



AMANDA: DENSITY-BASED ADAPTIVE MODEL FOR NON-STATIONARY  
DATA UNDER EXTREME VERIFICATION LATENCY SCENARIOS

Raul Sena Ferreira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Geraldo Zimbrão da Silva  
Leandro Guimarães Marques  
Alvim

Rio de Janeiro  
Junho de 2018

AMANDA: DENSITY-BASED ADAPTIVE MODEL FOR NON-STATIONARY  
DATA UNDER EXTREME VERIFICATION LATENCY SCENARIOS

Raul Sena Ferreira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO  
ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE  
ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO  
COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO  
GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E  
COMPUTAÇÃO.

Examinada por:

  
Prof. Geraldo Zimbrão da Silva, D.Sc.

  
Prof. Leandro Guimarães Marques Alvim, D.Sc.

  
Prof. Alexandre de Assis Bento Lima, D.Sc.

  
Prof. Eduardo Soares Ogasawara, D.Sc.

RIO DE JANEIRO, RJ – BRASIL  
JUNHO DE 2018

Ferreira, Raul Sena

AMANDA: Density-based Adaptive Model for Non-stationary Data under Extreme Verification Latency Scenarios/Raul Sena Ferreira. – Rio de Janeiro: UFRJ/COPPE, 2018.

XIII, 69 p.: il.; 29, 7cm.

Orientadores: Geraldo Zimbrão da Silva

Leandro Guimarães Marques Alvim

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Bibliography: p. 60 – 69.

1. Semi-supervised learning. 2. Concept-drift. 3. Core support extraction. 4. Data streams. 5. Non-stationary environments. 6. Extreme verification latency. I. Silva, Geraldo Zimbrão da *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*Labor Vincit Aurum.  
Amor Vincit Omnia.*

# Acknowledgments

I would like to thank the mysterious power that lies inside me and us all, whose name I do not know (around the world, people call it God, consciousness, intuition, nature, randomness and so on), whose shape I never saw but everybody felt or one day will feel. Even you have all support of your friends or family, there will be a moment that you will feel alone and questioning yourself if what you are doing will lead you to some place. That power helped me going through the void, It made me see what I would achieve in the end. I would like to thank my wife for that patiently supported me in the uncertainty moments. I cannot forget to thank my grandparents Maria do Céu and José de Moura, for their love and for their kindness dedicated to me for many time. I thank my parents "Kim" and "Mary", that even far from me always encouraged me to pursue my goals.

My thanks to my advisor, professor Geraldo Zimbrão, for believe in my potential and for the good insights provided in this work. I also thank my advisor Leandro Alvim, for his invaluable dedication in this work, my life doing this thesis would be very worst without his help and counseling. I would like to thank the presence of professor Alexandre Assis, which I have great esteem and a friendship since the first years of my studies in UFRJ. I would like to thank the professor Eduardo Ogasawara for his readiness to participate in the examining board of this work. My gratitude for professor Filipe Braida for help me enter in the scientific path and for our long friendship. I would like to thank my former undergraduate advisor, professor Carlos Eduardo, for his clever insights, helping me to try good solutions for some problems in this work and for the friendship that we built in all those years.

At last but not least, I want to thank my friends Kleyton, Ygor, Alexsander, Julio, Miguel and all other friends from UFRRJ and UFRJ, that helped me, in some way, revising this work or just chatting and laughing (some times complaining) about this hard thing, called scientific research. The reserved space in this page is not enough to honor everybody but I definitely will not forget those that stayed beside me in this journey.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

AMANDA: MODELO ADAPTATIVO BASEADO EM DENSIDADES PARA  
DADOS NÃO-ESTACIONÁRIOS EM CENÁRIOS DE LATÊNCIA DE  
VERIFICAÇÃO EXTREMA

Raul Sena Ferreira

Junho/2018

Orientadores: Geraldo Zimbrão da Silva

Leandro Guimarães Marques Alvim

Programa: Engenharia de Sistemas e Computação

*Concept-drift* gradual refere-se à mudança suave e gradual na distribuição dos dados conforme o tempo passa. Este problema causa obsolescência no modelo de aprendizado e queda na qualidade das previsões. Além disso, existe um complicador durante o processamento dos dados: a *latência de verificação extrema* (LVE) para se verificar os rótulos. Métodos do estado da arte propõem uma adaptação do modelo supervisionado usando uma abordagem de estimação de importância baseado em mínimos quadrados ou usando uma abordagem semi-supervisionada em conjunto com a extração de instâncias centrais, na sigla em inglês (CSE). Entretanto, estes métodos não tratam adequadamente os problemas mencionados devido ao fato de requererem alto tempo computacional para processar grandes volumes de dados, falta de correta seleção das instâncias que representam a mudança da distribuição, ou ainda por demandarem o ajuste de grande quantidade de parâmetros. Portanto, propomos um *modelo adaptativo baseado em densidades para dados não-estacionários* (AMANDA), que tem como base um classificador semi-supervisionado e um método CSE baseado em densidade. AMANDA tem duas variações: percentual de corte fixo (AMANDA-FCP); e percentual de corte dinâmico (AMANDA-DCP). Nossos resultados indicam que as duas variações da proposta superam o estado da arte em quase todas as bases de dados sintéticas e reais em até 27,98% em relação ao erro médio. Concluímos que a aplicação do método AMANDA-FCP faz com que a classificação melhore mesmo quando há uma pequena porção inicial de dados rotulados. Mais ainda, os classificadores semi-supervisionados são melhorados quando trabalham em conjunto com nossos métodos de CSE, estático ou dinâmico.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AMANDA: DENSITY-BASED ADAPTIVE MODEL FOR NON-STATIONARY  
DATA UNDER EXTREME VERIFICATION LATENCY SCENARIOS

Raul Sena Ferreira

June/2018

Advisors: Geraldo Zimbrão da Silva

Leandro Guimarães Marques Alvim

Department: Systems Engineering and Computer Science

Gradual concept-drift refers to a smooth and gradual change in the relations between input and output data in the underlying distribution over time. The problem generates a model obsolescence and consequently a quality decrease in predictions. Besides, there is a challenging task during the stream: The extreme verification latency (EVL) to verify the labels. For batch scenarios, state-of-the-art methods propose an adaptation of a supervised model by using an unconstrained least squares importance fitting (uLSIF) algorithm or a semi-supervised approach along with a core support extraction (CSE) method. However, these methods do not properly tackle the mentioned problems due to their high computational time for large data volumes, lack in representing the right samples of the drift or even for having several parameters for tuning. Therefore, we propose a density-based adaptive model for non-stationary data (AMANDA), which uses a semi-supervised classifier along with a CSE method. AMANDA has two variations: AMANDA with a fixed cutting percentage (AMANDA-FCP); and AMANDA with a dynamic cutting percentage (AMANDA-DCP). Our results indicate that the two variations of AMANDA outperform the state-of-the-art methods for almost all synthetic datasets and real ones with an improvement up to 27.98% regarding the average error. We have found that the use of AMANDA-FCP improved the results for a gradual concept-drift even with a small size of initial labeled data. Moreover, our results indicate that SSL classifiers are improved when they work along with our static or dynamic CSE methods. Therefore, we emphasize the importance of research directions based on this approach.

# Contents

<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xii</b>
<b>List of Algorithms</b>	<b>xiii</b>
<b>List of Abbreviations</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Objectives . . . . .	2
1.2 Summary of Results . . . . .	3
1.3 Contributions . . . . .	3
1.4 Document Structure . . . . .	4
<b>2 Concept-Drift Fundamentals</b>	<b>5</b>
2.1 Adaptive Learning . . . . .	8
2.1.1 Machine Learning in Non-stationary Environments . . . . .	9
2.1.2 Concept-Drift Types . . . . .	10
2.1.3 Extreme Verification Latency . . . . .	13
2.1.4 Density-based Methods for Core Support Extraction . . . . .	13
2.2 Related Work . . . . .	15
2.2.1 COMPOSE-GMM . . . . .	18
2.2.2 LEVELiw . . . . .	21
<b>3 AMANDA - Density-based Adaptive Model for Non-stationary   DAta</b>	<b>24</b>
3.1 Amanda Framework . . . . .	25
3.2 Core Support Extraction . . . . .	26
3.3 AMANDA-FCP . . . . .	26
3.4 AMANDA-DCP . . . . .	27
<b>4 Empirical Evaluation</b>	<b>31</b>
4.1 Experiment Objectives . . . . .	31

4.2	Datasets . . . . .	31
4.2.1	Synthetic Datasets . . . . .	31
4.2.2	Real Datasets . . . . .	33
4.3	Methodology and Empirical Setup . . . . .	34
4.3.1	Simulation Scenarios . . . . .	35
4.3.2	Validation and Metrics . . . . .	35
4.3.3	Classifiers and Baselines . . . . .	35
4.4	Results . . . . .	36
4.4.1	Density-based Algorithms Choice for CSE . . . . .	36
4.4.2	AMANDA Parameter's Influence . . . . .	37
4.4.3	Artificial Dataset - 2CDT . . . . .	39
4.4.4	Artificial Dataset - 1CSURR . . . . .	42
4.4.5	Artificial Dataset - 4CRE_V1 . . . . .	45
4.4.6	Real Dataset - Keystroke . . . . .	48
4.4.7	Real Dataset - NOAA . . . . .	50
4.4.8	Real Dataset - Electricity . . . . .	51
4.4.9	Overall Results . . . . .	54
<b>5</b>	<b>Conclusions</b>	<b>57</b>
5.1	Proposal . . . . .	57
5.2	Summary of Results . . . . .	57
5.3	Contributions . . . . .	58
5.4	Limitations and Research Directions . . . . .	58
	<b>Bibliography</b>	<b>60</b>

# List of Figures

1.1	Summary of results. . . . .	3
2.1	Concept-drift applications (KADWE e SURYAWANSHI, 2015). . . . .	7
2.2	Adaptive strategies for machine learning (ŽLIOBAITĚ <i>et al.</i> , 2016). . . . .	9
2.3	Stationary versus non-stationary learning (ŽLIOBAITĚ <i>et al.</i> , 2016). . . . .	10
2.4	Progress of a single class experiencing: a) Translational b) Rotational c) Volumetric drift (DYER <i>et al.</i> , 2014). . . . .	10
2.5	Patterns of changes over time (GAMA <i>et al.</i> , 2014a). . . . .	11
2.6	Original data, virtual and real drifts, respectively (KHAMASSI <i>et al.</i> , 2018). . . . .	12
2.7	COMPOSE - Framework flow (DYER <i>et al.</i> , 2014). . . . .	19
2.8	Evolution of the effects of decreasing the $\alpha$ parameter (DYER <i>et al.</i> , 2014). . . . .	19
3.1	AMANDA framework. . . . .	25
4.1	KDE performance versus GMM performance in CSurr dataset. . . . .	37
4.2	Bi-dimensional two classes with diagonal transitioning. Data distribu- tion evolution through the time (yellow arrow). . . . .	39
4.3	Bi-dimensional two classes with diagonal transitioning . . . . .	40
4.4	Bi-dimensional two classes with diagonal transitioning . . . . .	41
4.5	Error reduction over static classifier - Bi-dimensional two classes with diagonal transitioning. . . . .	41
4.6	1CSURR data distribution evolution through the time (yellow arrow). . . . .	42
4.7	AMANDA-FCP Core Support Extraction Performance in 1CSURR. . . . .	43
4.8	AMANDA-DCP Core Support Extraction Performance in 1CSURR. . . . .	43
4.9	COMPOSE-GMM Core Support Extraction Performance in 1CSURR. . . . .	43
4.10	One class surrounding another (1CSURR) . . . . .	44
4.11	One class surrounding another . . . . .	44
4.12	Error reduction over static model - One class surrounding another. . . . .	45
4.13	bi-dimensional four class rotating and expanding. Data distribution evolution through the time (yellow arrow). . . . .	45

4.14	Four class rotating and expanding (Version 1)	46
4.15	Four class rotating and expanding (Version 1)	47
4.16	Error reduction over static model - Four class rotating and expanding (Version 1).	47
4.17	Keystroke results	48
4.18	Keystroke results	49
4.19	Error reduction over static model - Keystroke dataset.	49
4.20	NOAA results	50
4.21	NOAA results	51
4.22	Error reduction over static model - NOAA dataset.	52
4.23	Electricity dataset results	52
4.24	Electricity dataset results	53
4.25	Error reduction over static model - Electricity dataset.	53

# List of Tables

4.1	Datasets. . . . .	32
4.2	Classifiers accuracy. . . . .	38
4.3	Average accuracy for the keystroke dataset . . . . .	38
4.4	Average error results. . . . .	54
4.5	Macro-F1 results. . . . .	55
4.6	Processing time results. . . . .	56

# List of Algorithms

1	COMPOSE-GMM . . . . .	20
2	IWLSPC . . . . .	22
3	LEVELiw . . . . .	22
4	AMANDA - FCP . . . . .	27
5	AMANDA - DCP . . . . .	28
6	Cutting percentage calculation . . . . .	29

# Chapter 1

## Introduction

Scenarios where data are not generated by law physics are fated to suffer changes in their statistical distribution (WIDMER e KUBAT, 1996). These complex environments are denominated non-stationary environments (MORRISON e DE JONG, 1999). Non-stationary environments are highly probable that class-distributions change over time (GAMA *et al.*, 2004) due to contextual and temporal reasons (GAMA *et al.*, 2014a). These changes create an important problem denominated concept-drift (WIDMER e KUBAT, 1996).

Concept-drift refers to the change in the relations between input and output data in the underlying distribution over time (GAMA *et al.*, 2004). For example, human preferences (DING *et al.*, 2006) and robotic tasks such as autonomous navigation (MARRS *et al.*, 2010) that changes through time. The problem generates a model obsolescence and quality decreasing in the predictions. Formally, the concept drift problem is defined by GAMA *et al.* (2014a) as:

$$\exists X : P_{t_0}(X, y) \neq P_{t_1}(X, y), \quad (1.1)$$

where  $P_{t_0}$  denotes the joint distribution at time  $t_0$  between the set of input variables  $X$  and the target variable  $y$ . Thus, the prior probabilities of classes  $P(y)$  and the class conditional probabilities  $P(X|y)$  may change resulting in changes in the posterior probabilities of the classes  $P(y|X)$ , affecting predictions (GAO *et al.*, 2007, KELLY *et al.*, 1999).

Beyond the definition, we must characterize the concept-drift problem as follows: type, behavior and data flow. Regarding its type, there are two distinct variations: gradual and abrupt. The first type refers to a smooth and gradual change in a data distribution over time, that is, the probability of an instance being in concept A declines linearly as the probability of an instance being in concept B increases until A is completely replaced by B (STANLEY, 2003). The second type refers to a sudden change, that is, the concept shifts instantaneously between A and B STANLEY

(2003).

Regarding its behaviour, we can divide it by rotational, translational and volumetric (DYER *et al.*, 2014). The rotational behaviour occurs when the data rotates around its center. The translational behaviour occurs when the data shifts in one direction. Finally, the volumetric behaviour occurs when the distribution expand or contract itself around its center.

In relation to data flow, the data can come in the form of stream or batch. Depending on its type, the algorithm must be prepared for it. Worth to mention that labeling data is an expensive task in stream scenarios. For example, in robotics, the acquisition of the labels is a challenging task due to the latency to verify the labels during the stream (XU *et al.*, 2017). This problem is denominated extreme verification latency (EVL) (KREMPL *et al.*, 2014). Therefore, due to its importance, we also investigate our method under the EVL condition.

In batch scenarios, DYER *et al.* (2014) and CAPO *et al.* (2014) proposed a semi-supervised learning (SSL) approach along with a core support extraction (CSE) method to deal with gradual drift and EVL. Their methods extract samples from a core support region and discard the remaining instances. Hence, the methods use these samples as training instances for the classifier in the next batches of unlabeled data. The main disadvantage of these methods is the expensive computational cost of the CSE methods chosen for these works. Besides, the CSE method of the second work (CAPO *et al.*, 2014) seems to be suitable for Gaussian distributions. However, Gaussian distributions are not common in real problems. Thus, further research to apply new CSE approaches is very important.

To address the same set of problems in a stream scenario, SOUZA *et al.* (2015b) and SOUZA *et al.* (2015a) proposed a semi-supervised approach guided by a clustering algorithm to classify an instance. The main disadvantage of these methods is the high processing time for datasets with more than 100,000 instances. Another weakness is the variability of its results due to the change of a parameter that controls the number of clusters.

## 1.1 Objectives

Taking into account the issues of the state-of-the-art methods, we propose a SSL classifier with a density-based CSE method that helps to adapt itself to the drift. This density-based CSE is responsible for weighting and filtering the instances for the next SSL algorithm iterations. Hence, instances containing old concepts are discarded while the remaining instances, that are able to represent the new concepts, are kept for the next subsequent steps. This approach is able to deal with the gradual drift in EVL scenarios and is faster than the state-of-the-art methods since it removes

up to 90% of instances needed for the training step in some datasets.

Therefore, we propose a density-based adaptive model for non-stationary data (AMANDA) under EVL scenarios, with two variations: fixed cutting percentage (AMANDA-FCP) and dynamic cutting percentage (AMANDA-DCP). We evaluate these two variations of AMANDA at seventeen artificial datasets and three real ones. We compare the proposal to the static, incremental and sliding window classifiers, and two other state-of-the-art methods as well.

## 1.2 Summary of Results

Our results are illustrated in Figure 1.1 and indicate that AMANDA-FCP outperforms the state-of-the-art method (LEVELiw) regarding the average error in the real datasets: Electricity (ELEC2), NOAA and Keystroke. However, AMANDA-DCP is surpassed in the Electricity dataset. Negative values mean an increase in the error. Additionally, AMANDA-FCP outperforms the LEVELiw and COMPOSE-GMM, another state-of-the-art method, in fifteen of seventeen artificial datasets.

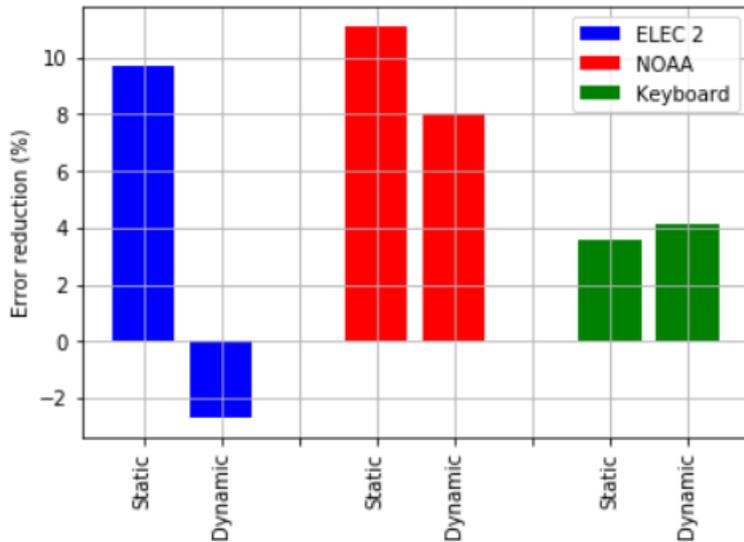


Figure 1.1: Summary of results.

## 1.3 Contributions

Our main contributions are:

- *Two methods for gradual concept-drift*

Two methods based on our adaptive framework AMANDA: a fixed cutting percentage (AMANDA-FCP); and an alternative dynamic cutting percentage (AMANDA-DCP) that diminish the only free parameter of the former method

- *A density-based CSE method for core support regions*  
A density approach for weighting and filtering samples that best represent the concepts in the data distribution
- *A method that learns under extreme verification latency scenarios*  
A semi-supervised approach that presents high accuracy for unlabeled data under EVL scenarios
- *A Comparison and analysis*  
A comparison between the proposal and the most relevant methods for the concept-drift problem

## 1.4 Document Structure

This work is organized as follows: in Chapter 2, we introduce the concept-drift fundamentals and related works regarding machine learning for dynamic environments. In Chapter 3, we present a semi-supervised approach for gradual concept-drift on EVL scenarios. Next, in Chapter 4, we describe our methodology for conducting the experiments, datasets and our results. Finally, in Chapter 5, we present our final considerations, work's limitations, and research directions.

# Chapter 2

## Concept-Drift Fundamentals

Often in data mining and machine learning problems, data is collected and processed offline. Hence, the model is built and refined with historical data and posteriorly applied to new data at online environment. However, it is usual that digital data are generated as streams which may present many challenges such as size, high rate arrival, data evolution over time (KHAMASSI *et al.*, 2018) and others.

In online environments, data processing needs to be real time, and preferentially has to be computational inexpensive since the high volume of data makes the strategy of processing all information in memory computationally infeasible (L'HEUREUX *et al.*, 2017). Online environments are often non-stationary (HAYKIN e LI, 1995). This condition is referred as an environment with concept-drift problem (TSYMBAL, 2004), also denominated as covariate shift (MORENO-TORRES *et al.*, 2012). Hence, the input data characteristics or the relation between the input data and the target variable may change, affecting the prediction.

In this environment, there is a necessity to build methods for extracting patterns from continuous batches of data. These methods are denominated incremental (online) learning algorithms. The incremental assumption is that upon receiving a new instance, it is much less expensive to update an existing model than to build a new one. On the other hand, as indicated by DOMINGOS e HULTEN (2000), incremental algorithms have several shortcomings such as high sensitivity to the order of training examples, and longer training times than the non-incremental (batch) methods. Pure incremental methods consider every new instance, which may be impractical in environments where transactions arrive at the rate of thousands per second. Thus, it is necessary to use adaptive algorithms, that is considered an extension of incremental-learning algorithms. Adaptive algorithms adapts and evolve according time, working properly in non-stationary environments.

