensembles for image classification purposes", *Image and Vision Computing*, vol. 19, no. 9-10, pp. 699–707, 2001.

[83] P. Cunningham and J. Carney, "Diversity versus quality in classification ensembles based on feature selection", in *European Conference on Machine Learning*, Springer, 2000, pp. 109–116.

[84] L. K. Hansen and P. Salamon, "Neural network ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[85] R. Kohavi, D. H. Wolpert, *et al.*, "Bias plus variance decomposition for zero-one loss functions", in *International Conference on Machine Learning (ICML)*, vol. 96, 1996, pp. 275–83.

[86] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization", *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000.

[87] J. L. Fleiss, B. Levin, and M. C. Paik, *Statistical methods for rates and proportions*. John Wiley and Sons, 2013.

[88] D. Partridge and W. Krzanowski, "Software diversity: Practical statistics for its measurement and exploitation", *Information and Software Technology*, vol. 39, no. 10, pp. 707–717, 1997.

[89] I. Žliobaitė, "Controlled permutations for testing adaptive learning models", *Knowledge and Information Systems*, vol. 39, no. 3, pp. 565–578, 2014.

[90] D. Brzezinski and J. Stefanowski, "Prequential auc for classifier evaluation and drift detection in evolving data streams", in *International Workshop on New Frontiers in Mining Complex Patterns*, Springer, 2014, pp. 87–101.

[91] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank, "Fast perceptron decision tree learning from evolving data streams", in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2010, pp. 299–310.

[92] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power", *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.

[93] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers", in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2015, pp. 59–68.

[94] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages", *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[95] J. Demšar, "Statistical comparisons of classifiers over multiple data sets", *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1–30, 2006.

[96] A. Bifet, G. Holmes, B. Pfahringer, J. Read, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: A real-time analytics open source framework", in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2011, pp. 617–620.

[97] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence", *Machine Learning*, vol. 98, no. 3, pp. 455–482, 2015.

[98] G. Hulten, L. Spencer, and P. Domingos, "Mining time changing data streams", in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2001, pp. 97–106.

[99] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, and L. Wan, "Heterogeneous ensemble for feature drifts in data streams", in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2012, pp. 1–12.

[100] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit multiflow: a multi output streaming framework", *CoRR*, vol. abs/-1807.04662, 2018. [Online]. Available: `https://github.com/scikit-multiflow/scikit-multiflow`.

[101] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions", *Sankhyā: The Indian Journal of Statistics (1933-1960)*, vol. 35, pp. 99–109, 1943.

[102] J. Wang, A. Belatreche, L. P. Maguire, and T. M. McGinnity, "Spiketemp: An enhanced rank-order-based learning approach for spiking neural networks with adaptive structure", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 1, pp. 30–43, 2017.

[103] J. Wang, A. Belatreche, L. Maguire, and T. M. Mcginnity, "An online supervised learning method for spiking neural networks with adaptive structure", *Neurocomputing*, vol. 144, pp. – 526–536, 2014.

[104] M. Kristan, D. Skočaj, and A. Leonardis, "Online kernel density estimation for interactive learning", *Image and Vision Computing*, vol. 28, no. 7, pp. 1106–1116, 2010.

[105] M. Kristan, A. Leonardis, and D. Skočaj, "Multivariate online kernel density estimation with gaussian kernels", *Pattern Recognition*, vol. 44, no. 10-11, pp. 2630–2642, 2011.

[106] M. S. Uddin, A. Kuh, Y. Weng, and M. Ilić, "Online bad data detection using kernel density estimation", in *IEEE Power Energy Society General Meeting*, 2015, pp. 1–5.

[107] T. K. Ho, "C4. 5 decision forests", in *International Conference on Pattern Recognition*, IEEE, vol. 1, 1998, pp. 545–549.

[108] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes", *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1994.

[109] L. Breiman, "Randomizing outputs to increase prediction accuracy", *Machine Learning*, vol. 40, no. 3, pp. 229–242, 2000.