According to ŽLIOBAITĖ *et al.* (2015), static machine-learning models assume that data is independent and identically distributed (iid). Identical distribution means that the joint probability  $P_t(X, y_i)$  of an observation  $X$  and its label  $y_i$  is

the same at any time  $t$ , that is,  $P_{t_0}(X, y_i) = P_{t_1}(X, y_i)$ . Independent distribution means that the probability of a label does not depend on what was observed earlier, that is,  $P(y_t) = P(y_t|y_{t-1})$ . However, non-stationary environments exhibit temporal dependence and data is not iid (GAMA *et al.*, 2014a, ŽLIOBAITĖ *et al.*, 2015), that is:

$$\exists X : P_{t_0}(X, y) \neq P_{t_1}(X, y), \quad (2.1)$$

ŽLIOBAITĖ *et al.* (2015) indicated that a naive classifier, considering the temporal dependence, can outperform several state-of-the-art classifiers on non-stationary environments.

Non-stationary data are found in several real problems, specially when there is a necessity for mining high-speed streams of data (DOMINGOS e HULTEN, 2000), where a drift occurs in the distribution due to the large amount of incoming data or seasonality factors. Intrusion detection in computers and computer networks (LANE e BRODLEY, 1998) is an example of non-stationary data. Here, the user’s behaviors and tasks change with time and the anomaly detection agent must be capable of adapting to these changes while still recognizing hostile actions and take care to not adapting to them. Another example is the e-mail classification task (CARMONA-CEJUDO *et al.*, 2011), where the criteria to classify emails within a folder may change over time. Traffic management (MOREIRA, 2008) also suffers influence of drifts in the distribution. Here a travel planning for a long-term prediction becomes challenging due to the drift that occurs in distribution of buses and their time travels through the time.

Non-stationary data also is produced from other problems such as activity recognition (ZHOU *et al.*, 2008), where drift occurs in data provided by humans in interactive applications. Sentiment classification in user’s opinions in Twitter data (BIFET e FRANK, 2010). Production quality control (PECHENIZKIY *et al.*, 2010), where control systems used in circulating fluidized bed (CFP) processes may fail to compensate the fluctuations due to fuel inhomogeneity. The author argues that these drifts makes the whole plant to suffer from dynamics, reducing efficiency and the lifetime of process components.

Concept-drift is also present in telecommunication monitoring (PAWLING *et al.*, 2007), where the task is to identify anomalous events in streaming cell phone data due to the way in which people use the services provided by a mobile communication network over time. Besides, concept-drift is present in problems such as controlling robots (PROCOPIO *et al.*, 2009) and controlling vehicles (THRUN *et al.*, 2006), where the navigation task requires identifying safe and crossable paths. This task is important to allow the robots to progress toward a goal while avoiding obstacles even occurring change in the patterns of the paths and obstacles.

Dynamic environments are also present in intelligent appliances problems (RASHIDI e COOK, 2009), where an environment has to adapt for the user behaviour through time. Computer games where the game adapts to player actions and increase its efficiency against the player (CHARLES *et al.*, 2005). Flight simulators (HARRIES *et al.*, 1998), where hidden changes of context are extracted from a flight simulator control with dynamic flight patterns. Change-detection in wireless sensor networks (ASL *et al.*, 2016) since the change in the distribution of sensors is quickly detected, reducing propagation time of environmental changes to sensors.

Many other important fields are illustrated in Figure 2.1. Thus, aiming to explain the importance of learning under non-stationary environments, in this chapter we describe common problems that occurs in such environments and common challenges that learning algorithms needs to face in order to perform learning tasks properly in non-stationary environments.

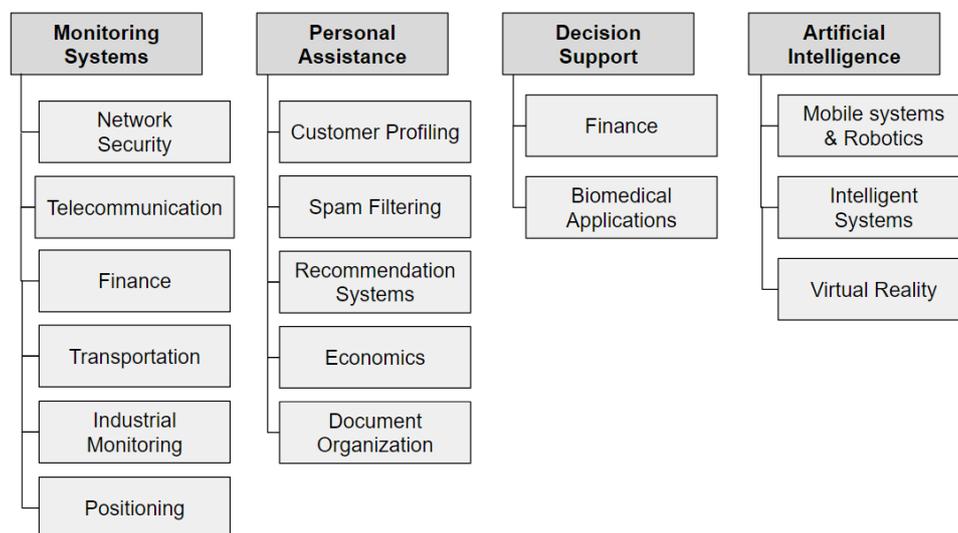


Figure 2.1: Concept-drift applications (KADWE e SURYAWANSHI, 2015).

There are many different strategies to deal with non-stationary environments in industry and academia (GAMA *et al.*, 2014b). The different types of concepts on these real scenarios present many challenges such as model obsolescence, necessity to build adaptive models and real time analysis (ŽLIJBAITĖ *et al.*, 2016). Worth mentioning that even though the temporal factor is present in the dataset, when we deal with concept-drift we are not necessarily dealing with temporal series. Temporal series may be non-stationary but not all non-stationary problems are temporal series. For example, if the prediction is conditioned on history, the problem is a time-series prediction and solutions to this problem are related to feeding the prediction history to the model as input. Otherwise, when there is a necessity to update the model then it is generally related to non-stationarity. Therefore, time-series prediction and prediction in non-stationary distributions are two different problems.

However, MUTHUKRISHNAN *et al.* (2007) and BIFET *et al.* (2013) suggest to inherit concepts and methods from time-series analysis to drift detection when there are temporal dependencies. For example, analyzing instances through a time window can be effective to exploit temporal dependence.

## 2.1 Adaptive Learning

A few non-stationary problems may be solved just retraining the learning model with the most recent data and determining according to performance if a model is inaccurate or not. However, for several types of problems that need a fast response and contain constant changes in distribution this approach is inaccurate. For instance, computational interfaces that collect brain signals and perform analysis in different human cerebral patterns. Therefore, learning algorithms often need to operate in dynamic environments. These environments change unexpectedly. One desirable property of these algorithms is their ability for incorporating new data. If the data that generates the process is not strictly stationary, and it applies to most of the real world applications, the underlying concept may change over time (GAMA *et al.*, 2014a). For example, interests of a user reading news.

Adaptive learning algorithms adapts to data in an iterative way with partial labeled data, updating itself when there are indications that the model is obsolete. Few adaptive models are incremental algorithms with partial memory (MALOOF e MICHALSKI, 2004) and assume that data comes in real time, considering this stream infinite and for this reason, it is normal that data change according time. The model adjustment is performed through triggering methods or evolving according to the time. There are strategies suitable for these two types of model adaptation (BOSE *et al.*, 2011). Figure 2.2 illustrates these strategies.

Inside these adaptive strategies there are recent efforts to detect when a drift will occur in data distribution. For example, a framework for detecting changes in multidimensional data using principal component analysis (PCA) (WOLD *et al.*, 1987), which is applied for projecting data into a lower dimensional space, facilitating density estimation and change-score calculations (QAHTAN *et al.*, 2015). However, in applications where drift is continual, these may be of limited use, as they should always flag the drift as present.

In such cases, rather than simply flagging whether drift is occurring or not, it may be more useful to generate a detailed description of the nature and form of the drift, a concept-drift map (WEBB *et al.*, 2017). WEBB *et al.* (2016) proposed four quantitative measures of concept-drift including the key measure drift magnitude which measures the distance between two concepts  $P_t(X, Y)$  and  $P_u(X, Y)$ . The author argues that any measure of distance between distributions could be employed

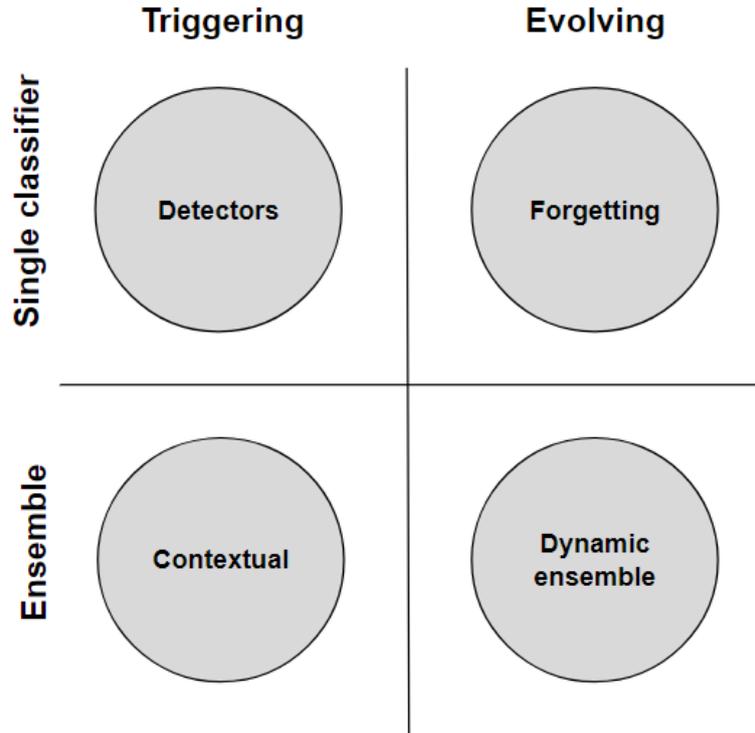


Figure 2.2: Adaptive strategies for machine learning (ŽLIJBAITĖ *et al.*, 2016).

and the chosen one was the Hellinger Distance (HOENS *et al.*, 2011).

### 2.1.1 Machine Learning in Non-stationary Environments

Machine learning in non-stationary environments increased its popularity due to the temporal information present in many real problems. Besides, up to 90's there were few mature solutions and systems collecting data for long periods of time. Thus, in the course of time, we fed these datasets and now, it contains drifts about our preferences, behaviors and other human characteristics collected over time. For instance, with the fast growth of users accessing virtual stores and buying through the internet, this trend made the problem of concept-drift easier to identify (DING *et al.*, 2006).

As previously explained, in dynamic environments a model learnt from the original data may become inaccurate over time since the fundamental processes generating most real-time data may change over years, months and even seconds (COHEN *et al.*, 2008). These changes may come from non-deterministic processes generated from humans, robots or sensors. For example, human preferences that changes through time or robotic tasks such as autonomous navigation. Beyond the fast model obsolescence, non-stationary environments are characterized by the delay to obtain the real labels. For example, in robotics, the acquisition of the real labels are an expensive and challenging task (XU *et al.*, 2017).

GAMA *et al.* (2014b) consider that at time  $t + 1$ , the previous label  $y_t$  of the sample  $(x_t, y_t)$  is available. However, this is a weak assumption and it is not applicable in several real problems. For instance, in sensors applications where exists failures in sensor readings; in robotics where the labels are not present or are outdated due to a new environment explored by a robot (MARRS *et al.*, 2010). This problem is denominated as extreme verification latency (EVL) and was recently pointed out as one of eight open challenges in data stream mining (KREMPL *et al.*, 2014). Worth to mention that a learning model needs to be differently addressed in stationary and non-stationary environments as illustrated in Figure 2.3.

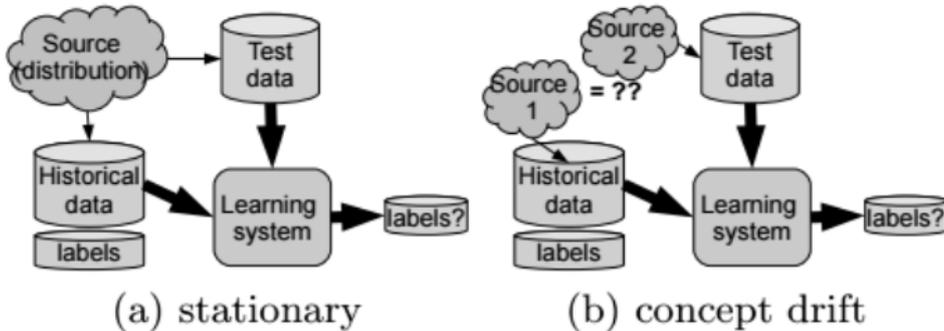


Figure 2.3: Stationary versus non-stationary learning (ŽLIOBAITĖ *et al.*, 2016).

## 2.1.2 Concept-Drift Types

A natural outcome of the gradual drift assumption is that class distributions overlap at subsequent time steps. As long as drift is limited, the core region of each class data distribution will have the most overlap with upcoming data, regardless of drift type (DYER *et al.*, 2014). Figure 2.4 illustrates three different types of drift: rotational, translational, and volumetric.

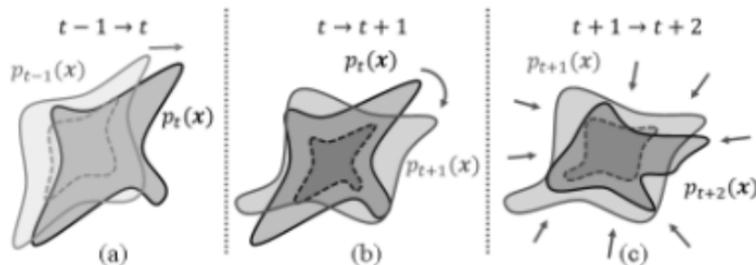


Figure 2.4: Progress of a single class experiencing: a) Translational b) Rotational c) Volumetric drift (DYER *et al.*, 2014).

The Figure 2.4 shows that the compacted core region (outlined) has the most overlap with the drifted distribution (dashed line). Besides, data distribution changes

in five different ways (GAMA *et al.*, 2014a) such as illustrated in Figure 2.5: abrupt, incremental, gradual, recurrent and due to outliers reasons.

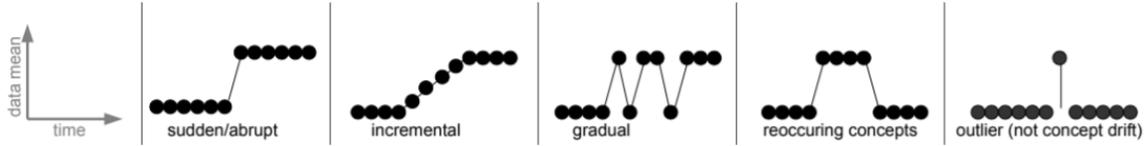


Figure 2.5: Patterns of changes over time (GAMA *et al.*, 2014a).

Recurrent concepts are previously active concepts that may reappear after a time. As stated by MINKU *et al.* (2010), recurrent drifts can have cyclic or acyclic behavior. The cyclic behaviour occur according to a periodicity or due to a seasonal trend. For instance, in electricity market, the prices may increase in winter due to the increase of demand, then return to previous price in the other season. The acyclic behaviour may not be periodic and is not clear when the concept may reappear. For instance, the fuel prices may suddenly increase due to the increase of petrol prices then return to previous price when petrol prices decrease.

Active drift is an abrupt drift in the distribution and may be detected by an update mechanism. The mechanism detect this change and apply a model correction routine since the distribution drastically changed and the error increased. For instance, when a sensor or group of sensors stop working. On the other hand, the passive drift is harder to perceive than active drift due to the gradual or incremental changes in the distribution.

These changes slightly decrease the model accuracy and when the problem is perceived the model is already inaccurate. For example, sensor measurements may present errors due to thermal or time effect conditions. A general observation is that, while active approaches are quite effective in detecting abrupt drift, passive approaches are efficient at overcoming gradual drift (WANG *et al.*, 2018). Besides, drifts can be permanent since the change is not limited according time. After a certain period of time the drift may disappear, becoming recurrent. Beyond the drifts in distribution, there are two types of drifts that a data can suffer in the concept-drift problem: real concept-drift and virtual concept-drift.

## Virtual Drift

Virtual drift refers to a change in distribution of the incoming data  $p(X)$  and does not affects the posterior probability  $p(y|X)$  (LAZARESCU *et al.*, 2004). This drift originally has been defined to occur due to incomplete data representation rather than change in concepts (WIDMER e KUBAT, 1993). It also corresponds to change in distribution that leads to changes in the decision boundary (TSYMBAL, 2004). Moreover, it does not affect the target concept (DELANY *et al.*, 2005). The term

virtual drift has been also referred to as sampling-shift (SALGANICOFF, 1997), temporary drift (LAZARESCU *et al.*, 2004) and feature change (GAO *et al.*, 2007). For example, it is considered as virtual drift when there is class imbalance but this not affect the decision boundaries.

## Real Drift

Real concept-drift refers to changes in posterior probability  $p(y|X)$  and can happen either with or without change in  $p(X)$ . Real concept-drift also is referred as concept-shift (SALGANICOFF, 1997) or conditional change (GAO *et al.*, 2007). Techniques that handle real concept-drift typically rely on feedback from the predictive performance while virtual concept-drift can operate without such feedback. The term *real drift* does not mean that other types of drift are not concept-drifts. Figure 2.6 illustrates the main differences between the drifts. For example, it is considered as real drift when there are prior class evolution, affecting the decision boundaries (DITZLER *et al.*, 2015).

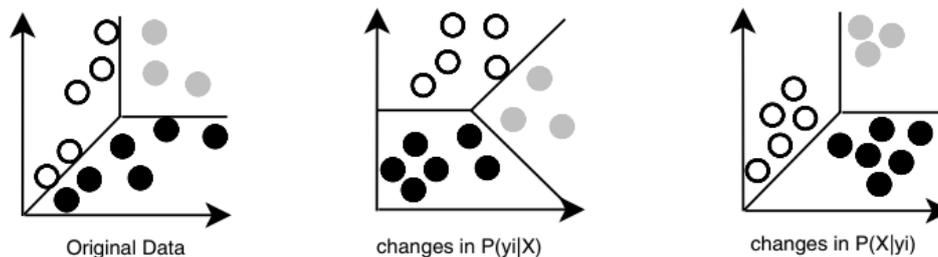


Figure 2.6: Original data, virtual and real drifts, respectively (KHAMASSI *et al.*, 2018).

Beyond these two types of concept-drift, there is the class prior concept-drift (KHAMASSI *et al.*, 2015). This is considered as a distinct drift type where the main challenge is to precise that the class prior distribution has changed. For example, when users have to rate a movie that they initially enjoyed for their special effects. After a long period of time, they may no longer enjoy the movie as their special effects become outdated. This is denominated real concept-drift due to a change in the user preference.

On the other hand, when users enjoyed a particular movie, and after a certain period of time, their preferences changed but they are still enjoying the movie. This evolution is denominated virtual concept-drift since it does not affect their preference. Finally, when the users may no longer be interested by a particular movie because of the emergence of new TV series, this can considered as class prior concept-drift.

### 2.1.3 Extreme Verification Latency

Often, we presume that classification algorithms always will receive labeled data from the same set of classes and this presumption not always is true. The incoming data right after the initial labeled data may not contain labels for a long period of time or even for the rest of the processing. Thus, that kind of environment demand a high effort to obtain new labels in the classification phase. This time between classification and label availability is denominated as verification latency (MARRS *et al.*, 2010) and the scenario with high time delay or label unavailability is denominated as extreme verification latency (CHAO, 2015).

The initial labeled data, generally 5% of the data, are necessary to define the problem as classification, also the number of classes and their initial disposition in the feature space. A simple explanation would be an autonomous robot previously trained inside a specific environment and sent to explore an unknown environment without external help or human supervising. This robot needs to adapt itself to the scenario changes without have the actual label of the incoming data. In this scenario, retraining the learning model while it explores new environments is important for robotic field (CHAO, 2015).

Environments with such extreme conditions are rapidly growing due to massive automated and autonomous acquisition of data. Creating a labeled database for this scenario is difficult and expensive. For instance, robots, drones and autonomous vehicles encountering surrounding environment. This change happens at a pace too fast for a human to verify all actions. A concrete application that requires extreme verification latency in non-stationary environments: consider a behavioral biometric application where users are recognized by their typing profile (ARAÚJO *et al.*, 2005). In this security system, all users type the same password and should be recognized by their typing pace. Such a characteristic evolves over time, the system need to constantly adapt to the new behavior of each user without any supervision.

### 2.1.4 Density-based Methods for Core Support Extraction

Aiming to detect the change in the distribution through the time, an usual approach is to extract the most representative instances from the core region of the new distribution. For this purpose, CSE methods may be applied along with semi-supervised classifiers CAPO *et al.* (2014). CSE methods attempt to accurately identify which instances lies in the core region of the existing class distributions. However, a common problem of these methods is to decide how many instances to store and what region of the decision space should be considered WILSON e MARTINEZ (2000).

In probability theory, a probability density function (PDF) is the probability of

a random variable falling within a particular distribution. The greater PDF of that instances the greater the probability that instances belongs to that distribution. This assumption is considered in this work to select the most representative instances from a distribution. In an online scenario, this is possible comparing the old instances with the new arriving distribution. Therefore, density-based algorithms are able to select instances containing the new concepts of the new data distribution, capturing the drift of the distribution. Density-based algorithms can be suitable for work as a CSE. For example, CAPO *et al.* (2014) use Gaussian mixture models (GMM) as CSE method along with a semi-supervised algorithm for capturing new concepts in the distribution.

## GMM

GMM is a parametric probability density function represented as a weighted sum of Gaussian component densities (REYNOLDS, 2015) and uses the expectation-maximization algorithm (MOON, 1996) to estimate its parameters. Its complexity is  $O(nk)$ , where  $n$  is the data cardinality and  $k$  is the number of components.

One of the GMM's drawbacks is that the algorithm is not robust against noise (XIE *et al.*, 2011). Moreover, GMM seems to be suitable when the distribution is Gaussian and in the majority of the real datasets this condition is not true. A density-based algorithm that works even with noisy distributions (WU e QIN, 2018) is the Kernel Density Estimation (KDE).

## KDE

KDE, also denominated as Parzen window (DUDA *et al.*, 2012), is a density-based method that uses non-linear functions such as Sigmoid and Gaussian functions to compute local density values in the distribution and is widely applied to normalize and to smooth data. It can be considered as a generalization of the histogram. The complexity of the algorithm is  $O(n^2k)$  due to the fact that this method is a sum of matrix products. Thus, depending on the number of dimensions  $k$ , the algorithm become slow to calculate the PDF.

KDE has two variations: Univariate, that is suitable for one dimensional data and multivariate, suitable for data with two or more dimensions. It is widely applied in many machine learning algorithms such as support vector machines (HEARST *et al.*, 1998). The kernel function needs to be carefully adjusted due to underfitting or overfitting problems (BISHOP *et al.*, 2006) and has many variants (SILVERMAN, 2018). However, KDE may have difficulties to distinguish noise and drift in the distribution. Thus, another density-based algorithm can be applied: the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (ESTER *et al.*,

1996).

## DBSCAN

DBSCAN is a non-parametric density-based clustering technique that assumes that a cluster is a region in the data space with a high density. It considers the proximity and the number of instances inside a radius. If the instance has a minimum quantity of other instances inside its radius, then this instance is considered a core sample. Instances that does not have other instances inside its radius is considered noise. The remaining instances that are neither core sample or noise are influenced by the decision of the core samples. The complexity of the algorithm is at least  $O(n \log(n))$ . The algorithm requires two parameters:  $\epsilon$ , that is, the minimum distance between the points and  $minPts$ , that is, the minimum points that forms a dense region. Thus, if  $\epsilon$  is too small, a large part of data will not be clustered and the algorithm will consider this data as outlier, or noise. The  $minPts$  parameter, generally is adjusted from the number of dimensions  $D$  in the data set, as  $minPts \geq D + 1$  SANDER *et al.* (1998).

## 2.2 Related Work

There are frameworks that provides environments for making experiments with concept-drift. The Massive online analysis (MOA) (BIFET *et al.*, 2010), is a software environment for implementing algorithms and running experiments for evolving online learning. MOA includes a collection of offline and online methods as well as tools for evaluation. In particular, it implements boosting, bagging, and Hoeffding trees, all with or without Naive bayes classifiers at the leaves. MOA supports bidirectional interaction with the Waikato environment for knowledge analysis (WEKA) (HOLMES *et al.*, 1994) and is released under the GNU GPL license.

Change detection framework for massive online analysis (CD-MOA) (BIFET *et al.*, 2013) is a framework for evaluating change detection methods against intended outcomes. Another software, denominated scalable advanced massive online analysis (SAMOA) (MORALES e BIFET, 2015) is a platform for mining big data streams. It provides a collection of distributed streaming algorithms for the most common data mining and machine learning tasks such as classification, clustering, and regression, as well as programming abstractions to develop algorithms. It features a flexible architecture that allows it to run on several distributed stream processing engines.

TAVASOLI *et al.* (2016) applies the stochastic learning weak estimator (SLWE) (OOMMEN e RUEDA, 2006) for non-stationary environments. This algorithm uses weak estimators and counters to keep important data statistics at each time

instant as a new labeled instance arrives. Thus, the algorithm does not need retrain with each upcoming instances. SLWE was built to estimate the parameters of a binomial/multinomial distribution. DEMŠAR e BOSNIĆ (2018) proposed a concept-drift detector based on computing multiple model explanations over time and observing the magnitudes of their changes. Next, the model explanation is computed using a methodology that yields attribute-value contributions for prediction outcomes. Thus, it provides insight into the decision-making process and enables its transparency.

An ensemble learning method, denominated diversity and transfer based ensemble learning (DTEL) (SUN *et al.*, 2018), was proposed for incremental learning with concept-drift. DTEL employs a decision tree as the base learner and a diversity-based strategy for storing previous models. When a new data chunk arrives, the preserved models are exploited as initial points for searching/training new models via transfer learning. Thus, the newly obtained models are combined to form the new ensemble.

In semi-supervised learning, the data stream is typically divided into equal-sized chunks where only a small fraction of data is labeled. Thus, the goal is to label the remaining unlabeled data in the chunk. These algorithms can use labeled and unlabeled data together to perform the classification. One of the first semi-supervised algorithms applied to the problem of concept-drift was proposed by DITZLER e POLIKAR (2011b) and denominated weight estimation algorithm (WEA). The algorithm performs GMM on the unlabeled data and use the Bhattacharyya distance (KAILATH, 1967) between the resulting components from GMM.

WEA assumes that there is a limited concept-drift in the incremental learning scenario. By limited concept-drift, the assumption is that the drift is not completely random but there is a pattern in the drift. The work formalize limited drift assumption using the Bhattacharyya distance between a labeled component and its upcoming position. Thus, the unlabeled data at the time of testing must be less than the Bhattacharyya distance between the known component and every other unlabeled component from a different class.

DYER *et al.* (2014) proposed a semi-supervised framework for concept-drift, denominated compacted object sample extraction (COMPOSE). COMPOSE receive an initial amount of labeled data and perform the cluster-and-label algorithm (GULDEMIR e SENGUR, 2006) for labeling the remaining samples divided in equal-sized batches. COMPOSE uses geometric techniques, applying convex-hull and  $\alpha$ -shape algorithms, in order to map core support regions and extract optimal instances from predicted data. After, COMPOSE uses these extracted instances as training data in the next batch of unlabeled data.

Despite the interesting results of COMPOSE, the authors mentioned that the method becomes slow when the dimension of the data is greater than eight or when

the method is applied in streams. This is due to  $\alpha$ -shape algorithm that performs the convex-hull algorithm whose complexity is  $O(n^{\lfloor d/2 \rfloor})$  for dimensions  $d$  higher than three (EDELSBRUNNER *et al.*, 1983). The authors propose that this part of framework could be changed by a GMM.

An improvement of COMPOSE was developed using GMM in place of geometrical methods (CAPO *et al.*, 2014). The best number of GMM’s components is chosen through a number of tests with  $K$  number of components. These tests range between a set of numbers that represents the GMM’s components. The Bayesian information criteria (BIC) (SCHWARZ *et al.*, 1978) adds a penalty for large  $K$  to the negative log likelihood in order to prevent overfitting. Once the best model is chosen, core supports are extracted by calculating the Mahalanobis distance (MAHALANOBIS, 1936) for each  $x_i$  and each component in the GMM. The number that returns the smallest distance in these tests is chosen.

Even though COMPOSE using GMM achieved competitive accuracy results and better processing time than the original COMPOSE, there is still a high computational cost in the GMM’s components choice. The reason is that for each batch of data, the algorithm needs to process GMM several times whose complexity is  $O(nk)$ , where  $n$  is the data cardinality and  $k$  is the number of components. Besides, GMM may not fit well in distributions that is not derived from a Gaussian process.

Seeking for a balance between accuracy and processing time, UMER *et al.* (2016) proposed to improve the performance by replacing the selection of the core instances using a semi-supervised learning classifier with a sliding window approach. Their results indicate that training a semi-supervised classifier with all data overcome the previous issues.

SOUZA *et al.* (2015b) proposed the stream classification guided by clustering (SCARGC) algorithm. It was developed to perform classification on streams over a clustering strategy for non-stationary environments with EVL. The authors provide an amount of initial labeled data in the beginning of the stream. The classes are known and the algorithm tries to discriminate the classes apart. SCARGC classifies one instance at a time and updates the current and past cluster positions from clustering of the unlabeled data. Therefore, their approach can track the drifting classes over time.

After SCARGC, the same authors proposed MClassification (SOUZA *et al.*, 2015a). Based on its predecessor, they apply a clustering method to classify one instance at a time. This method is an adaptation from BIRCH algorithm (ZHANG *et al.*, 1997) also denominated as Micro-Cluster (AGGARWAL *et al.*, 2003). BIRCH is a compact representation that uses a triple: number of data points in a cluster; the linear sum of the  $N$  data points; the square sum of data points. Thus, for the adaptation to the classification problem, the MClassification method introduces

the received label in this triple, becoming a 4-tuple of information, doing the same approach of the BIRCH algorithm.

MClassification update the values of the micro-clusters according to the changes in the underlying distribution, retraining the classifier without the need of all instances but with the information contained in this 4-tuple. Thus, the method achieve a better processing speed than its predecessor SCARGC, since this model needs less instances to retrain due to the information contained in these 4-tuple. However, SCARGC presented better results.

UMER *et al.* (2017) proposed a framework for learning in EVL scenarios using importance weighting (LEVELiw). The authors argue that there is a significant overlap in gradual concept-drift scenarios and suggest to apply the importance weighting approach on this overlap. According to HACHIYA *et al.* (2012), importance weighting approaches have two assumptions: shared support of class-conditional distributions at two consecutive time steps; posterior distribution for each class remains the same.

However, according to UMER *et al.* (2017), importance weighting is intended for a single time step scenario with mismatched training and test datasets. The authors argues that iteratively performing importance weighting for each consecutive time step makes it a suitable method for online environments. Therefore, the authors propose that the classifier and CSE steps of COMPOSE can be replaced with an importance weighting based approach. In the next subsections, we detail COMPOSE GMM and LEVELiw, the most relevant algorithms for concept-drift. The main objective is to elucidate such algorithms for future comparisons with the proposal.

### 2.2.1 COMPOSE-GMM

The first version of COMPOSE (DYER *et al.*, 2014), applies a semi-supervised learning model using a geometric approach. Firstly, it learns a model with initial labeled data. Next, with the current labeled samples, it selects a subset of labeled instances, named core samples, from a geometric compaction method. Hence, samples that are not in the core subset are removed, remaining this new subset of samples. Finally, with the subset, the overall process repeats for the next unlabeled samples.

The authors suggest a cluster-and-label or label-propagation as a semi-supervised algorithm. According to the authors, COMPOSE iteratively adapts itself to the drifting environment and is intended for non-stationary environments that face incremental or gradual drift, rather than abrupt drift. Figure 2.7 illustrate the entire process performed by COMPOSE.

In the first step of the process, illustrated in the step a, COMPOSE trains with 5% of labeled data  $L^0$ . Next, in the step b, the framework receives batch of unlabeled

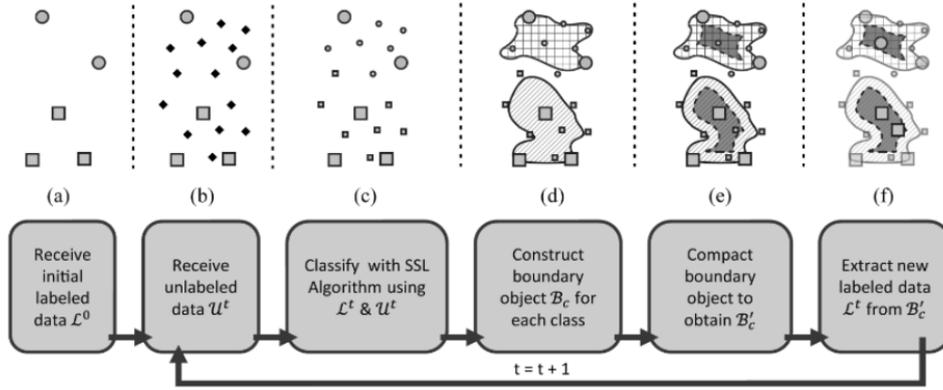


Figure 2.7: COMPOSE - Framework flow (DYER *et al.*, 2014).

samples  $U^t$  at time  $t$  and as illustrated in the step c, the SSL algorithm classifies the unlabeled data  $U^t$ . Step d shows the use of convex-hull for construct the boundaries of each class while the step e, illustrates the use of  $\alpha$ -shape algorithm that shrinks the boundaries of each class. At last, in step f, the framework extract the samples  $L^t$  from these boundaries. These samples will be used in the next batch of data (step b), restarting the process.

COMPOSE has two parameters:  $\alpha$  and  $CP$ . The first, controls the size of the convex-hull algorithm. The higher this value is, the greater is the region. Hence, less core samples are included in the core subset. Figure 2.8 illustrates how the parameter  $\alpha$  influence the construction of the core support region. On the other side, reducing the value, the region's size reduces and splits in more than one disjointed regions. The second parameter, controls the compaction percentage of the core region previously formed by the convex-hull and  $\alpha$ -shape methods. However, instead of shape size, the  $CP$  parameter controls the density of the core set. Therefore, a high value of  $CP$  decreases the probability of getting instances in the core set. Conversely, we have a density increase in the core set for low values of  $CP$ .



Figure 2.8: Evolution of the effects of decreasing the  $\alpha$  parameter (DYER *et al.*, 2014).

An alternative approach of COMPOSE,denominated COMPOSE-GMM, replaces the geometric alpha-shape to GMM. As previously stated by DYER *et al.* (2014), alpha-shape has a high execution time for data with dimensions higher than eight. Hence, infeasible for a stream scenario. Thus, the GMM approach of the COMPOSE is considered as an improvement and for this reason, this approach was chosen for

comparative purposes.

Additionally, the COMPOSE-GMM approach has two more distinct features than the original COMPOSE approach. One of these features is that the algorithm applies a Mahalanobis distance to measure the distance between an instance and a distribution. If the distance is zero then the point lies in the mean of the points from that distribution. How far a point is from that distribution more the distance increases. The second feature is that it depends on the EM learning procedure that has no guarantees of global optimum. However, local optimum are often sufficient.

In real world scenario is hard to know the global optimum values of its parameters. In order to optimize the GMM parameter  $K$ , the authors select the best value from a set of predetermined values. Additionally, the authors apply BIC to add a penalty for large values in order to prevent overfitting. In Algorithm 1, we detail the COMPOSE-GMM approach.

---

**Algorithm 1** COMPOSE-GMM

---

**Input:** Initial training data  $T$ ; batch of unlabeled data  $U^t$ ; Number of core supports  $p$

**Output:** Updated **BaseClassifier**; Label  $y$  for each  $x \in U^t$

**for**  $t = 0, 1, \dots$  **do**

    Receive unlabeled data,  $U^t \leftarrow \{x_u^t \in X, u = 1, \dots, N\}$

    Call **BaseClassifier** with  $L^t, y^t$  and  $U^t$ ; Obtain  $h^t : X \rightarrow Y$ , Let  $D^t \leftarrow \{(x_i^t, y_i^t) : x \in L^t \forall l\} \cup \{(x_u^t, h_u^t) : x \cup U^t \forall u\}$

    Set  $L^{t+1} \leftarrow \phi, y^{t+1} \leftarrow \phi$

**end for**

**for** each class  $c = 1, \dots, C$  **do**

$\alpha \leftarrow \infty$

**for** each number of components  $K \leftarrow \{1, 2, 3, 4, 5, 10, 20\}$  **do**

        Apply **GMM** with  $K$  and stores the log likelihood result

        Apply **BIC** in the stored log likelihood

**if** BIC value  $\leq \alpha$  **then**

$\alpha \leftarrow$  BIC value

**end if**

    Choose the best model according to the smallest BIC value

    Extract core support and add to labeled data for next time step,  $L_c^t \leftarrow$

**Mahalanobis\_distance**( $p, U_c^t$ )

**end for**

$L^{t+1} \leftarrow L^{t+1} \cup L_c^t$

$y^{t+1} \leftarrow y^{t+1} \cup \{y_u : u \in [|L_c^t|], y = c\}$

**end for**

---

Since COMPOSE-GMM only replaces the geometric procedures by GMM algorithm, the initial steps are the same of its predecessor, that is, the steps a, b and c in Figure 2.7. Next, GMM method is applied  $k$  times to labeled data  $L_c^t$ , for each class  $C$ . Each  $k$  is extracted from a set of elements representing a number of GMM components. Next, BIC procedure is applied at the log likelihood previously stored from GMM results. The  $\alpha$  parameter stores the lower value of BIC. Hence, the best model is chosen based on its smallest BIC associated. After that, core supports are extracted from the labeled data  $L_c^t$  by calculating the Mahalanobis distance for each sample and each component in the GMM. The number of core supports  $p$  with the smallest distances in these tests is chosen. Thus, the process is restarted with labeled data  $L^{t+1}$  for training the classifier and receive the next batch of unlabeled data, as previously illustrated at step b and c, in Figure 2.7.

### 2.2.2 LEVELiw

The algorithm matches distributions between two consecutive time intervals, estimating a posterior distribution of the unlabeled data using the importance weighted least squares probabilistic classifier algorithm (IWLSPC) (HACHIYA *et al.*, 2012). The estimated labels are then iteratively chosen as the training data for the next time step, providing an alternate solution to COMPOSE GMM with reduced parameter sensitivity.

The importance weights are estimated through unconstrained least squares importance fitting (uLSIF). In summary, the authors suitably modified IWLSPC, originally proposed for only single time step problems, where it was applied to match the divergence in the training and test distributions on a non-streaming data application. The method was extended to problems in which data arrive in a continuous streaming fashion with EVL.

At initial time step  $t = 0$ , LEVELiw receives data  $x$  with their corresponding labels  $y$ . Initializes the test data  $x_{te}^{t=0}$  to initial data  $x$  received and sets their corresponding labels  $y_{te}^{t=0}$  equal to the initial labels  $y$ . Then, the algorithm iteratively processes the data at each time step  $t$ . Hence, a new unlabeled test dataset  $x_{te}^t$  is first received.

The previously unlabeled test data from previous time step  $x_{te}^{t-1}$  which is now labeled by the IWLSPC subroutine, becomes the labeled training data  $x_{tr}^t$  for the current time step. Similarly, the labels  $y_{te}^{t-1}$  obtained by IWLSPC during the previous time step become the labels of the current training data  $x_{tr}^t$ . The training data at the current time step  $x_{tr}^t$ , the corresponding label information at the current time step  $y_{tr}^t$ , the kernel bandwidth value  $\sigma$  and the unlabeled test data at the current time step  $x_{te}^t$  are then passed into the IWLSPC algorithm, which predicts the labels

$y_{te}^t$  for the test unlabeled data.

The critical parameter in model selection for IWLSPC is the kernel width  $\alpha$ , which is obtained through importance weighted cross validation (IWCV) in IWLSPC’s original description and is updated each time step separately. In this effort, the author keeps the parameter constant through the experiment. The reason is because a cross validation is unrealistic for each time step in an online environment, and the authors wanted to determine the sensitivity of this parameter on the algorithm classification performance. We detail the Algorithms correspondent to IWLSPC 2 and LEVELiw 3.

---

**Algorithm 2** IWLSPC

---

**Input:** Unconstrained least squares importance fitting **uLSIF**; Importance weighted cross validation **IWCV**  
 Receive training data  $x_{tr}$   
 Receive test data  $x_{te}$   
 Run **uLSIF** to estimate importance weights  
 Run **IWCV** to estimate Gaussian kernel width  $\sigma$   
 Compute Gaussian Kernel Function using  $\sigma$  as defined in Equation 2  
 Estimate parameter  $\theta_y$  by minimizing squared error  $J_y(\theta_y)$  as defined in Equation 3  
 Use  $\theta_y$ , and the Gaussian Kernel function to compute posterior probability as defined in Equation 1

---



---

**Algorithm 3** LEVELiw

---

**Input:** Initial training data  $T$ ; batch of unlabeled data  $U^t$ ; Importance weighted least squares probabilistic classifier - **IWLSPC**; Kernel bandwidth value  $\sigma$   
**Output:** Updated classifier; Label  $y$  for each  $x \in U^t$   
 At  $t = 0$ , receive initial data  $x \in X$  and the corresponding labels  $y \in Y = 1, \dots, C$   
 Set  $x_{te}^{t=0} \leftarrow x$   
 Set  $y_{te}^{t=0} \leftarrow y$   
**for**  $t = 1, \dots$  **do**  
   Receive unlabeled test data  $x_{te}^t \in X$   
   Set  $x_{tr}^t \leftarrow x_{te}^{t-1}$   
   Set  $y_{tr}^t \leftarrow y_{te}^{t-1}$   
   Call **IWLSPC** with  $x_{tr}^t, x_{te}^t, y_{tr}^t$  and  $\sigma$  to estimate  $y_{te}^t$   
**end for**

---

As stated by UMER *et al.* (2017), if there is a significant overlap, there is also a significant shared support. Thus, LEVELiw assigns higher weights to those instances

that are drawn from the shared support region. The authors observed its robustness and higher tolerance to fluctuations around the optimal value of its free parameter.

# Chapter 3

## AMANDA - Density-based Adaptive Model for Non-stationary Data

In non-stationary environments, single adaptive, or dynamic learning algorithms, are widely applied for handling concept changes (KHAMASSI *et al.*, 2018). Despite the efficiency of ensemble algorithms (JACKOWSKI, 2014, LU *et al.*, 2015), using single learner approaches are interesting for controlling the complexity of the system, since they were designed to be adapted in real time and with minimum computational efforts (KHAMASSI *et al.*, 2018). Therefore, the first experiments of AMANDA were developed using this strategy. Besides, the key point of handling concept drift is to define the way how the learner will be adapted. For this purpose, there are two main categories: informed methods and blind methods (KHAMASSI *et al.*, 2018).

Informed methods, explicitly detect the drift using triggering mechanisms whereas blind methods implicitly adapts to changes without any drift detection. Blind methods discard old concepts at a constant speed, independently of whether changes have happened or not. For this work we chose a blind method approach since this approach is suitable for handling gradual continuous drifts where the dissimilarity between consecutive data sources is not quite relevant to trigger a change (KHAMASSI *et al.*, 2018).

As explained in the Chapter 2.2, there are a few drawbacks in current state-of-the-art methods. For example, COMPOSE-GMM and LEVELiw do not exploit properly the temporal dependence in concept-drift datasets, since they have difficulties to keep the most representative instances in the last batch of data. Besides, these methods have higher computational times, decreasing their performances in batch scenarios for a large number of samples or high dimensionality of data. Taking into account the mentioned problems, here we propose a semi-supervised framework,

denominated AMANDA, with two derived classifiers: AMANDA-FCP and AMANDA-DCP. Additionally, we propose a core selection method in order to support these classifiers for dealing with gradual drift on scenarios with extreme verification latency.