[110] P. Melville and R. J. Mooney, "Creating diversity in ensembles using artificial data", *Information Fusion*, vol. 6, no. 1, pp. 99–111, 2005.

[111] G. Martínez-Muñoz and A. Suárez, "Switching class labels to generate classification ensembles", *Pattern Recognition*, vol. 38, no. 10, pp. 1483–1494, 2005.

[112] G. Martínez-Muñoz, A. Sánchez-Martínez, D. Hernández-Lobato, and A. Suárez, "Class-switching neural network ensembles", *Neurocomputing*, vol. 71, no. 13-15, pp. 2521–2528, 2008.

[113] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Self-adaptive windowing approach for handling complex concept drift", *Cognitive Computation*, vol. 7, no. 6, pp. 772–790, 2015.

[114] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii", *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[115] J. Knowles and D. Corne, "The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation", in *IEEE Congress on Evolutionary Computation*, IEEE, vol. 1, 1999, pp. 98–105.

[116] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm", *TIK-report*, vol. 103, 2001.

[117] S. M. Winkler, M. Affenzeller, G. Kronberger, M. Kommenda, B. Burlacu, and S. Wagner, "Sliding window symbolic regression for detecting changes of system dynamics", in *Genetic Programming Theory and Practice XII*, Springer, 2015, pp. 91–107.

[118] N. Wagner, Z. Michalewicz, M. Khouja, and R. R. McGregor, "Time series forecasting for dynamic environments: The dyfor genetic program model", *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 4, pp. 433–452, 2007.

[119] X.-S. Yang and M. Karamanoglu, "Swarm intelligence and bio inspired computation: An overview", in *Swarm intelligence and bio-inspired computation*, Elsevier, 2013, pp. 3–23.

[120] H. Wang, L. Jiao, and X. Yao, "Two_arch2: An improved two archive algorithm for many objective optimization", *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 524–541, 2015.

[121] K. Tumer and J. Ghosh, "Error correlation and error reduction in ensemble classifiers", *Connection science*, vol. 8, no. 3-4, pp. 385–404, 1996.

[122] J. L. Lobo, J. Del Ser, E. Villar-Rodriguez, M. N. Bilbao, and S. Salcedo-Sanz, "On the creation of diverse ensembles for non-stationary environments using bio-inspired heuristics", in *International Conference on Harmony Search Algorithm*, Springer, 2017, pp. 67–77.

[123] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Ensemble diversity measures and their application to thinning", *Information Fusion*, vol. 6, no. 1, pp. 49–62, 2005.

[124] R. M. Moraes and L. S. Machado, "Gaussian naive bayes for online training assessment in virtual reality-based simulators", *Mathware and Soft Computing*, vol. 16, no. 2, pp. 123–132, 2009.

[125] R. Marcos De Moraes and L. dos Santos Machado, "Online assessment in medical simulators based on virtual reality using fuzzy gaussian naive bayes.", *Journal of Multiple-Valued Logic and Soft Computing*, vol. 18, 2012.

[126] G. Ditzler, R. Polikar, and N. Chawla, "An incremental learning algorithm for non-stationary environments and class imbalance", in *International Conference on Pattern Recognition (ICPR)*, IEEE, 2010, pp. 2997–3000.

[127] J. Kok, L. F. Gonzalez, N. A. Kelson, and J. Periaux, "An fpga-based approach to multi-objective evolutionary algorithm for multi-disciplinary design optimisation", 2011.

[128] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification", in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2001, pp. 377–382.

[129] A. P. Dawid, V. G. Vovk, *et al.*, "Prequential probability: Principles and properties", *Bernoulli*, vol. 5, no. 1, pp. 125–162, 1999.

[130] S. Roberts, "Control chart tests based on geometric moving averages", *Technometrics*, vol. 1, no. 3, pp. 239–250, 1959.

[131] C. Alippi and M. Roveri, "An adaptive cusum-based test for signal change detection", in *IEEE International Symposium on Circuits and Systems (ICSAS)*, IEEE, 2006, 4–pp.