### 3.1 Amanda Framework

AMANDA framework, as illustrated in Figure 3.1, has five processing steps: starting, learning, classification, weighting, and filtering. The first step is the starting phase. At this step, the framework receives labeled samples. This step is critical because it defines where the distributions of each class begin. Moreover, it occurs once since labeled samples are supplied only here, according to EVL assumption. The second step is the learning phase. At this step, a model learns using a SSL classifier.

The third step is the classification phase. At this step the SSL classifies upcoming batches with unlabeled samples. The fourth step is the instance weighting phase. At this step, a density-based algorithm measures the importance of the classified samples by weighting them. Finally, the fifth step that is the filtering phase. At this step, the weighted samples are filtered and only the most representative samples remains. With these selected samples, the process backs to the second step. Worth to mention that the dashed line in Figure 3.1 is related to the core support extraction method. This method contains the weighting and filtering phases that is explained in Section 3.2.

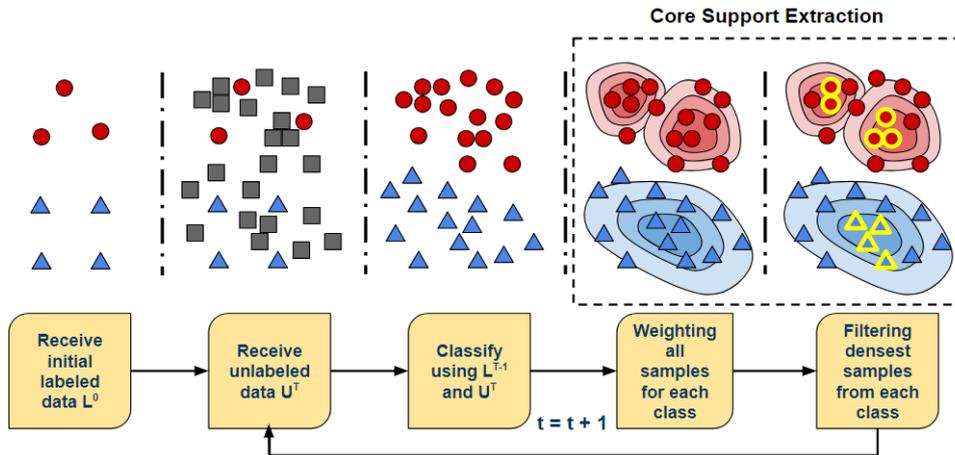


Figure 3.1: AMANDA framework.

## 3.2 Core Support Extraction

The core support extraction (CSE) procedure that we explore attempts to accurately identify which instances lies into the core region of the existing class distributions. These instances are previously labeled by the last SSL step. After that, these instances are weighted and filtered by our CSE method. Hence, these filtered instances are chosen as training data for the next iteration’s SSL step for labeling the new unlabeled data. Our assumption is that these instances are the most representatives samples of the distribution. Therefore, in this section, we detail the two phases that compose our CSE method: weighting and filtering methods.

The weighting method receives a set of instances as input and returns the same set of instances associated with weights. These weights are estimated by the KDE algorithm. Therefore, KDE calculates density curves from a given number of observations  $n$ . These curves takes into account the distance of each point in distribution from a central value, the kernel. The kernel is a standardized weighted function and determine the smoothing of the probability density function (PDF). Thus, each instance is associated with a PDF value. These values indicate the denser samples of the distribution. The choice of this algorithm among the other existent density-based algorithms is explained in Chapter 4.

The filtering method receives a set of weighted instances and the percentage  $\alpha$  of available observations to retain as training data for upcoming time steps. We select the  $\alpha$ -most-dense instances from the set of weighted instances. The remaining samples are discarded and the filtering method returns the subset of instances that are determined to be the core supports.

## 3.3 AMANDA-FCP

AMANDA with fixed cutting percentage (AMANDA-FCP) uses a SSL classifier along with a density-based algorithm that works as a CSE method. Besides, the amount of instances to be discarded is determined by the  $\alpha$  parameter. The AMANDA-FCP is presented by the Algorithm 4.

Firstly, AMANDA-FCP receives a batch of labeled data  $L^0$ , a SSL Classifier  $\phi$  and a cutting percentage parameter  $\alpha$ . The online processing begins and AMANDA-FCP receives  $T$  batches of unlabeled data in an iterative way. Thus, for each batch  $U^t$  for an instant  $t$ , the SSL uses the initial instances  $X^{t-1}$  and their respective labels  $y^{t-1}$  from  $L^{t-1}$  for learning. After, SSL classifies the unlabeled data  $U^t$ . The instances  $U^t$  and their recent classified labels  $y^t$  are stored in  $L^t$ . Thus, the current distribution  $L^t$  along with the past distribution  $L^{t-1}$  becomes an input of the CSE method.

Our CSE method uses the KDE algorithm as a weighting method. The weighting

---

**Algorithm 4** AMANDA - FCP

---

**Input:** Labeled data  $L^0$ ; Unlabeled data  $U$ ; SSL Classifier  $\phi$ ; Cutting percentage  $\alpha$

**Output:** Updated classifier  $\phi$ ; Label  $y$  for each  $x \in U^t$

**for**  $t = 1, \dots, T$  **do**

$X^{t-1} \leftarrow \{X_l^{t-1} \in L^{t-1}, l = 1, \dots, N\}$

$y^{t-1} \leftarrow \{y_l^{t-1} \in L^{t-1}, l = 1, \dots, N\}$

$\phi \leftarrow \text{Train } \phi(X^{t-1}, y^{t-1})$

$y^t \leftarrow \phi(U^t)$

$L^t \leftarrow U^t y^t$

$D^t \leftarrow \text{CSE}(L^t \cup L^{t-1})$

$L^{t-1} \leftarrow \emptyset$

**for** each class  $c \in C$  **do**

$L_c^{t-1} \leftarrow \sigma(D^t, 1 - \alpha)$

**end for**

**end for**

---

method calculates a PDF for  $L^t$  in relation to  $L^t \cup L^{t-1}$ . After, the filtering method receives the weighted instances  $D^t$  and for each class, the method  $\sigma$  selects the densest instances  $L_c^{t-1}$ . The number of instances to be kept is given by  $1 - \alpha$ . Finally, these densest instances become the training data  $L^{t-1}$  for the SSL in the next batch and the overall process repeats until no batch of data is available.

### 3.4 AMANDA-DCP

AMANDA with dynamic cutting percentage (AMANDA-DCP) uses a SSL classifier and a density-based algorithm along a modified version of HDDDM method (DITZLER e POLIKAR, 2011a) that works as CSE method. AMANDA-DCP receives a batch of labeled data  $L^0$  and a SSL Classifier  $\phi$ . The online processing begins and AMANDA-DCP receives  $T$  batches of unlabeled data in an iterative way. Thus, at each batch  $U^t$  for an instant  $t$ , the SSL uses the instances  $X^{t-1}$  and their respective labels  $y^{t-1}$  from  $L^{t-1}$  for learning.

After, SSL classifies the unlabeled data  $U^t$ . The instances  $U^t$  and their recent classified labels  $y^t$  are stored in  $L^t$ . Thus, the current distribution  $L^t$  along with past distribution  $L^{t-1}$  becomes an input of the CSE method. Our CSE method uses the KDE algorithm as a weighting method. Our weighting method calculates the PDF of  $L^t$  in relation to  $L^t \cup L^{t-1}$ . Until this phase, AMANDA-DCP and AMANDA-FCP works in a similar way.

Next,  $L^{t-1}$  and  $L^t$  distributions become input for the filtering method. Our

cutting percentage calculation method  $\rho$  sets dynamically the parameter  $\alpha$ . It uses the support of a modified version of HDDDM method. The HDDDM method compares the past and the current attributes of instances using the Hellinger distance. Therefore, the amount of discarded instances is calculated dynamically. The cutting percentage calculation method is presented in Algorithm 6.

Finally, similarly to the AMANDA-FCP, the filtering method  $\sigma$  receives the weighted instances  $D^t$  and, for each class, selects the densest instances  $L_c^t$ . The number of instances to be kept is given by  $1 - \alpha$ . These densest instances become the training data  $L^{t-1}$  for the SSL and the process repeats until there are no more batches. The Algorithm 5 illustrates AMANDA-DCP.

---

**Algorithm 5** AMANDA - DCP

---

**Input:** Labeled data  $L^0$ ; Unlabeled data  $U$ ; Classifier  $\phi$

**Output:** Updated classifier  $\phi$ ; Label  $y$  for each  $x \in U^t$

**for**  $t = 1, \dots, T$  **do**

$X^{t-1} \leftarrow \{X_l^{t-1} \in L^{t-1}, l = 1, \dots, N\}$

$y^{t-1} \leftarrow \{y_l^{t-1} \in L^{t-1}, l = 1, \dots, N\}$

$\phi \leftarrow \text{Train } \phi(X^{t-1}, y^{t-1})$

$y^t \leftarrow \phi(U^t)$

$L^t \leftarrow U^t y^t$

$\alpha \leftarrow \rho(L^t, L^{t-1})$

$D^t \leftarrow \text{CSE}(L^t \cup L^{t-1})$

$L^{t-1} \leftarrow \emptyset$

**for** each class  $c \in C$  **do**

$L_c^{t-1} \leftarrow \sigma(D^t, 1 - \alpha)$

**end for**

**end for**

---

The cutting percentage algorithm receives two distributions  $L^{t-1}$  and  $L^t$ . The algorithm iterates through  $N$  number of data attributes creating two vectors,  $u$  and  $v$ , with the  $n_{th}$  attributes from  $L^{t-1}$  and  $L^t$ , respectively. After, two histograms,  $h_u$  and  $h_v$ , are calculated from the vectors  $u$  and  $v$ , respectively. The number of bins used in the histograms is the squared root of the length of the vectors. Thus, the Hellinger distance can be applied to a multidimensional data. Hence, the Hellinger distance  $\tau$  is computed between  $h_u$  and  $h_v$ .

After, the algorithm computes the average distance between the two distributions and sets the temporary value of cutting percentage  $\alpha$ . This temporary value is normalized by the difference between  $Z$  and the average distance, calculated early. Here,  $Z = \sqrt{2}$  since this value is the upper bound value of the metric. Finally, the algorithm verifies if the value of the temporary  $\alpha$  parameter is between the range

that we stipulated for our  $\alpha$  parameter. Thus, it corrects values lower than the  $\beta$  parameter or greater than the  $\omega$  parameter. More details about the experiments regarding this parameter  $\alpha$  is presented in Section 4.4.2.

---

**Algorithm 6** Cutting percentage calculation

---

**Input:** Two distributions  $L^{t-1}$  and  $L^t$   
**Output:** Cutting percentage  $\alpha$   
 $N \leftarrow$  number of attributes of  $L^{t-1}$   
**for**  $i = 1, \dots, N$  **do**  
     $u \leftarrow$  vector of  $i_{th}$  attribute of the instances  $X^{t-1}$  from  $L^{t-1}$   
     $v \leftarrow$  vector of  $i_{th}$  attribute of the instances  $X^t$  from  $L^t$   
     $h_u \leftarrow histogram(u, bins)$   
     $h_v \leftarrow histogram(v, bins)$   
     $h \leftarrow h + \tau(h_u, h_v)$   
**end for**  
 $\alpha \leftarrow \max(\min(Z - \frac{h}{N}, \omega), \beta)$

---

CIESLAK e CHAWLA (2009) suggest Hellinger distance, not for detecting concept drift in an incremental learning setting, but rather to detect bias between training and test data distributions. In probability and statistics, the Hellinger distance is applied to quantify the similarity between two probability distributions. It is a type of f-divergence (LIN, 1991) and is closely related to, although different from, the Bhattacharyya distance (KAILATH, 1967). The authors applied a non-parametric statistical test, measuring the significance between the probability estimates of the classifier on a validation set and the corresponding test dataset. Thus, a baseline comparison is made by calculating the Hellinger distance between the original training and test datasets. Bias is then injected into the testing set. Finally, the results between the baseline Hellinger distance and the distance after bias is injected are observed.

For this work, two cutting percentage approaches were applied and define how much data needs to be discarded. The first is fixed during all process and must be chosen *a priori*. However, determining a cutting percentage value is not an easy task and demands that the initial training phase identifies the distribution changes, to posteriorly apply the model in production. This concern motivated the second approach, where we suggest the use of the Hellinger distance metric to compare two distributions, past and current. Thus, this metric determines how much these two distributions are similar between them.

This approach is applied for every batch, and indicates how much data could be discarded from these two distributions. Therefore, we assume that the cutting percentage is the percentage of similar data inside the union of the two distributions.

Since this calculation is applied for every batch, the cutting percentage becomes dynamic, assuming different values during the process.

The cutting percentage method was inspired by a work that suggested a Hellinger distance drift detection method (HDDDM), that is, a Hellinger distance adaptation for drift detection for non-stationary environments (DITZLER e POLIKAR, 2011a). The HDDDM is a feature based drift detection method, using the Hellinger distance between current data distribution and a reference distribution that is updated as new data are received. The Hellinger distance is an example of divergence measure, similar to the Kullback-Leibler (KL) divergence (JOYCE, 2011). However, unlike the KL-divergence, the Hellinger distance is a symmetric metric and makes three assumptions: firstly, labeled training datasets are presented in batches to the drift detection algorithm, as the Hellinger distance is computed between two histograms of data.

Secondly, data distributions have finite support, fixing the number of bins in the histogram required to compute the Hellinger distance at  $\sqrt{N}$ , where  $N$  is the number of instances at each time stamp presented to the drift detection algorithm. Finally, in order to follow a true incremental learning setting, each instance is only seen once by the algorithm.

The Hellinger distance is given by  $\frac{\|\sqrt{p}-\sqrt{q}\|}{\sqrt{2}}$  and forms a bounded metric on the space of probability distributions over a given probability space. The maximum distance is achieved when  $p$  assigns probability zero to every set to which  $q$  assigns a positive probability, and vice-versa. Hellinger distance ranges from zero to  $\sqrt{2}$ .

The choice of Hellinger distance over other measures such as the Mahalanobis (MAHALANOBIS, 1936), is due to the no assumptions made about the distribution of the data. Also, the Hellinger distance is a measure of distributional divergence that allow to measure the change between the distributions of data at two subsequent time stamps.

# Chapter 4

## Empirical Evaluation

This chapter presents the objectives of the experiments, the chosen datasets, the methodology for conducting experiments and our empirical findings.

### 4.1 Experiment Objectives

One objective of the experiments is to evaluate how the core support extraction method along with a SSL is capable to overcome the concept-drift problem. Another objective is to validate AMANDA-FCP and AMANDA-DCP through the most representative datasets with concept drift, comparing against three baselines and two state-of-the-art classifiers: COMPOSE (CAPO *et al.*, 2014) and LEVELiw (UMER *et al.*, 2017).

### 4.2 Datasets

For this work we chose seventeen synthetic datasets and three real ones. These datasets were extensively applied as a benchmark of semi-supervised learning algorithms on batch and stream scenarios with extreme verification latency (CAPO *et al.*, 2014, DYER *et al.*, 2014, SOUZA *et al.*, 2015a,b). Table 4.1 present the properties of artificial and real datasets (marked with '\*'). All datasets present at least one type of concept-drift changes. Moreover, the datasets are balanced, except by: 1CSurr, Electricity and NOAA.

#### 4.2.1 Synthetic Datasets

For the synthetic datasets, we adopt the following acronyms: One Class Diagonal Translation (1CDT), Two Classes Diagonal Translation (2CDT), One Class Horizontal Translation (1CHT), Two Classes Horizontal Translation (2CHT), Four

Table 4.1: Datasets.

Datasets	Classes	Features	Instances	Drift
1CDT	2	2	$1.6 \times 10^4$	$4 \times 10^2$
2CDT	2	2	$1.6 \times 10^4$	$4 \times 10^2$
1CHT	2	2	$1.6 \times 10^4$	$4 \times 10^2$
2CHT	2	2	$1.6 \times 10^4$	$4 \times 10^2$
4CR	4	2	$1.4 \times 10^5$	$4 \times 10^2$
4CRE-V1	4	2	$1.2 \times 10^5$	$1 \times 10^3$
4CRE-V2	4	2	$1.8 \times 10^5$	$1 \times 10^3$
5CVT	5	2	$4 \times 10^4$	$1 \times 10^3$
1CSurr	2	2	$5.5 \times 10^4$	$6 \times 10^2$
4CE1CF	5	2	$1.7 \times 10^5$	$7 \times 10^2$
UG_2C_2D	2	2	$1 \times 10^5$	$1 \times 10^3$
MG_2C_2D	2	2	$2 \times 10^5$	$2 \times 10^3$
FG_2C_2D	2	2	$2 \times 10^5$	$2 \times 10^3$
UG_2C_3D	2	3	$2 \times 10^5$	$2 \times 10^3$
UG_2C_5D	2	5	$2 \times 10^5$	$2 \times 10^3$
GEARS_2C_2D	2	2	$2 \times 10^5$	$2 \times 10^3$
CheckerBoard	2	2	$6 \times 10^4$	$3 \times 10^2$
NOAA*	2	8	18159	unknown
Electricity*	2	7	27552	unknown
Keyboard*	4	10	1600	200

Classes Rotating Separated (4CR), Four Classes Rotating with Expansion V1 (4CRE-V1), Four Classes Rotating with Expansion V2 (4CRE-V2), Five Classes Vertical Translation (5CVT), Two Bidimensional Unimodal Gaussian Classes (UG-2C-2D), Two Bidimensional Multimodal Gaussian Classes (MG-2C-2D), Two Bidimensional Classes as Four Gaussians (FG-2C-2D), Two 3-dimensional Unimodal Gaussian Classes (UG-2C-3D), Two 5-dimensional Unimodal Gaussian Classes (UG-2C-5D), Two Rotating Gears (GEARS-2C-2D) and Rotating Checkerboard.

It should be noted that the majority of the synthetic datasets contains two dimensions with two balanced classes. However, the UG\_2C\_3D and UG\_2C\_5D

datasets have three and five dimensions, respectively, while 4CR, 4CRE-V1 and 4CRE-V2 datasets have two dimensions and four classes. The 4CE1CF, 5CVT datasets contain two dimensions and five classes. The only synthetic imbalanced dataset is the 1CSurr with two dimensions and two classes. Thus, we believe that conducting the experiments in these synthetic data can lead us to a better understanding of the behavior of the proposal and other approaches, since these datasets represent the various types of drift.

## 4.2.2 Real Datasets

The first real dataset is provided by The U.S. National Oceanic and Atmospheric Administration (NOAA). It has weather measurements from over 9000 weather stations worldwide <sup>1</sup>. Records date back to the 1930s, providing a wide scope of weather trends. Daily measurements include a variety of features such as temperature, pressure, wind speed, indicators of precipitation and other weather-related events. Data comes from the Offutt Air Force Base in Bellevue, Nebraska, due to its extensive range of 50 years (1949-1999) and diverse weather patterns.

This dataset has eight features based on their availability, eliminating those with a missing feature rate above 15%. The remaining missing values were imputed by the mean of features in the preceding and following instances. Class labels are based on the binary indicator(s) provided for each daily reading of rain with 18,159 daily readings: 5,698 (31%) positive (rain) and 12,461 (69%) negative (no rain). This normalized data was built and applied in the experiments of DITZLER e POLIKAR (2011a), DYER *et al.* (2014), ELWELL e POLIKAR (2011). The dataset contains 583 consecutive months time steps covering 50 years. Besides, this dataset was applied in two of the state-of-the-art algorithms (CAPO *et al.*, 2014, SOUZA *et al.*, 2015a) chosen for this work.

The second dataset is the electricity market dataset (ELEC2). Data was collected from the Australian New South Wales electricity market and it was first described by HARRIES e WALES (1999). In this market, prices change every time and are affected by demand and supply. HARRIES e WALES (1999) shows the seasonality of prices and their sensitivity to short-term events such as weather fluctuations. Another factor on price changes is the the electricity market evolution through time. During the time of market events, the electricity market was expanded with the inclusion of adjacent areas. This allowed for a more elaborated management of the supply. The excess production of one region could be sold on the adjacent region. A consequence of this expansion is a dampener of the extreme prices (HARRIES e WALES, 1999).

---

<sup>1</sup>FTP: <ftp.ncdc.noaa.gov/pub/data/gsod>

The normalized and cleaned ELEC2 dataset contains 27,552 instances dated from 7 May 1996 to 5 December 1998. Each example of the dataset refers to an interval of 30 minutes. Hence, there are 48 instances for each interval of day. Each example of the dataset has five attributes: the day of the week, the time stamp, the NSW electricity demand, the Vic electricity demand, the scheduled electricity transfer between states and a class label. The class label represents the change of the price related to a moving average of the last 24 hours. The class label only reflects deviations of the price on a one day average without impact of longer term price trends. Finally, it should be noted that this dataset were extensively applied regarding non-stationary environments research (BAENA-GARCÍA *et al.*, 2006, BIFET e GAVALDA, 2007, BIFET *et al.*, 2009, BRZEZIŃSKI e STEFANOWSKI, 2011, GAMA *et al.*, 2004).

The third real dataset is the Keystroke. It is based on the use of keystroke dynamics to recognize users by their typing rhythm instead of the simple login and password verification. The detection of this pattern inserted as a second security layer for user authentication without any additional hardware costs. However, the system needs regularly updating the user profile because it evolves incrementally over time as suggested by ARAÚJO *et al.* (2005). This dataset was built from keystroke dynamics based on CMU data (KILLOURHY e MAXION, 2010). In CMU data, 51 users type the password *.tie5Roanl* plus the *Enter* key 400 times. This typed word is captured in 8 sessions performed in different days.

In the keystroke dataset, the task is to classify between four users based on typing patterns. To perform the user classification task, SOUZA *et al.* (2015a) chose ten features extracted from the flight time for each pressed key. The flight time corresponds to the time interval between a key is released and a next key is pressed. The data is generated by having each user type a phrase repeatedly. It is expected that the users will become faster at typing the phrase over time and therefore induce concept drift. In this stream dataset, SOUZA *et al.* (2015a) randomly chose four users and merged them respecting the chronological order in a total of 1,600 examples. It is worth mentioning that this dataset was applied for non-stationary environments (SOUZA *et al.*, 2015a, UMER *et al.*, 2017).

### 4.3 Methodology and Empirical Setup

In this subsection we show the scenarios simulated for this work and the chosen metrics to validate the experiments. Besides, we present the methods that were chosen as baselines for this work.

### 4.3.1 Simulation Scenarios

In our experiments, we simulate a batch scenario, beginning with 5% of labeled samples. The remaining samples are unlabeled and are divided in 100 parts such that they arrive in a chronological order (CAPO *et al.*, 2014, DYER *et al.*, 2014, UMER *et al.*, 2016, 2017). It should be noted that the amount of initial labeled data follows the definition of extreme verification latency scenarios (CAPO *et al.*, 2014, DYER *et al.*, 2014, UMER *et al.*, 2016, 2017).

### 4.3.2 Validation and Metrics

In order to evaluate the classifiers, we apply a prequential evaluation with a sliding window (DAWID, 1984). In prequential evaluation, the error is computed sequentially and in chronological order. The overall error is computed based on an accumulated sum of a loss function (GAMA *et al.*, 2009). Moreover, once each classifier has hyper-parameters, we fine tuned them through a grid search (MARKELLOS *et al.*, 1974).

Regarding the metrics attached into prequential, we chose the classification error and macro-F1 metrics for balanced datasets. Once the classification error measure the overall error considering all classes, it is not suitable for imbalanced datasets. Therefore, we switched to Matthews Correlation Coefficient (MCC) (MATTHEWS, 1975). It should be noted that macro-F1 is already suitable for both balanced and imbalanced datasets. We decided not applying micro-f1 due to its preference for the majority class. Finally, we measure processing time and the average reduction error over the static classifier for all datasets.

### 4.3.3 Classifiers and Baselines

For the choice of the SSL classifier, five different learning classifiers were chosen: Stochastic gradient descent (SGD) (ROBBINS e SIEGMUND, 1971), K-nearest neighbors (K-NN) (DUDANI, 1976), label propagation (LP) (ZHU e GHARAMANI, 2002), random forests (RF) (LIAW *et al.*, 2002) and Gaussian naive Bayes (GNB) (JOHN e LANGLEY, 1995). In order to define reliable baselines for the problem (CHAO, 2015, DE SOUZA *et al.*, 2013, ŽLIOBAITĚ *et al.*, 2015), we chose three specific baseline methods for non-stationary environments:

- **Static Classifier:** A classifier learnt from the first labeled samples. The goal here is to measure the relevance of the first labeled samples regarding upcoming drifts.
- **Sliding Window Classifier:** A classifier that learns initially with the labeled

samples and updates its model with the predicted upcoming samples. This classifier uses a sliding window for discarding old samples from training.

- **Incremental Classifier:** Similar to the Sliding Window Classifier. However, the window increases with each insertion of upcoming samples without discarding old ones.

## 4.4 Results

In this section, we analyze the choice of density-based algorithm as CSE method; we analyze the parameter's influence of the AMANDA-FCP method; we present a detailed analyses of our proposal results for the artificial datasets and real ones; and present a comparative of our proposal against the baselines and the state-of-the-art classifiers for all datasets.

### 4.4.1 Density-based Algorithms Choice for CSE

We tested four approaches for selecting instances: KDE, GMM and two approaches using DBSCAN. In the first DBSCAN approach, we kept only the core samples, discarding the noise and the non-core instances. For each batch, the SSL classifier used only the last core-samples determined by DBSCAN. We denominated this approach as DBSCAN-1. In the second DBSCAN approach, we only discarded the samples marked as noise by the algorithm, keeping the core samples along with non-core samples. The number of instances that the SSL used was bigger than the DBSCAN-1, decreasing the chance of overfitting. We denominated this approach as DBSCAN-2.

We tested four approaches for all artificial datasets and for three real ones as well. The approach using GMM obtained the best results for nine artificial datasets. The approach using KDE had the best results in five artificial datasets and in one real dataset. DBSCAN-1 obtained the best results on four artificial datasets whereas DBSCAN-2 had the best results on three artificial datasets. Therefore, the GMM and KDE approaches seemed to be the two best approaches for using as a CSE method.

Figure 4.1 shows how KDE and GMM spread inside a class surrounding another class. GMM spread in an elliptic way while KDE spread in other directions beyond the center of data, what gives advantage to select the core sets for the subsequent steps. Another advantage of KDE over GMM is that KDE does not need to make strong assumptions about the distribution such as mean and standard deviation. Besides, GMM needs that the number of mixture models must be chosen *a priori*. In real problems this information is not known and may be hard to test and to find

the best number of components. Therefore, we chose the KDE algorithm to work as our weighting algorithm for our CSE method. For this work, the KDE uses the Gaussian kernel due to the smooth estimation that this kernel generates.

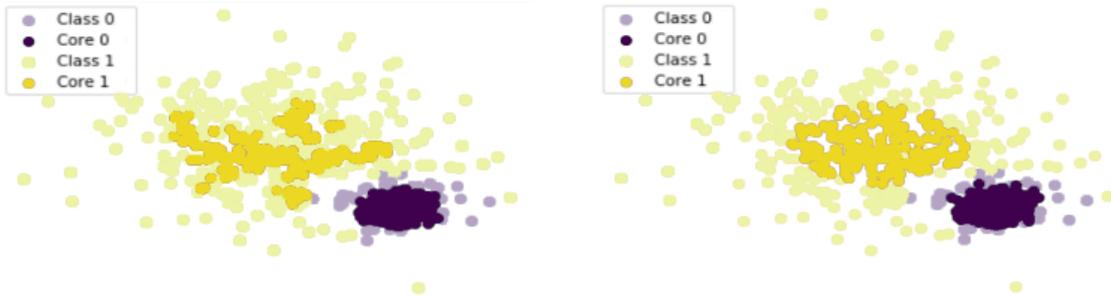


Figure 4.1: KDE performance versus GMM performance in CSurr dataset.

#### 4.4.2 AMANDA Parameter’s Influence

The main parameter that has influence over AMANDA-FCP results is the cutting percentage. Even this parameter reduces the computational time, it must be selected with caution. With a high value, a large number of samples are discarded from a training set. Then, the bias of a learnt model increases. Otherwise, for lower values, there is an increase in variance.

The choice of the values 0.5 and 0.9 as lower bound and upper bound, respectively, is due to two problems realized during the experiments: increasing of training data; and overfitting. When the parameter  $\alpha$  is lower than  $\frac{|L^t|+|L^{t-1}|}{2}$ , there is the risk of the quantity of the kept instances becomes greater through the time. It generates an increase in the training data and an increase in the possibility of the SSL to use old concepts to learn working similarly to the incremental classifier. Besides, the processing time is increased due to the CSE processing.

Otherwise, when the parameter  $\alpha$  is greater than 90%, there is a risk to retain only a few instances for training, generating an overfitting in the training data. Besides, with a very few instances the model has difficulty to deal with imbalanced distributions or distributions where the class boundary is surrounded by another. Therefore, we vary the cutting percentage parameter and perform experiments for all SSL classifiers described in subsection 4.3.3.

We tested the five classifiers for all datasets and even though label propagation obtained the best results in 70% of the artificial datasets, the K-NN algorithm had the best results in two of three real datasets. In Table 4.2, we show the best accuracy results of five classifiers applied to five datasets on a development set. Our results indicate that Label Propagation and K-NN obtained the most accurate results. Thus, the experiments were conducted only with label propagation or K-NN algorithms as

a parameter of the AMANDA method.

Dataset	SGD	K-NN	LP	RF	GNB
UG2C5D	50.33	91.79	90.64	73.23	<b>92.77</b>
1CSURR	48.46	93.3503	<b>95.61</b>	61.77	92.58
NOAA	59.48	<b>69.11</b>	68.68	62.30	59.85
Electricity	60.23	67.87	66.53	<b>69.83</b>	65.32
Keyboard	66.44	<b>88.81</b>	88.22	86.51	77.63

Table 4.2: Classifiers accuracy.

In order to elucidate the variation of the cutting percentage parameter  $\alpha$  of the previous experiment, we show the parameter variation over the most accurate classifiers for the keystroke dataset in Table 4.3. The best parameters for Label Propagation were  $K = 4$  for the Keystroke dataset and  $K = 7$  for the 1CSURR dataset, whereas the best parameters for K-NN were  $K = 7$ , for the Keystroke and for the 1CSURR datasets.

Algorithms	$\alpha = 0.5$	$\alpha = 0.55$	$\alpha = 0.6$	$\alpha = 0.65$	$\alpha = 0.7$
K-NN	85.39	86.18	87.17	87.76	87.76
LP	72.04	70.13	71.38	73.15	87.23
Algorithms	$\alpha = 0.75$	$\alpha = 0.8$	$\alpha = 0.85$	$\alpha = 0.9$	–
K-NN	88.68	90.13	89.27	89.14	–
LP	87.43	88.68	85.46	88.29	–

Table 4.3: Average accuracy for the keystroke dataset

For the next subsections, we present the batch results for artificial and real datasets. For the artificial datasets, we analyze the five most representative datasets among the seventeen synthetic datasets. For the real datasets, we detail the results for the three datasets. After that, we present a summary of the results for all datasets.

### 4.4.3 Artificial Dataset - 2CDT

This is a balanced dataset with two dimensions and two classes that shifts through the Euclidean space. The evolution of the drift is illustrated in Figure 4.2.

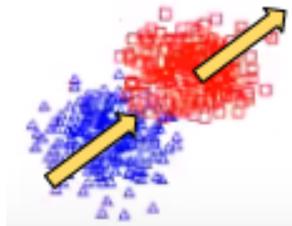


Figure 4.2: Bi-dimensional two classes with diagonal transitioning. Data distribution evolution through the time (yellow arrow).

The Figure 4.3a illustrates the accuracy of the classifiers over this dataset. Between the batches 0 and 10 a translational drift occurs on data. At this point, samples from the two classes are changing in diagonal, where the first class is changing to the region of the second class and the second class moving to the same direction. Hence, both static and incremental classifiers start to decrease their accuracy. Once the static classifier remains with the same classification hypothesis, misclassification occurs for the samples that are gradually changing from one class region to the other.

The incremental classifier, however, does not discard outdated samples and learns with data that contains old concepts, hindering the drift detection in data distribution. After the tenth batch, almost all data of a particular class shifts to the region of the another class. After interval 90, both classifiers increase their accuracy due to the fact that data returns to a same state as the initial batch. However, LEVELiw does not recover the efficiency since at this point, unlikely static classifier, the method has not the points that represents the original concept. Besides, LEVELiw seems to be hindered by the previous change in data and now it classifies only one class, achieving a similar result of a random classifier.

The three methods that use a CSE method to extract core instances, COMPOSE-GMM, AMANDA-FCP and AMANDA-DCP, were able to select core instances and continuously capture the gradual change of data. Therefore, they obtained the most accurate results along with the Sliding Window classifier. In figure 4.3b, we

illustrate the computational time of all classifiers. It should be noted that LEVELiw, Incremental SSL and COMPOSE-GMM present higher computational time.

For LEVELiw, the reason is that it has to process a Least Squares Importance Fitting and a importance weighted cross validation for all batches that both have high computational time. For COMPOSE-GMM, the reason is that it needs to learn a set of Gaussian mixture models for each batch of data, that has a high computational time. The same issue occurs for the Incremental, that has a higher computational time due to the to continuous growth of samples in the training set. Finally, regarding AMANDA-FCP and AMANDA-DCP, their processing time were similar to Static and Sliding SSL classifiers. This is an interesting result, once our proposals are more accurate than the two classifiers.

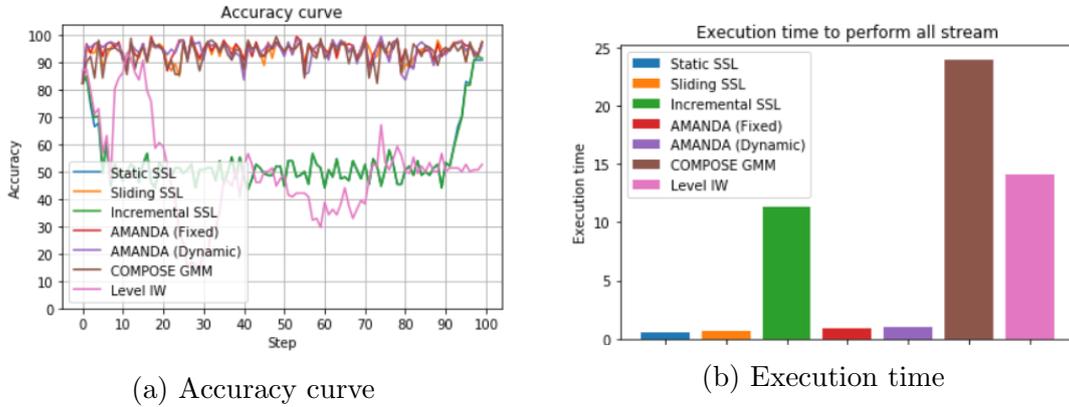
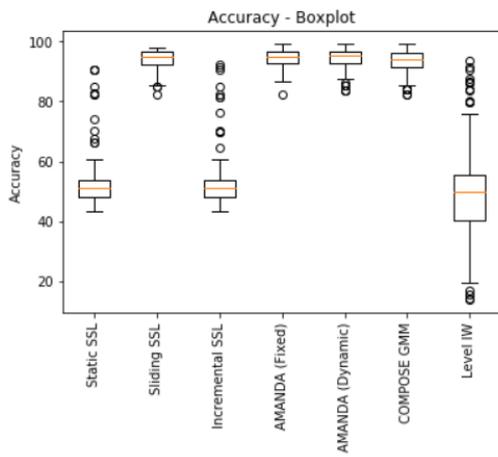


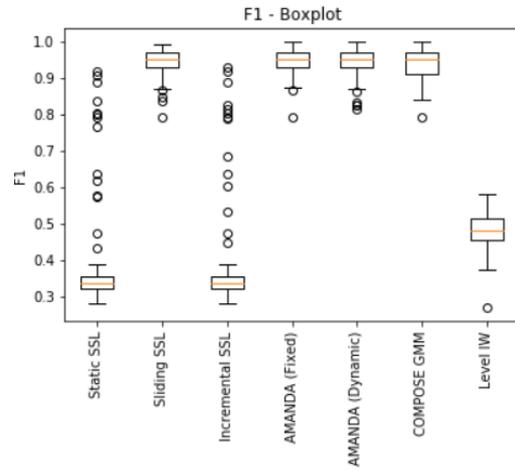
Figure 4.3: Bi-dimensional two classes with diagonal transitioning

In the figures 4.4a and 4.4b, we show the accuracy and macro-F1 results, respectively. AMANDA-DCP AMANDA-FCP, COMPOSE-GMM and Sliding SSL presented similar accuracy and macro-f1 results. However, according to the boxplots, AMANDA-FCP presented less variance and less outliers on its results. Static SSL and Incremental SSL have the worst results, with the lowest averages and with the highest number of outliers. LEVELiw also had low accuracy and f1 results. This is due to the difficulty to capture the drift on data from the interval 20. The consequence was the classification of only one class, decreasing the F1 results.

In Figure 4.5, we show the error reduction percentage over the static classifier. All classifiers except incremental and LEVELiw obtained similar results and superior than the static classifier. However, since this dataset has a drift characteristic easily captured by classifiers, the error reduction was low. The incremental classifier, as previously explained, obtained the same accuracy than static classifier. LEVELiw obtained 0.06% less accuracy than the static classifier.



(a) Accuracy boxplot.



(b) F1 boxplot.

Figure 4.4: Bi-dimensional two classes with diagonal transitioning

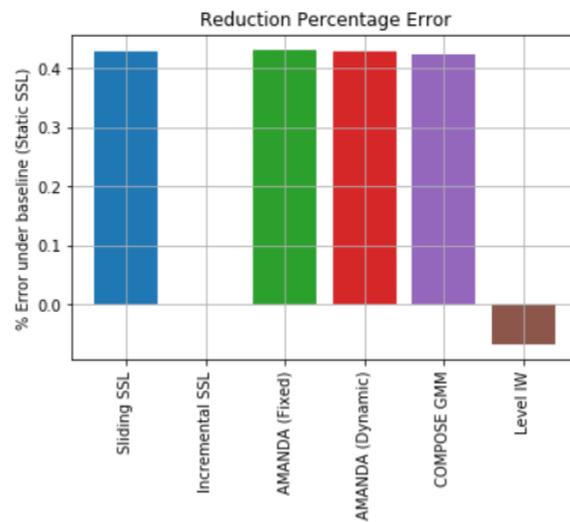


Figure 4.5: Error reduction over static classifier - Bi-dimensional two classes with diagonal transitioning.

#### 4.4.4 Artificial Dataset - 1CSURR

Figure 4.6 illustrates the evolution regarding time for 1CSURR, an imbalanced dataset with a majority class surrounding a minority class. The class proportion of this dataset is 60/40 and it contains two dimensions. This is a challenging dataset due to the classification bounds, that are near each other.

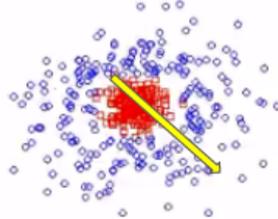


Figure 4.6: 1CSURR data distribution evolution through the time (yellow arrow).

The Figure 4.10a illustrates the accuracy of the classifiers over this dataset. Between the batches 0 and 10, a gradual drift occurs at data. Thus, samples from one class are changing to the original region of the other class. Then, it should be noted that both, static and incremental classifiers, start to decrease their accuracy. Once the static classifier remains with the same classification hypothesis, misclassification occurs for the samples that are gradually changing from one class region to the other.

The incremental classifier, however, does not discard outdated samples and has a different reason for its decrease on performance. Once samples from one class gradually change to the region of the outdated samples of the another class, the decision region of the incremental classifier remains dense with samples from both classes. Moreover, the incremental classifier learns with data that contains old concepts, hindering the drift detection in data distribution. After the tenth batch, almost all data of a particular class shifts to the region of the another class.

After interval 70, both classifiers present a growing accuracy curve due to the fact that the dataset returns to a similar data distribution as the initial batch. Then, between batches 85 and 95, both classifiers, static and incremental, has a decrease on their performance due to a critical drift between the majority and minority classes. This critical change occurs when the minority class is surrounded by the majority one. At this moment, the static classifier presents the same problem of initial batches: The use of the initial classification hypothesis. The incremental classifier, is also negatively affected since all data of the decision boundary of the classes are kept. Hence, the overlapping between the classes and the old concepts present in train data, hinders the classification.

The AMANDA-FCP, obtained satisfactory results due to the cutting method that selects the most relevant instances from dense regions of data. This selection

was important to keep the instances that influence the classifier at the moment that the minority class is surrounded by the majority class, specifically between batches 85 and 95. Worth to mention that selecting fewer and representative instances is better than keeping all instances at the moment of the minority class is surrounded. This occurs since a few instances in decision boundary fool the classifier, specially for lower dimensions. The three methods that uses a CSE method to extract the best instances, COMPOSE-GMM and AMANDA-FCP and AMANDA-DCP, obtained the best results and the best recoverability between batches 85 and 95.

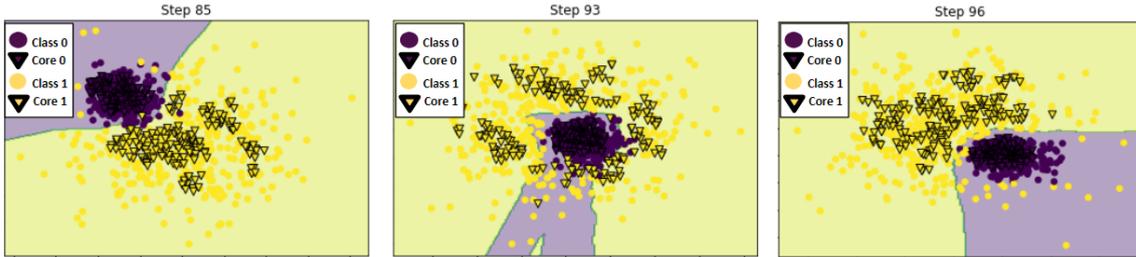


Figure 4.7: AMANDA-FCP Core Support Extraction Performance in 1CSURR.

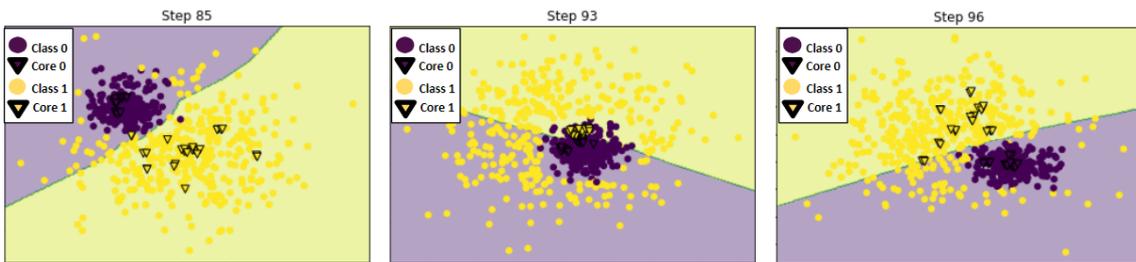


Figure 4.8: AMANDA-DCP Core Support Extraction Performance in 1CSURR.

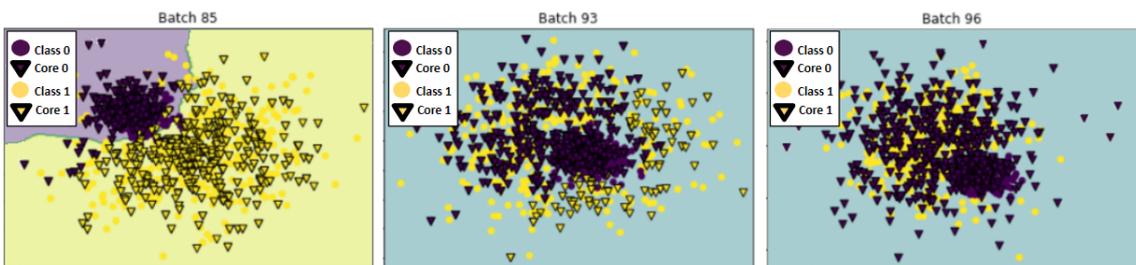


Figure 4.9: COMPOSE-GMM Core Support Extraction Performance in 1CSURR.