[132] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 20-02.

[133] S. Soltic and N. Kasabov, "Knowledge extraction from evolving spiking neural networks with rank order population coding", *International Journal of Neural Systems*, vol. 20, no. 06, pp. 437–445, 2010.

[134] S. Schliebs and N. Kasabov, "Evolving spiking neural network: A survey", *Evolving Systems*, vol. 4, no. 2, pp. 87–98, 2013.

[135] N. K. Kasabov, *Evolving connectionist systems: The knowledge engineering approach*. Springer Science and Business Media, 2007.

[136] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Adaptive learning procedure for a network of spiking neurons and visual pattern recognition", in *International Conference on Advanced Concepts for Intelligent Vision Systems*, Springer, 2006, pp. 1133–1142.

[137] S. Thorpe and J. Gautrais, "Rank order coding", in *Computational Neuroscience*, Springer, 1998, pp. 113–118.

[138] S. M. Bohte, J. N. Kok, and J. A. La Poutré, "Spikeprop: Backpropagation for networks of spiking neurons.", in *ESANN*, 2000, pp. 419–424.

[139] A. Belatreche, L. P. Maguire, and M. McGinnity, "Advances in design and application of spiking neural networks", *Soft Computing*, vol. 11, no. 3, pp. 239–248, 2007.

[140] F. Ponulak, "Resume-new supervised learning method for spiking neural networks", *Institute of Control and Information Engineering, Poznan University of Technology*, vol. 42, 2005.

[141] F. Ponulak, "Analysis of the resume learning process for spiking neural networks", *International Journal of Applied Mathematics and Computer Science*, vol. 18, no. 2, pp. 117–127, 2008.

[142] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting", *Neural Computation*, vol. 22, no. 2, pp. 467–510, 2010.

[143] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Evolving spiking neural networks for audiovisual information processing", *Neural Networks*, vol. 23, no. 7, pp. 819–835, 2010.

[144] J. Li and Y. Wang, "Prototype selection based on multi objective optimisation and partition strategy", *International Journal of Sensor Networks*, vol. 17, no. 3, pp. 163–176, 2015.

[145] L. Meena and V. S. Devi, "Prototype selection on large and streaming data", in *International Conference on Neural Information Processing*, Springer, 2015, pp. 671–679.

[146] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 86–100, 2012.

[147] W. Hu and Y. Tan, "Prototype generation using multiobjective particle swarm optimization for nearest neighbor classification", *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2719–2731, 2016.

[148] H. J. Escalante, M. Graff, and A. Morales-Reyes, "Pggp: Prototype generation via genetic programming", *Applied Soft Computing*, vol. 40, pp. 569–580, 2016.

[149] N. K. Kasabov, "Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data", *Neural Networks*, vol. 52, pp. 62–76, 2014.

[150] N. K. Kasabov, N. Scott, E. Tu, S. Marks, N. Sengupta, E. Capecci, M. Othman, M. Doborjeh, N. Murli, R. Hartono, *et al.*, "Design methodology and selected applications of evolving spatio-temporal data machines in the neucube neuromorphic framework", *Neural Networks*, vol. 78, pp. 1–14, 2016.

[151] N. Kasabov, K. Dhoble, N. Nuntalid, and G. Indiveri, "Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition", *Neural Networks*, vol. 41, pp. 188–201, 2013.

[152] S. J. Thorpe and J. Gautrais, "Rapid visual processing using spike asynchrony", in *Advances in Neural Information Processing Systems*, 1997, pp. 901–907.

[153] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error backpropagation in temporally encoded networks of spiking neurons", *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, 2002.

[154] S. Soltic, S. G. Wysoski, and N. K. Kasabov, "Evolving spiking neural networks for taste recognition", in *IEEE International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2008, pp. 2091–2097.

[155] F. Alnajjar, I. B. M. Zin, and K. Murase, "A spiking neural network with dynamic memory for a real autonomous mobile robot in dynamic environment", in *IEEE International Joint Conference on Neural Networks*, IEEE, 2008, pp. 2207–2213.