Figure 4.7 and 4.8 illustrates the behaviour of the AMANDA-FCP and AMANDA-DCP, respectively, whereas Figure 4.9 illustrates the behaviour of COMPOSE-GMM, over the critical batches. The three methods use a CSE approach to extract the best instances and for this reason they obtained the best results and the best recoverability between batches 85 and 95. However, AMANDA-FCP obtained the best result among them due to the cutting method that selects instances from dense

regions of data better than the other two methods. This selection was important to keep the instances that influence the classifier at the moment that the minority class is entirely surrounded by the majority class, specifically for the batch 93.

In figure 4.10b, we illustrate the classifiers performance for this dataset. The computational time results are similar to the results indicated by the previous experiment and the reasons are the same.

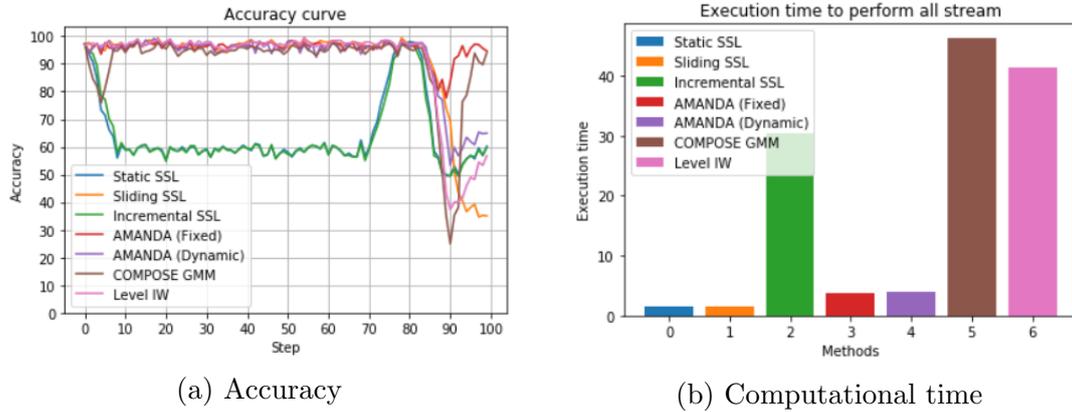


Figure 4.10: One class surrounding another (1CSURR)

In the figures 4.11a and 4.11b, we show the MCC and macro-F1 results, respectively. Both, AMANDA-DCP and FCP present similar macro-f1 results compared COMPOSE-GMM and Sliding SSL. However, AMANDA-FCP presents higher results and less outliers. Hence, AMANDA-FCP seems to be a better method regarding macro-f1. LEVELiw had the worst MCC and F1 results regarding the state-of-the-art due to surrounding-class properties.

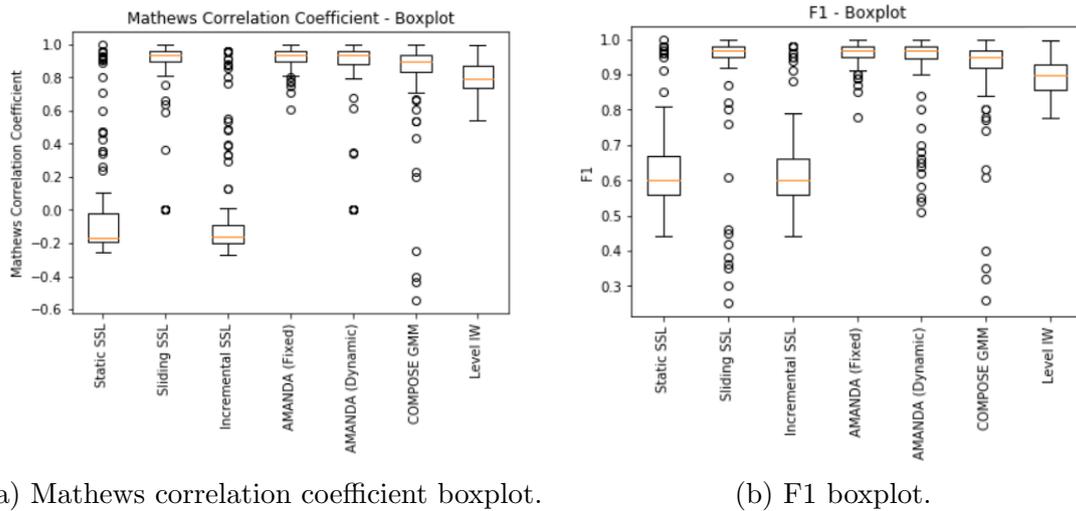


Figure 4.11: One class surrounding another

Figure 4.12 illustrates the error reduction over the static classifier. The sliding window classifier had a similar efficiency in comparison with the state-of-the-art.

However, this classifier has the advantage to be simpler than the state-of-the-art algorithms. The AMANDA-DCP achieves more than 30% of improvement whereas AMANDA-FCP is approximately 5% more assertive than all other state-of-the-art methods. As previously detailed, the incremental classifier obtained the same average error than the baseline.

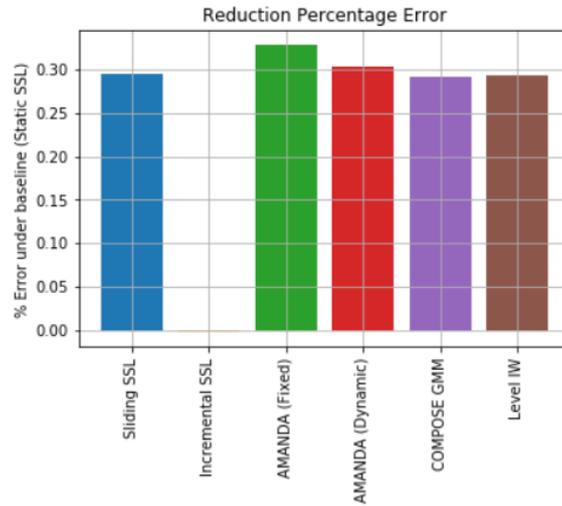


Figure 4.12: Error reduction over static model - One class surrounding another.

#### 4.4.5 Artificial Dataset - 4CRE\_V1

This is the bi-dimensional four class rotating and expanding dataset. This dataset is balanced and has as a main characteristic a fast translational and rotational drift, as illustrated in Figure 4.13. Its a challenging task since data distribution changes very fast through the time. Thus, adapting to these changes becomes increasingly difficult in the subsequent batches.

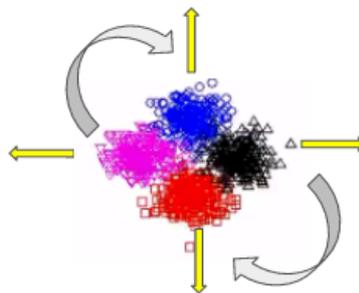


Figure 4.13: bi-dimensional four class rotating and expanding. Data distribution evolution through the time (yellow arrow).

Figure 4.14a shows the accuracy curve achieved by all methods. All classifiers presented lower results and similar to the Static Classifier. However, even in this

adverse scenario, AMANDA-FCP obtained stable results between the interval 75 and 100. At the same interval, the state-of-the-art classifiers alternated regarding the predictions' quality.

As illustrated in Figure 4.14b, the incremental classifier, COMPOSE-GMM and LEVELiw had the higher processing times. AMANDA-FCP obtained lower computational time than the other state-of-the-art methods. The processing time of the COMPOSE-GMM and LEVELiw were higher compared to AMANDA-DCP. AMANDA-DCP processing time reduces markedly for high dimensionality of data. Once this dataset has only two dimensions, AMANDA-DCP performs equally to AMANDA-FCP. In contrast, COMPOSE-GMM and LEVELiw have difficulty for fast processing when the cardinality of data is high, even when data has low dimensionality, as previously explained.

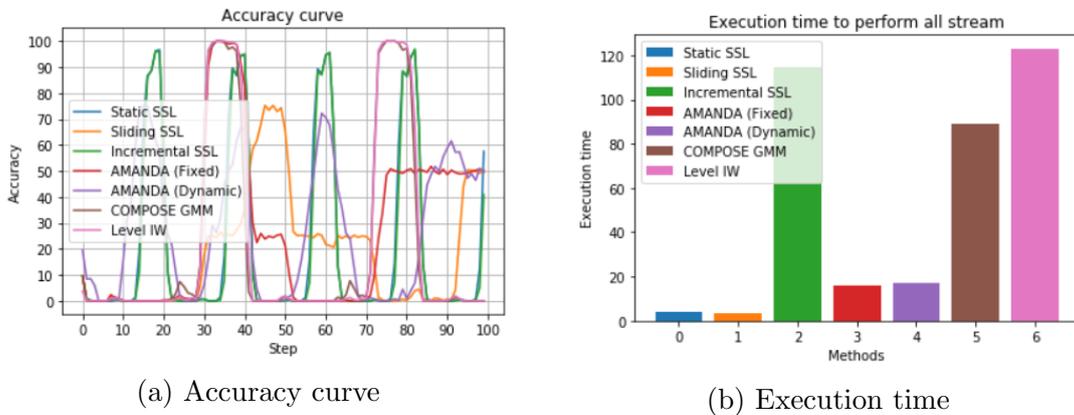
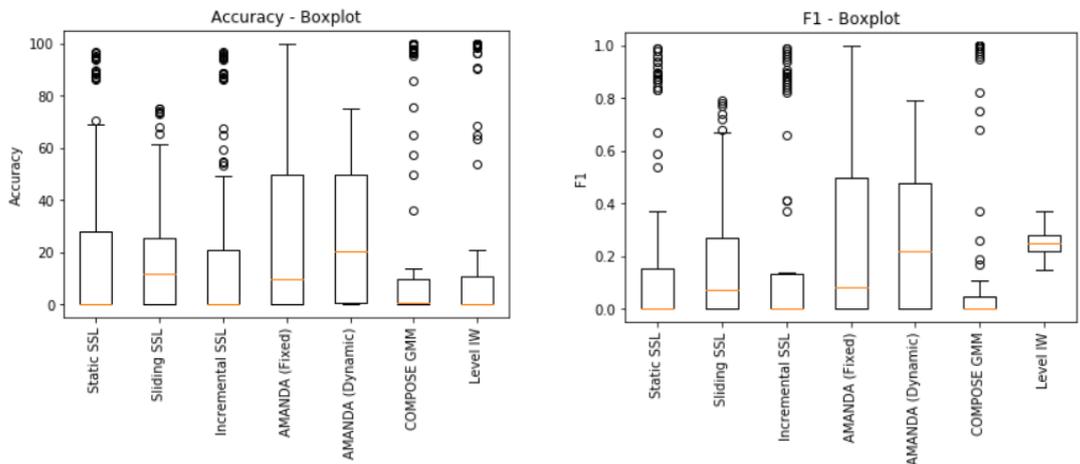


Figure 4.14: Four class rotating and expanding (Version 1)

As shown in Figure 4.15a, the classifiers were not able to achieve reliable results. This is due to the inefficiency to detect the drift. Static, Incremental, COMPOSE-GMM and LEVELiw classifiers obtained 50% of mistakes. The classifiers were hindered each time the dataset expanded. Figure 4.15b shows the overall quality of the predictions. AMANDA-FCP obtained the highest macro F1, but the best results were obtained by AMANDA-DCP.

In the figure 4.16, we note that only AMANDA-DCP and AMANDA-FCP have overcome the Static Classifier. The remaining algorithms obtained errors greater than the baseline. This indicates that the CSE improve the predictions' quality.



(a) Accuracy boxplot.

(b) F1 boxplot.

Figure 4.15: Four class rotating and expanding (Version 1)

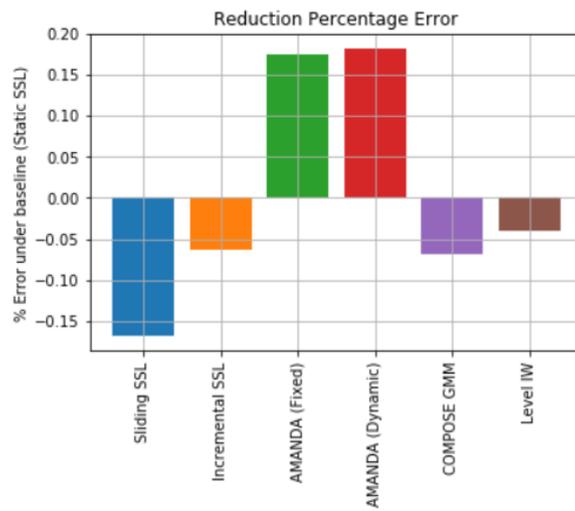


Figure 4.16: Error reduction over static model - Four class rotating and expanding (Version 1).

#### 4.4.6 Real Dataset - Keystroke

Keystroke is a balanced dataset with four classes and ten dimensions. The dataset is divided into eight batches that correspond to the number of times that users typed their passwords, as explained in the subsection 4.2. We expect a gradual drift of data distribution due to the users repeatedly type their same passwords. The Figure 4.17a illustrates the accuracy of the classifiers. From the second batch until the sixth, the static classifier had its prediction quality decreased. This behaviour is due to the presence of gradual drift in data. In the subsequent batches, the static classifier had a similar performance of a random classifier. The remaining classifiers obtained similar performance. However, AMANDA-FCP and AMANDA-DCP had the higher values in classification accuracy.

As illustrated in Figure 4.17b, the processing time of the COMPOSE-GMM is almost five times higher than AMANDA-FCP. The reason is that even though the number of batches and instances is low, the number of dimensions makes the processing of COMPOSE-GMM become slow. The number of dimensions is also the reason for the best processing time of AMANDA-DCP. The value was six times higher than COMPOSE-GMM. AMANDA-DCP has the disadvantage to perform the multi-dimensional Hellinger distance beyond the KDE processing.

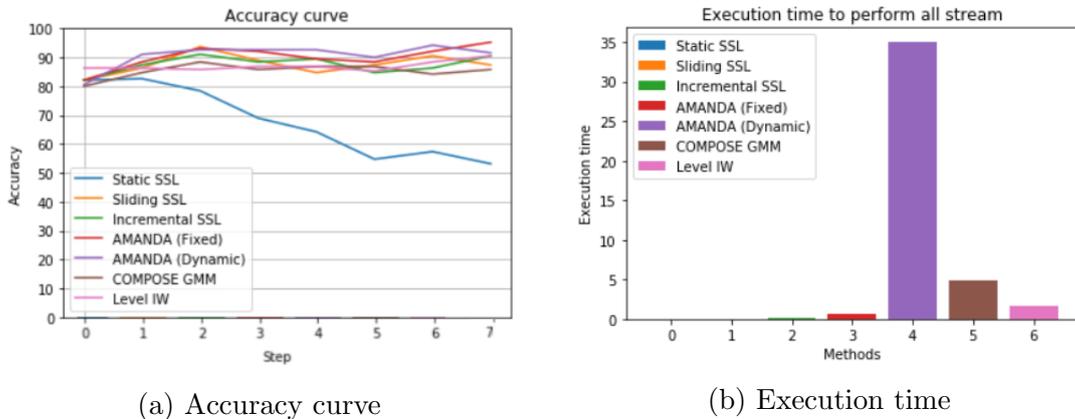
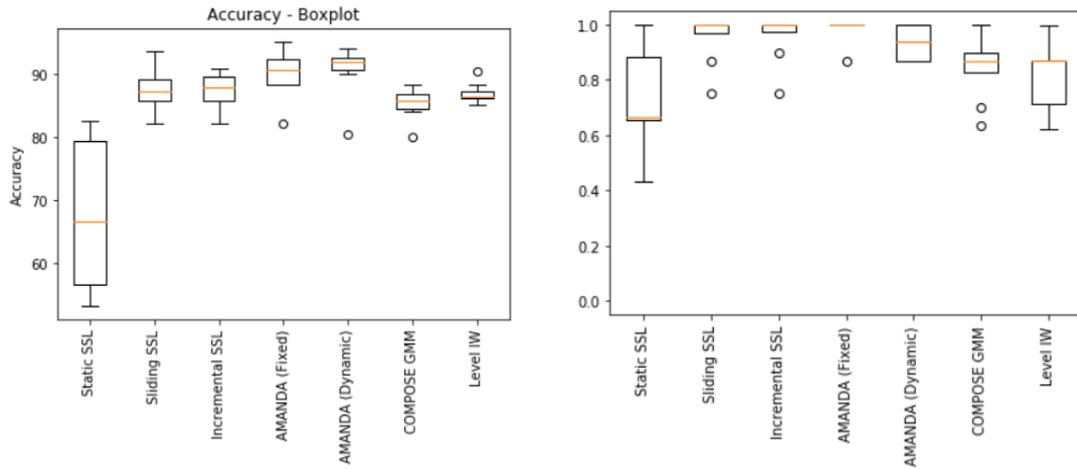


Figure 4.17: Keystroke results

Figure 4.18a shows the accuracy boxplot of all classifiers. It corroborates that AMANDA-FCP and AMANDA-DCP had higher values regarding accuracy. Besides, AMANDA-DCP, COMPOSE-GMM and LEVELIw have the most stable results. However, as illustrated in Figure 4.18b, AMANDA-FCP obtained the best F1 result. Thus, even the AMANDA-DCP obtained the best average accuracy, AMANDA-FCP had better quality between the classes. The Incremental classifier had the second best result due to the dataset characteristic that is gradual and has strong temporal dependency. Sliding window classifier had similar results to the incremental classifier, that is, the choice of SSL is more determinant for the results.



(a) Accuracy boxplot.

(b) F1 boxplot.

Figure 4.18: Keystroke results

Figure 4.19 illustrates the percentage reduction of error over the static classifier. AMANDA-DCP has the lower average error, 34% whereas AMANDA-FCP has the second lower average error 33%. Besides, even COMPOSE-GMM and LEVELiw have more than 25% of improvement over the static classifier, their results are worse than the incremental and sliding classifiers. COMPOSE-GMM worst results are due to GMM function that was able to select the most representative instances as KDE performed for AMANDA methods. LEVELiw was a bit better than COMPOSE-GMM but the unconstrained importance least square function does not capture the drift between the first and sixth batches. Hence, the average error was compromised and worse than the other classifiers.

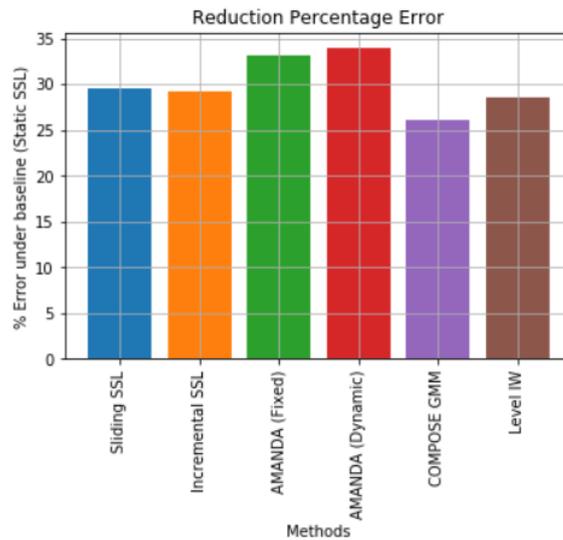


Figure 4.19: Error reduction over static model - Keystroke dataset.

#### 4.4.7 Real Dataset - NOAA

NOAA is an imbalanced dataset with a class proportion of 60/40 with eight dimensions and 50 batches, each batch representing a year. Figure 4.20a illustrates the accuracy curve of all classifiers. The classifiers results are close with an exception of LEVELiw, that presented its results 2% lower than the static classifier.

This small discrepancy between the remaining classifiers indicates that the Label Propagation classifier, that is used in the remaining algorithms, was better suited than the least square classifier used by LEVELiw. The only advantage of CSE on this dataset is for AMANDA-FCP, that had an increase on its performance over the other classifiers, between the 33th and 37th batches. Even occurring close results for all classifiers, the Figure 4.20 shows a drawback of AMANDA-DCP. The method had its processing time highly increased as the dimension of data grew.

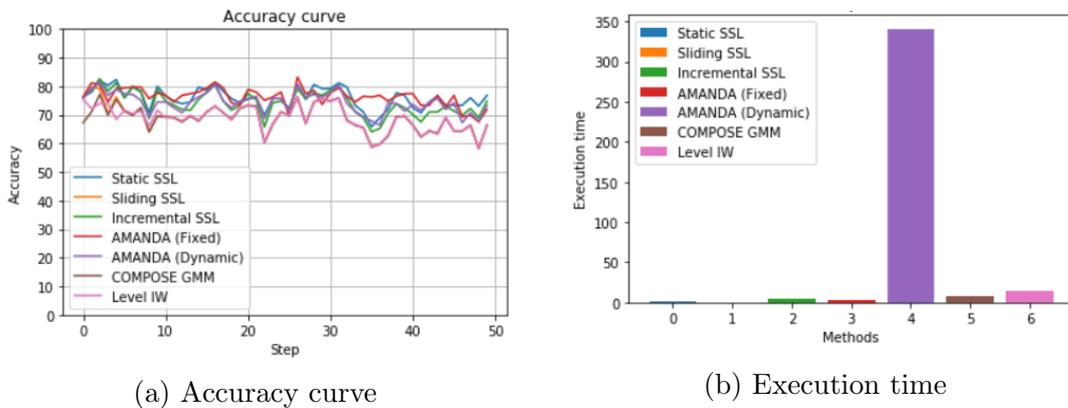
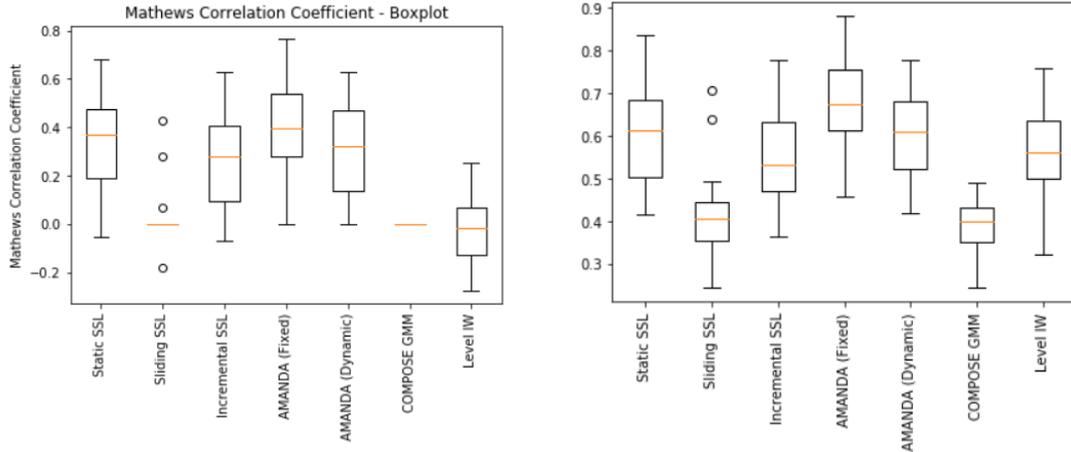


Figure 4.20: NOAA results

In Figure 4.21a, we show MCC boxplots of all classifiers. The results indicate that COMPOSE-GMM presented similar results compared to a Sliding Classifier. This lead us to conclude that the approach of using all instances of a recent batch is as efficient as the selection of the instances of the same batch using GMM. However, both classifiers is as efficient as a random classifier indicating that the two approaches tends to classify all instances as one class. On the other hand, AMANDA-FCP had the best results due to the gradual forgetting approach of the CSE algorithm.

LEVELiw had the worst results since this algorithm considers only the last batch of data. Thus, it does not take advantage of the temporal dependence of data. The Sliding Window Classifier has the same approach of LEVELiw regarding the use of last batch of data. However, the K-NN classifier contained in the Sliding Window Classifier is more efficient than the Gaussian Kernel function that compute posterior probability contained in LEVELiw.

Figure 4.21b shows F1 results for all classifiers. The results are similar to the MCC results illustrated in figure 4.21a. However, LEVELiw MCC results are lower



(a) Mathews correlation coefficient boxplot.

(b) F1 boxplot.

Figure 4.21: NOAA results

compared to its F1 results. This indicates that its results are favoring the majority class and are not correlated to the labels.

Regarding error reduction, illustrated in Figure 4.22, the Sliding Window and the COMPOSE-GMM results indicate that discarding old instances is more harmful than keeping all instances such as the Incremental Classifier performs. The reason is that this dataset has a small and recurrent drift. LEVELiw had similar results than COMPOSE and Sliding Window classifiers due to the difficulties of the uLSIF algorithm, contained in LEVELiw, to adapt to upcoming data. Hence, taking into account the characteristics of this dataset, the choice of a SSL classifier has more impact than the use of CSE method, since AMANDA-FCP was only 1% better than the static classifier. AMANDA-DCP and Incremental Classifier also performed worse than a Static Classifier. Despite both classifiers have similar average errors, the Incremental Classifier is faster than AMANDA-DCP.

#### 4.4.8 Real Dataset - Electricity

Electricity is an imbalanced dataset, divided into 100 batches with two classes and five dimensions. This dataset is challenging since there are several drifts inside each batch. Figure 4.23a illustrates the accuracy curve for all classifiers. Between the interval 0 and 55, all methods perform similarly to each other, alternating between 60% and 90% of accuracy. After that, the classifiers are divided into two groups: The first group achieves more than 60% of accuracy and the second group achieves less than 60% of accuracy.

Moreover, the performance of COMPOSE-GMM and LEVELiw increases while the performance of sliding window classifier and AMANDA-DCP decreases. Besides, from batch 70 to the last batch, AMANDA-DCP and the sliding window classifier

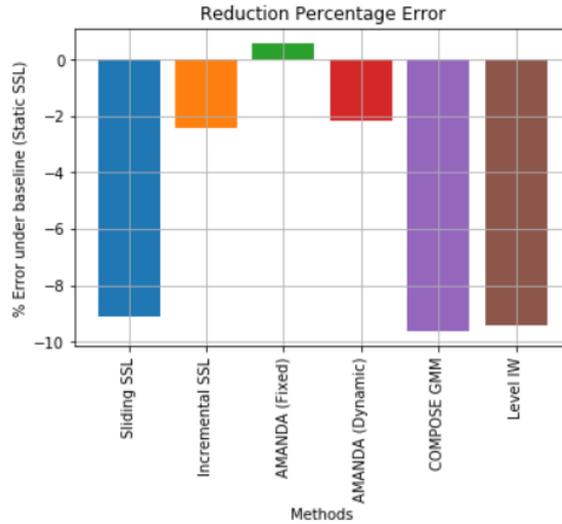
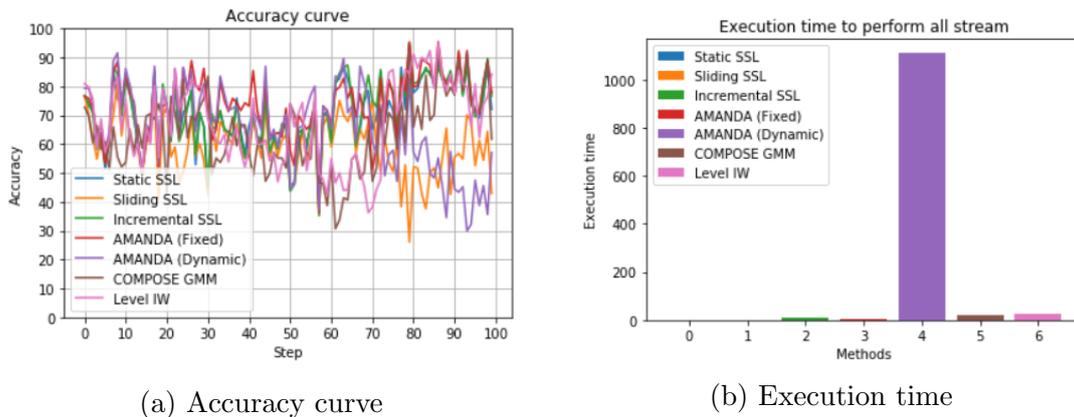


Figure 4.22: Error reduction over static model - NOAA dataset.

predicts the inverse of each other. Figure 4.23b illustrates the processing time results. It should be noted that all classifiers except AMANDA-DCP obtained faster processing time results. The reason, again, is due to the high dimensionality of the data.



(a) Accuracy curve

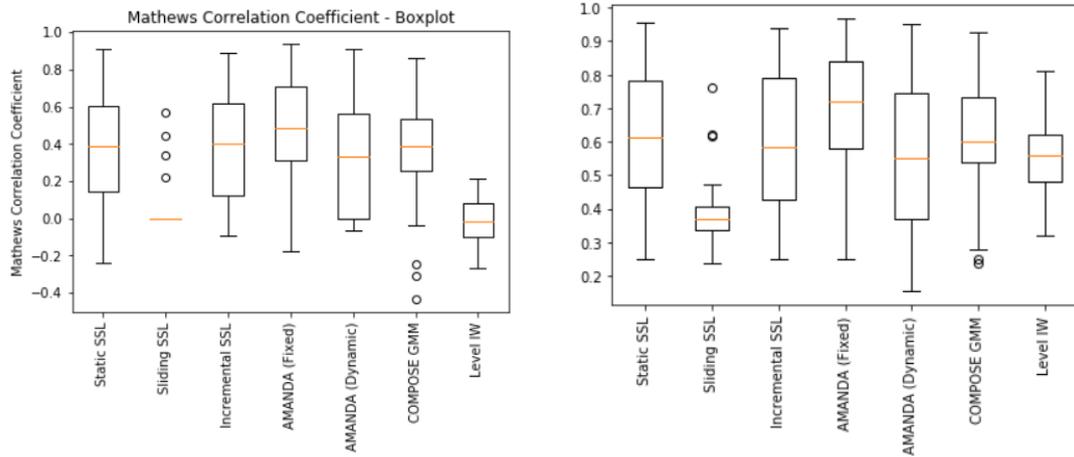
(b) Execution time

Figure 4.23: Electricity dataset results

Figure 4.24a shows the MCC boxplot for all classifiers. Worth to mention that the Sliding Window Classifier had a similar performance of a random classifier. Since the Sliding Window Classifier forgets past data, the classifier was hindered due to the temporal dependency of dataset. For this reason, static and incremental classifiers obtained most accurate results than sliding window classifier. LEVELIw also uses the last batch of data, the classifier applied in the LEVELIw does not learn with a new distribution.

Figure 4.24b shows that AMANDA-FCP has the most accurate F1 results. The gradual forgetting of AMANDA-FCP captures the temporal information of the dataset. COMPOSE-GMM and AMANDA-DCP also have better results than

Sliding Window Classifier due to the CSE in the method. However, the quantity of kept instances of these two methods are not enough to learn the new concept. On the other hand, AMANDA-FCP is better on this aspect.



(a) Mathews correlation coefficient boxplot.

(b) F1 boxplot.

Figure 4.24: Electricity dataset results

Figure 4.25 illustrates the reduction of percentage error over the static classifier. It should be noted that only AMANDA-FCP had better results than the static classifier. As previously explained, the temporal dependency of dataset decreases the performance of the Sliding Window Classifier. The number of core instances kept by AMANDA-DCP and COMPOSE-GMM also hinders their performances.

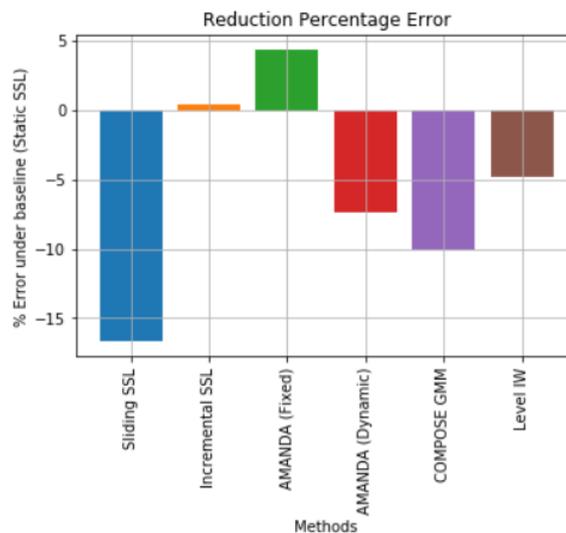


Figure 4.25: Error reduction over static model - Electricity dataset.

## 4.4.9 Overall Results

Aiming to visualize all the information in one table, we renamed the classifiers using more compact acronyms. Therefore, the acronyms of the classifiers are: Static classifier (STC), sliding window classifier (SLD), incremental classifier (INC), COMPOSE-GMM (CMP), LEVELiw (LVL), and the two proposals, AMANDA-FCP (A-FCP) and AMANDA-DCP (A-DCP). The average error results of all methods for all datasets are shown in Table 4.4. The reduction of the percentage error over the static classifier is presented between parentheses.

Table 4.4: Average error results.

Datasets	STC	SLD	INC	CMP	LVL	A-FCP	A-DCP
1CDT	0.76	0.06 (-92.10%)	0.30 (-60.52%)	0.08 (-89.47%)	0.04 (-94.73%)	<b>0.02 (-97.36%)</b>	0.05 (-93.42%)
1CHT	3.93	0.43 (-89.05%)	3.20 (-18.57%)	0.48 (-87.78%)	0.40 (-89.92%)	<b>0.33 (-91.60%)</b>	0.39 (-90.07%)
2CDT	46.30	6.13 (-86.76%)	46.14 (-0.34%)	6.73 (-85.46%)	49.74 (+7.42%)	<b>5.46 (-88.20%)</b>	5.83 (-87.40%)
2CHT	45.97	48.45 (+5.39%)	46.01 (+0.08%)	47.41 (+3.13%)	47.41(+3.13%)	<b>14.39 (-68.69%)</b>	19.93 (-56.64%)
4CRT	78.83	<b>0.02 (-99.97%)</b>	78.75 (-0.10%)	0.04 (-99.94%)	<b>0.02(-99.97%)</b>	<b>0.02 (-99.97%)</b>	0.03 (-99.96%)
4CRE-V1	78.15	81.29 (+4.01%)	79.44 (+1.65%)	79.55 (+1.79%)	79.00 (+1.08%)	73.50 (-5.95%)	<b>73.28 (-6.23%)</b>
4CRE-V2	79.61	82.88 (+4.10%)	79.67 (+0.07%)	77.38 (-10.91%)	80.77 (+1.45%)	<b>69.97 (-12.10%)</b>	71.81 (-9.79%)
5CVT	54.51	60.97 (+11.81%)	52.04 (-4.53%)	65.50 (+24.16%)	59.18 (+8.56%)	<b>24.11 (-55.76%)</b>	52.38 (-3.90%)
1CSURR	35.86	9.05 (-74.76%)	36.06 (+0.55%)	9.43 (-73.70%)	9.20 (-74.34%)	<b>4.39 (-87.75%)</b>	7.93 (-77.88%)
4CE1CF	1.98	1.90 (-4.04%)	1.82 (-8.08%)	2.09 (+5.55%)	2.21 (+11.61%)	<b>1.73 (-12.62%)</b>	1.92 (-3.03%)
UG2C2D	55.81	4.97 (-91.09%)	54.42 (-2.49%)	5.32 (-90.46%)	26.34 (-52.80%)	<b>4.30 (-92.29%)</b>	12.64 (-77.35%)
MG2C2D	51.63	22.86 (-55.72%)	50.66 (-1.87%)	49.17 (-4.76%)	9.31 (-81.96%)	<b>8.70 (-83.14%)</b>	14.88 (-71.17%)
FG2C2D	17.79	4.43 (-75.09%)	18.29 (+2.81%)	12.15 (-31.70%)	<b>4.31 (-75.77%)</b>	5.12 (-71.21%)	16.39 (-7.86%)
UG2C5D	30.97	20.11 (-35.06%)	30.62 (-1.13%)	20.82 (-32.77%)	20.82 (-32.77%)	<b>8.21 (-73.49%)</b>	8.53 (-72.45%)
GEARS	5.43	<b>0.81 (-85.08%)</b>	5.33 (-1.84%)	4.03 (-25.78%)	6.18 (+13.81%)	<b>0.81 (-85.08%)</b>	3.74 (-31.12%)
BOARD	51.08	49.96 (-2.19%)	50.98 (-0.19%)	<b>49.73 (-2.64%)</b>	49.73 (-2.64%)	50.11 (-1.89%)	49.98 (-2.15%)
ELEC2	29.22	41.04 (+16.69%)	28.92 (-0.41%)	36.31 (+10.02%)	32.64 (+4.83%)	<b>26.12 (-4.37%)</b>	34.45 (+7.38%)
NOAA	24.07	30.97 (+9.09%)	25.92 (+2.42%)	31.40 (+9.64%)	31.25 (+9.44%)	<b>23.62 (-0.59%)</b>	25.71 (+2.15%)
KEYSTROKE	32.30	12.37 (-29.44%)	12.50 (-29.25%)	14.67 (-26.04%)	12.97 (-28.56%)	9.87 (-33.13%)	<b>9.34 (-33.91%)</b>

It should be noted that AMANDA-FCP obtained the most accurate results for fourteen datasets among the seventeen artificial datasets and the most accurate results for two of the three real datasets. Besides, AMANDA-DCP had the best results in one artificial dataset and one real dataset. Hence, the two variations of AMANDA together presented better results than the other classifiers for nineteen of twenty datasets. COMPOSE-GMM along with LEVELiw, had the best result for a dataset that AMANDA variations did not won: Rotational checkerboard dataset.

LEVELiw obtained the best results for two artificial datasets. In addition, in all cases where LEVELiw presented inferior results to COMPOSE, the dataset had experienced drifting posterior probabilities, a violation of the primary assumption of importance sampling approaches (UMER *et al.*, 2017). Table 4.5 shows all macro-F1 results. AMANDA-FCP had the most accurate results for thirteen datasets among seventeen artificial datasets. Besides, AMANDA-FCP obtained the best results for all real datasets. AMANDA-DCP had the best results for two artificial datasets. COMPOSE-GMM, LEVELiw, Incremental and Sliding Window classifiers had best results for one artificial dataset.

Datasets	STC	SLD	INC	CMP	LVL	A-FCP	A-DCP
1CDT	0.9935	0.9994	0.9971	0.9995	0.9996	<b>0.9997</b>	0.9994
1CHT	0.9600	0.9950	0.9681	0.9949	0.9960	<b>0.9963</b>	0.9955
2CDT	0.3871	0.9418	0.3884	0.9362	0.4836	<b>0.9480</b>	0.9416
2CHT	0.3954	0.3560	0.3942	0.4758	0.4758	<b>0.8526</b>	0.7880
4CRT	0.2099	0.9998	0.2154	<b>0.9999</b>	0.9998	0.9998	<b>0.9999</b>
4CRE-V1	0.2073	0.1804	0.1997	0.2035	0.2486	<b>0.2670</b>	0.2651
4CRE-V2	0.2043	0.1259	0.2039	0.1971	0.2464	<b>0.3035</b>	0.1810
5CVT	0.3537	0.1812	0.3707	0.2385	0.1767	<b>0.7297</b>	0.3802
1CSURR	0.6403	0.9137	0.6384	0.9094	0.6368	<b>0.9607</b>	0.9267
4CE1CF	0.9807	0.9795	<b>0.9821</b>	0.9781	0.9779	0.9808	0.9803
UG2C2D	0.4425	0.9514	0.4546	0.9491	0.7366	<b>0.9581</b>	0.8706
MG2C2D	0.4795	0.7543	0.4936	0.5050	0.5923	<b>0.9143</b>	0.8499
FG2C2D	0.7322	0.9391	0.7298	0.8596	<b>0.9469</b>	0.9319	0.8190
UG2C3D	0.5046	0.9245	0.4916	0.9217	0.6032	<b>0.9461</b>	0.9426
UC2C5D	0.6680	0.7549	0.6782	0.7918	0.7918	<b>0.9151</b>	0.9129
GEARS	0.9474	<b>0.9957</b>	0.9485	0.9637	0.9382	<b>0.9957</b>	0.9630
BOARD	0.4983	0.4901	0.4957	0.4956	0.4956	0.4841	<b>0.4985</b>
ELEC2	0.6165	0.3740	0.6095	0.6230	0.4469	<b>0.6900</b>	0.5548
NOAA	0.6041	0.4060	0.5509	0.3907	0.4745	<b>0.6790</b>	0.6029
KEYSTROKE	0.7266	0.9520	0.9562	0.85	0.7875	<b>0.9833</b>	0.9351

Table 4.5: Macro-F1 results.

Table 4.6 shows the processing time of all methods datasets. It should be noted that the Static and Sliding Window classifiers are faster than the other methods. Static classifier learns its own model only once and perform classification in the subsequent batches. Sliding Window only retrain using the last batch and classifies the current samples, the quantity of instances for training and classification is fixed. Therefore, we put the marker ‘\*’ indicating the best processing time between the proposal and state-of-the-art algorithms. AMANDA-FCP obtained the fastest processing time for eight artificial datasets and all real datasets in comparison with COMPOSE-GMM and LEVELiw, that are the state-of-the-art classifiers. AMANDA-DCP obtained the fastest processing time among the state-of-the-art classifiers for seven artificial datasets. LEVELiw obtained the best results among the state-of-the-art in three artificial datasets.

LEVELiw and COMPOSE-GMM had more difficulties to keep low processing time since these methods perform high cost algorithms to classify data. As explained in Section 2.2, COMPOSE-GMM performs several times the EM algorithm for each

batch. Besides, this method applies a distance calculation along with a Bayesian criteria information method. On the other hand, LEVELiw applies a combination of quadratic algorithms for each batch of data. Thus, the algorithm becomes slow in an iterative scenario, since this algorithm was originally built for one-shot learning way.

Worth to mention that, AMANDA-DCP had the worst processing time among the state-of-the-art algorithms in NOAA and ELEC2 datasets. The reason is the calculation of the Hellinger distance for each batch of data. The processing time of this calculation grows according with the quantity of dimensions and the quantity of instances. Despite the high processing time in NOAA and ELEC2 datasets, the AMANDA-DCP method has a low processing time in Keystroke dataset with ten dimensions. For this reason, AMANDA-DCP has poor results when the data has a combination of high dimensions and vast amount of data.

<b>Datasets</b>	<b>STC</b>	<b>SLD</b>	<b>INC</b>	<b>CMP</b>	<b>LVL</b>	<b>A-FCP</b>	<b>A-DCP</b>
1CDT	0.44	0.40	4.90	23.68	1.72	0.83	0.81*
1CHT	0.50	0.49	7.08	24.07	25.94	1.07	0.80*
2CDT	0.57	0.71	11.30	23.96	14.81	0.94*	0.98
2CHT	0.93	0.14	4.31	17.32	14.77	0.55*	0.62
4CRT	4.26	2.97	148.12	79.25	14.77*	20.62	21.57
4CRE-V1	3.82	3.18	114.82	88.99	14.34*	15.97	17.40
4CRE-V2	5.14	4.41	177.64	70.90	14.30*	30.38	35.12
5CVT	0.76	0.67	14.58	79.12	156.46	1.68	1.10*
1CSURR	1.62	1.44	30.46	46.34	123.08	3.66*	3.97
4CE1CF	97.15	10.19	499.34	128.50	213.72	34.45*	44.28
UG2C2D	3.25	2.91	111.52	58.17	25.24	11.77*	12.89
MG2C2D	5.79	5.02	191.34	55.06	41.35	37.35	34.72*
FG2C2D	5.22	5.39	197.09	60.06	187.92	31.34*	33.58
UG2C3D	11.97	5.96	220.57	88.06	95.47	40.63*	41.20
UC2C5D	125.77	12.43	763.18	69.43	229.52	50.78	34.35*
GEARS	6.37	6.35	270.16	50.40	230.34	42.07	35.93*
BOARD	3.76	3.25	99.51	47.23	–	28.63*	28.63*
ELEC2	1.40	0.16	10.11	21.14	236.78	2.82*	1276.07
NOAA	0.64	0.13	4.11	8.28	230.24	2.87*	407.33
KEYSTROKE	0.01	0.01	0.03	4.83	230.53	0.33*	22.77

Table 4.6: Processing time results.