[156] I. Kononenko and M. Kukar, *Machine learning and data mining: Introduction to principles and algorithms*. Horwood Publishing, 2007.

[157] S. Garcia, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 417–435, 2012.

[158] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavalda, "Improving adaptive bagging methods for evolving data streams", in *Asian Conference on Machine Learning*, Springer, 2009, pp. 23–37.

[159] R. Chang, Z. Pei, and C. Zhang, "A modified editing k-nearest neighbor rule", *Journal of Computers*, vol. 6, no. 7, pp. 1493–1500, 2011.

[160] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data", *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, pp. 408–421, 1972.

[161] P. Hart, "The condensed nearest neighbor rule (corresp.)", *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515–516, 1968.

[162] I. Tomek, "An experiment with the edited nearest-neighbor rule", *IEEE Transactions on Systems, Man, and Cybernetics*, no. 6, pp. 448–452, 1976.

[163] I. Tomek, "Two modifications of cnn", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, pp. 769–772, 1976.

[164] J. Derrac, S. García, and F. Herrera, "Stratified prototype selection based on a steady-state memetic algorithm: A study of scalability", *Memetic Computing*, vol. 2, no. 3, pp. 183–199, 2010.

[165] H. A. Fayed, S. R. Hashem, and A. F. Atiya, "Self-generating prototypes for pattern classification", *Pattern Recognition*, vol. 40, no. 5, pp. 1498–1509, 2007.

[166] D. V. Oliveira, G. R. Magalhaes, G. D. Cavalcanti, and T. I. Ren, "Improved self-generating prototypes algorithm for imbalanced datasets", in *International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, vol. 1, 2012, pp. 904–909.

[167] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments", *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[168] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification", *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[169] L. Breiman, "Random forests", *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[170] I. Khamassi and M. Sayed-Mouchaweh, "Self-adaptive ensemble classifier for handling complex concept drift", in *CEUR Workshop*, vol. 1958, 2017.

[171] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. Morales Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds", *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2015.

[172] J. Gao, B. Ding, W. Fan, J. Han, and S. Y. Philip, "Classifying data streams with skewed class distributions and concept drifts", *IEEE Internet Computing*, vol. 12, no. 6, 2008.

[173] R. Pears, S. Sakthithasan, and Y. S. Koh, "Detecting concept change in dynamic data streams", *Machine Learning*, vol. 97, no. 3, pp. 259–293, 2014.

[174] T. A. O'Brien, K. Kashinath, N. R. Cavanaugh, W. D. Collins, and J. P. O'Brien, "A fast and objective multidimensional kernel density estimation method: Fastkde", *Computational Statistics and Data Analysis*, vol. 101, pp. 148–160, 2016.

[175] J. Gama, *Knowledge discovery from data streams*. CRC Press, 2010.

[176] A. Bouchachia, "Incremental learning with multi-level adaptation", *Neurocomputing*, vol. 74, no. 11, pp. 1785–1799, 2011.

[177] M. Pratama, W. Pedrycz, and G. I. Webb, "An incremental construction of deep neuro fuzzy system for continual learning of non-stationary data streams", *ArXiv e-prints*, 2018. arXiv: `1808.08517 [cs.AI]`.

[178] S. J. Pan, Q. Yang, *et al.*, "A survey on transfer learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[179] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge", *Nature*, vol. 550, no. 7676, p. 354, 2017.

[180] R. S. Sutton, A. G. Barto, *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.

[181] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, p. 436, 2015.

[182] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning", in *International Conference on Machine Learning (ICML)*, 2011, pp. 689–696.

[183] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[184] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2013, pp. 6645–6649.

[185] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets", in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[186] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review", *ArXiv preprint arXiv:1802.07569*, 2018.

[187] P. Domingos, *The master algorithm: How the quest for the ultimate learning machine will remake our world*, ser. Basic Books. Basic Books, 2015, ISBN: 9780465065707.