# Chapter 5

## Conclusions

In this chapter, we presented our final considerations, a summary of the results, our main contributions, limitations of the proposal and research directions.

### 5.1 Proposal

In this work, we investigated gradual concept-drift and extreme verification latency problems for non-stationary environments. State-of-the-art methods do not tackle properly the mentioned problems due to the following issues: (a) their high computational time for large volume of data; (b) their lack of representing the right samples of the drift or (c) even for having several parameters for tuning.

Therefore, we proposed AMANDA, a density-based adaptive model for non-stationary data, using a SSL along with a density-based CSE method. AMANDA has two variations: AMANDA with a fixed cutting percentage (AMANDA-FCP); and AMANDA with a dynamic cutting percentage (AMANDA-DCP). Both variations improve the mentioned issues (a) and (b) and, the last one, improves the issue (c).

### 5.2 Summary of Results

Aiming to evaluate our proposal, we applied the AMANDA-FCP and AMANDA-DCP in seventeen synthetic datasets and three real ones: Keystroke, Electricity Market and NOAA. The parameters were chosen through the use of an optimization method along with a prequential evaluation with 5% of data for each dataset.

Our results indicate that AMANDA-FCP, outperformed the state-of-art methods in fifteen synthetic datasets and in two real datasets, regarding the average error. In relation to AMANDA-DCP, it was outperformed by the state-of-the-art methods and AMANDA-FCP in majority of the datasets. However, AMANDA-DCP achieved the best result in one real dataset and presented promising results in several datasets.

Worth to mention that AMANDA-FCP obtained better quality of predictions between the classes in thirteen synthetic datasets and in all real datasets. The computational time of AMANDA-FCP was better, in general, than the state-of-the-art and AMANDA-DCP. Besides, the computational time can be reduced since that KDE is a parallelizable algorithm.

### 5.3 Contributions

We have found that applying a semi-supervised learning classifier supported by our density-based CSE method with a fixed cutting percentage improved the results for a concept-drift environment. This is due to the samples that contain new concepts from data that are kept for subsequent steps. CSE method presented relevant results and with the right value of a cutting percentage parameter, AMANDA-FCP outperformed state-of-the-art methods, regarding the average error, using only a few instances. Additionally, AMANDA-FCP presented a computational time lower than the state-of-the-art, a significant contribution for non-stationary environments.

Another contribution is the dynamic cutting percentage, an alternative for the core support extraction with a fixed parameter. The dynamic cutting method is parameter free. We adapted and combined a distance measurement for comparing two distributions and determine, dynamically, the percentage of instances to be discarded. Even the dynamic cutting approach was outperformed by the fixed cutting approach and state-of-the-art, the dynamic approach presented better results than the baselines. Our results indicate that SSL classifiers are improved when they work along with our dynamic CSE method.

### 5.4 Limitations and Research Directions

One limitation of our proposal is the fact that AMANDA-FCP needs some parameter tuning in order to find the optimal cutting percentage parameter. In the meantime, once the AMANDA-DCP does not need this tuning, it does not find a representative data for a cutting percentage value. Hence, the results of AMANDA-DCP were surpassed by AMANDA-FCP in almost all datasets. Thus, as future work, we will investigate better approaches for finding the optimal cutting percentage parameter.

Another limitation is that the two variations of AMANDA had difficulties in real datasets where the drift is recurrent. As stated by KHAMASSI *et al.* (2018), using single learner approaches are not recommended for handling recurrent drifts. As they process online, they are continuously adapted to the current concept. Hence, when a previous concept reoccurs, these approaches relearn it from scratch without taking

benefit from its previously existence. Therefore, deal better with this limitation is an important issue to be investigated as future work.

Another research direction is the selection of the most representative instances from data. The results indicate a promising direction regarding non-parametric methods based on densities. Therefore, alternative methods could be experimented.

Local concept-drift is the change that occurs in some regions of the instance space. Thus, when looking at the overall instance space, only some subsets are affected by the drift. Besides, in some cases, local concept-drift can be confused with noise, which makes the model unstable. Hence, to overcome the instability, the model has to effectively differentiate between local changes and noises (KHAMASSI *et al.*, 2018). Therefore, for this reason, a research direction could be the better understanding of what is noise and how to discard noise from data without harming the model.

Finally, the handling of concept drift under missing values is a promising research direction. The problem of missing values, which corresponds to incompleteness of features, has been investigated extensively for offline and static data. However, only few works address non-stationary environments. Hence, handling concept drift under missing value remains an open challenge (KREMPL *et al.*, 2014).

# Bibliography

- AGGARWAL, C. C., PHILIP, S. Y., HAN, J., et al., 2003, “-A Framework for Clustering Evolving Data Streams”. In: *Proceedings 2003 VLDB Conference*, pp. 81–92. Elsevier.
- ARAÚJO, L. C., SUCUPIRA, L. H., LIZARRAGA, M. G., et al., 2005, “User authentication through typing biometrics features”, *IEEE transactions on signal processing*, v. 53, n. 2, pp. 851–855.
- ASL, V. F., VAZIRI, B., RAVANMEHR, R., 2016, “A method to detect data stream changes in the wireless sensor network using the gossiping protocol”, *Indian Journal of Science and Technology*, v. 9, n. 27.
- BAENA-GARCÍA, M., DEL CAMPO-ÁVILA, J., FIDALGO, R., et al., 2006, “Early drift detection method”, .
- BIFET, A., FRANK, E., 2010, “Sentiment knowledge discovery in twitter streaming data”. In: *International conference on discovery science*, pp. 1–15. Springer.
- BIFET, A., GAVALDA, R., 2007, “Learning from time-changing data with adaptive windowing”. In: *Proceedings of the 2007 SIAM international conference on data mining*, pp. 443–448. SIAM.
- BIFET, A., HOLMES, G., PFAHRINGER, B., et al., 2009, “New ensemble methods for evolving data streams”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 139–148. ACM.
- BIFET, A., HOLMES, G., KIRKBY, R., et al., 2010, “Moa: Massive online analysis”, *Journal of Machine Learning Research*, v. 11, n. May, pp. 1601–1604.
- BIFET, A., READ, J., PFAHRINGER, B., et al., 2013, “CD-MOA: change detection framework for massive online analysis”. In: *International Symposium on Intelligent Data Analysis*, pp. 92–103. Springer.
- BISHOP, C. M., OTHERS, 2006, “Pattern recognition and machine learning, vol. 1Springer”, *New York*, , n. 4, pp. 12.

- BOSE, R. J. C., VAN DER AALST, W. M., ŽLIOBAITĚ, I., et al., 2011, “Handling concept drift in process mining”. In: *International Conference on Advanced Information Systems Engineering*, pp. 391–405. Springer.
- BRZEZIŃSKI, D., STEFANOWSKI, J., 2011, “Accuracy updated ensemble for data streams with concept drift”. In: *International conference on hybrid artificial intelligence systems*, pp. 155–163. Springer.
- CAPO, R., SANCHEZ, A., POLIKAR, R., 2014, “Core support extraction for learning from initially labeled nonstationary environments using COMPOSE”. In: *Neural Networks (IJCNN), 2014 International Joint Conference on*, pp. 602–608. IEEE.
- CARMONA-CEJUDO, J. M., BAENA-GARCÍA, M., DEL CAMPO-AVILA, J., et al., 2011, “Gnusmail: Open framework for on-line email classification”, .
- CHAO, M.-T., 2015, “Data stream classification guided by clustering on nonstationary environments and extreme verification latency”, .
- CHARLES, D., KERR, A., MCNEILL, M., et al., 2005, “Player-centred game design: Player modelling and adaptive digital games”. In: *Proceedings of the digital games research conference*, v. 285, p. 00100.
- CIESLAK, D. A., CHAWLA, N. V., 2009, “A framework for monitoring classifiers’ performance: when and why failure occurs?” *Knowledge and Information Systems*, v. 18, n. 1, pp. 83–108.
- COHEN, L., AVRAHAM-BAKISH, G., LAST, M., et al., 2008, “Real-time data mining of non-stationary data streams from sensor networks”, *Information Fusion*, v. 9, n. 3, pp. 344–353.
- DAWID, A. P., 1984, “Present position and potential developments: Some personal views: Statistical theory: The prequential approach”, *Journal of the Royal Statistical Society. Series A (General)*, pp. 278–292.
- DE SOUZA, V. M., SILVA, D. F., BATISTA, G. E., 2013, “Classification of data streams applied to insect recognition: Initial results”. In: *Intelligent Systems (BRACIS), 2013 Brazilian Conference on*, pp. 76–81. IEEE.
- DELANY, S. J., CUNNINGHAM, P., TSYMBAL, A., et al., 2005, “A case-based technique for tracking concept drift in spam filtering”, *Knowledge-Based Systems*, v. 18, n. 4, pp. 187–195.
- DEMŠAR, J., BOSNIĆ, Z., 2018, “Detecting concept drift in data streams using model explanation”, *Expert Systems with Applications*, v. 92, pp. 546–559.

- DING, Y., LI, X., ORLOWSKA, M. E., 2006, “Recency-based Collaborative Filtering”. In: *Proceedings of the 17th Australasian Database Conference - Volume 49, ADC '06*, pp. 99–107, Darlinghurst, Australia, Australia. Australian Computer Society, Inc. ISBN: 1-920682-31-7. Disponível em: <<http://dl.acm.org/citation.cfm?id=1151736.1151747>>.
- DITZLER, G., POLIKAR, R., 2011a, “Hellinger distance based drift detection for nonstationary environments”. In: *Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2011 IEEE Symposium on*, pp. 41–48. IEEE, a.
- DITZLER, G., POLIKAR, R., 2011b, “Semi-supervised learning in nonstationary environments”. In: *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 2741–2748. IEEE, b.
- DITZLER, G., ROVERI, M., ALIPPI, C., et al., 2015, “Learning in nonstationary environments: A survey”, *IEEE Computational Intelligence Magazine*, v. 10, n. 4, pp. 12–25.
- DOMINGOS, P., HULTEN, G., 2000, “Mining high-speed data streams”. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80. ACM.
- DUDA, R. O., HART, P. E., STORK, D. G., 2012, *Pattern classification*. John Wiley & Sons.
- DUDANI, S. A., 1976, “The distance-weighted k-nearest-neighbor rule”, *IEEE Transactions on Systems, Man, and Cybernetics*, , n. 4, pp. 325–327.
- DYER, K. B., CAPO, R., POLIKAR, R., 2014, “Compose: A semisupervised learning framework for initially labeled nonstationary streaming data”, *IEEE transactions on neural networks and learning systems*, v. 25, n. 1, pp. 12–26.
- EDELSBRUNNER, H., KIRKPATRICK, D., SEIDEL, R., 1983, “On the shape of a set of points in the plane”, *IEEE Transactions on information theory*, v. 29, n. 4, pp. 551–559.
- ELWELL, R., POLIKAR, R., 2011, “Incremental learning of concept drift in nonstationary environments”, *IEEE Transactions on Neural Networks*, v. 22, n. 10, pp. 1517–1531.

- ESTER, M., KRIEGEL, H.-P., SANDER, J., et al., 1996, “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *Kdd*, v. 96, pp. 226–231.
- GAMA, J. A., ŽLIOBAITĚ, I., BIFET, A., et al., 2014a, “A Survey on Concept Drift Adaptation”, *ACM Comput. Surv.*, v. 46, n. 4 (mar.), pp. 44:1–44:37. ISSN: 0360-0300. doi: 10.1145/2523813. Disponível em: <<http://doi.acm.org/10.1145/2523813>>.
- GAMA, J., MEDAS, P., CASTILLO, G., et al., 2004, “Learning with drift detection”. In: *Brazilian symposium on artificial intelligence*, pp. 286–295. Springer.
- GAMA, J., SEBASTIÃO, R., RODRIGUES, P. P., 2009, “Issues in evaluation of stream learning algorithms”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 329–338. ACM.
- GAMA, J., ŽLIOBAITĚ, I., BIFET, A., et al., 2014b, “A survey on concept drift adaptation”, *ACM Computing Surveys (CSUR)*, v. 46, n. 4, pp. 44.
- GAO, J., FAN, W., HAN, J., et al., 2007, “A general framework for mining concept-drifting data streams with skewed distributions”. In: *Proceedings of the 2007 SIAM International Conference on Data Mining*, pp. 3–14. SIAM.
- GULDEMIR, H., SENGUR, A., 2006, “Comparison of clustering algorithms for analog modulation classification”, *Expert Systems with Applications*, v. 30, n. 4, pp. 642–649.
- HACHIYA, H., SUGIYAMA, M., UEDA, N., 2012, “Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition”, *Neurocomputing*, v. 80, pp. 93–101.
- HARRIES, M., WALES, N. S., 1999, “Splice-2 comparative evaluation: Electricity pricing”, .
- HARRIES, M. B., SAMMUT, C., HORN, K., 1998, “Extracting hidden context”, *Machine learning*, v. 32, n. 2, pp. 101–126.
- HAYKIN, S., LI, L., 1995, “Nonlinear adaptive prediction of nonstationary signals”, *IEEE Transactions on signal processing*, v. 43, n. 2, pp. 526–535.
- HEARST, M. A., DUMAIS, S. T., OSMAN, E., et al., 1998, “Support vector machines”, *IEEE Intelligent Systems and their Applications*, v. 13, n. 4, pp. 18–28.

- HOENS, T. R., CHAWLA, N. V., POLIKAR, R., 2011, “Heuristic updatable weighted random subspaces for non-stationary environments”. In: *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pp. 241–250. IEEE.
- HOLMES, G., DONKIN, A., WITTEN, I. H., 1994, “Weka: A machine learning workbench”. In: *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pp. 357–361. IEEE.
- JACKOWSKI, K., 2014, “Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers”, *Pattern Analysis and Applications*, v. 17, n. 4, pp. 709–724.
- JOHN, G. H., LANGLEY, P., 1995, “Estimating continuous distributions in Bayesian classifiers”. In: *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 338–345. Morgan Kaufmann Publishers Inc.
- JOYCE, J. M., 2011, “Kullback-leibler divergence”. In: *International Encyclopedia of Statistical Science*, Springer, pp. 720–722.
- KADWE, Y., SURYAWANSHI, V., 2015, “A review on concept drift”, *IOSR J. Comput. Eng*, v. 17, pp. 20–26.
- KAILATH, T., 1967, “The divergence and Bhattacharyya distance measures in signal selection”, *IEEE transactions on communication technology*, v. 15, n. 1, pp. 52–60.
- KELLY, M. G., HAND, D. J., ADAMS, N. M., 1999, “The impact of changing populations on classifier performance”. In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 367–371. ACM.
- KHAMASSI, I., SAYED-MOUCHAWEH, M., HAMMAMI, M., et al., 2015, “Self-adaptive windowing approach for handling complex concept drift”, *Cognitive Computation*, v. 7, n. 6, pp. 772–790.
- KHAMASSI, I., SAYED-MOUCHAWEH, M., HAMMAMI, M., et al., 2018, “Discussion and review on evolving data streams and concept drift adapting”, *Evolving Systems*, v. 9, n. 1, pp. 1–23.

- KILLOURHY, K., MAXION, R., 2010, “Why did my detector do that?!” In: *International Workshop on Recent Advances in Intrusion Detection*, pp. 256–276. Springer.
- KREMPL, G., ŽLIOBAITE, I., BRZEZIŃSKI, D., et al., 2014, “Open challenges for data stream mining research”, *ACM SIGKDD explorations newsletter*, v. 16, n. 1, pp. 1–10.
- LANE, T., BRODLEY, C. E., 1998, “Approaches to Online Learning and Concept Drift for User Identification in Computer Security.” In: *KDD*, pp. 259–263.
- LAZARESCU, M. M., VENKATESH, S., BUI, H. H., 2004, “Using multiple windows to track concept drift”, *Intelligent data analysis*, v. 8, n. 1, pp. 29–59.
- LIAW, A., WIENER, M., OTHERS, 2002, “Classification and regression by randomForest”, *R news*, v. 2, n. 3, pp. 18–22.
- LIN, J., 1991, “Divergence measures based on the Shannon entropy”, *IEEE Transactions on Information theory*, v. 37, n. 1, pp. 145–151.
- LU, Z., WU, X., BONGARD, J. C., 2015, “Active learning through adaptive heterogeneous ensembling”, *IEEE Transactions on Knowledge and Data Engineering*, v. 27, n. 2, pp. 368–381.
- L’HEUREUX, A., GROLINGER, K., ELYAMANY, H. F., et al., 2017, “Machine learning with big data: Challenges and approaches”, *IEEE Access*, v. 5, pp. 7776–7797.
- MAHALANOBIS, P. C., 1936, “On the generalized distance in statistics”. National Institute of Science of India.
- MALOOF, M. A., MICHALSKI, R. S., 2004, “Incremental learning with partial instance memory”, *Artificial intelligence*, v. 154, n. 1-2, pp. 95–126.
- MARKELLOS, V., BLACK, W., MORAN, P., 1974, “A grid search for families of periodic orbits in the restricted problem of three bodies”, *Celestial mechanics*, v. 9, n. 4, pp. 507–512.
- MARRS, G. R., HICKEY, R. J., BLACK, M. M., 2010, “The impact of latency on online classification learning with concept drift”. In: *International Conference on Knowledge Science, Engineering and Management*, pp. 459–469. Springer.

- MATTHEWS, B. W., 1975, “Comparison of the predicted and observed secondary structure of T4 phage lysozyme”, *Biochimica et Biophysica Acta (BBA)-Protein Structure*, v. 405, n. 2, pp. 442–451.
- MINKU, L. L., WHITE, A. P., YAO, X., 2010, “The impact of diversity on online ensemble learning in the presence of concept drift”, *IEEE Transactions on Knowledge and Data Engineering*, v. 22, n. 5, pp. 730–742.
- MOON, T. K., 1996, “The expectation-maximization algorithm”, *IEEE Signal processing magazine*, v. 13, n. 6, pp. 47–60.
- MORALES, G. D. F., BIFET, A., 2015, “SAMOA: scalable advanced massive online analysis.” *Journal of Machine Learning Research*, v. 16, n. 1, pp. 149–153.
- MOREIRA, J. P. C. L. M., 2008, “Travel time prediction for the planning of mass transit companies: a machine learning approach”, .
- MORENO-TORRES, J. G., RAEDER, T., ALAIZ-RODRÍGUEZ, R., et al., 2012, “A unifying view on dataset shift in classification”, *Pattern Recognition*, v. 45, n. 1, pp. 521–530.
- MORRISON, R. W., DE JONG, K. A., 1999, “A test problem generator for non-stationary environments”. In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, v. 3, pp. 2047–2053. IEEE.
- MUTHUKRISHNAN, S., VAN DEN BERG, E., WU, Y., 2007, “Sequential change detection on data streams”. In: *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pp. 551–550. IEEE.
- OOMMEN, B. J., RUEDA, L., 2006, “Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments”, *Pattern Recognition*, v. 39, n. 3, pp. 328–341.
- PAWLING, A., CHAWLA, N. V., MADEY, G., 2007, “Anomaly detection in a mobile communication network”, *Computational and Mathematical Organization Theory*, v. 13, n. 4, pp. 407–422.
- PECHENIZKIY, M., BAKKER, J., ŽLIJBAITĖ, I., et al., 2010, “Online mass flow prediction in CFB boilers with explicit detection of sudden concept drift”, *ACM SIGKDD Explorations Newsletter*, v. 11, n. 2, pp. 109–116.

- PROCOPIO, M. J., MULLIGAN, J., GRUDIC, G., 2009, “Learning terrain segmentation with classifier ensembles for autonomous robot navigation in unstructured environments”, *Journal of Field Robotics*, v. 26, n. 2, pp. 145–175.
- QAHTAN, A. A., ALHARBI, B., WANG, S., et al., 2015, “A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 935–944. ACM.
- RASHIDI, P., COOK, D. J., 2009, “Keeping the resident in the loop: Adapting the smart home to the user”, *IEEE Transactions on systems, man, and cybernetics-part A: systems and humans*, v. 39, n. 5, pp. 949–959.
- REYNOLDS, D., 2015, “Gaussian mixture models”, *Encyclopedia of biometrics*, pp. 827–832.
- ROBBINS, H., SIEGMUND, D., 1971, “A convergence theorem for non negative almost supermartingales and some applications”. In: *Optimizing methods in statistics*, Elsevier, pp. 233–257.
- SALGANICOFF, M., 1997, “Tolerating concept and sampling shift in lazy learning using prediction error context switching”. In: *Lazy learning*, Springer, pp. 133–155.
- SANDER, J., ESTER, M., KRIEGEL, H.-P., et al., 1998, “Density-based clustering in spatial databases: The algorithm dbscan and its applications”, *Data mining and knowledge discovery*, v. 2, n. 2, pp. 169–194.
- SCHWARZ, G., OTHERS, 1978, “Estimating the dimension of a model”, *The annals of statistics*, v. 6, n. 2, pp. 461–464.
- SILVERMAN, B. W., 2018, *Density estimation for statistics and data analysis*. Routledge.
- SOUZA, V. M., SILVA, D. F., BATISTA, G. E., et al., 2015a, “Classification of evolving data streams with infinitely delayed labels”. In: *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pp. 214–219. IEEE, a.
- SOUZA, V. M., SILVA, D. F., GAMA, J., et al., 2015b, “Data stream classification guided by clustering on nonstationary environments and extreme verifica-

- tion latency”. In: *Proceedings of the 2015 SIAM International Conference on Data Mining*, pp. 873–881. SIAM, b.
- STANLEY, K. O., 2003, “Learning concept drift with a committee of decision trees”, *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*.
- SUN, Y., TANG, K., ZHU, Z., et al., 2018, “Concept Drift Adaptation by Exploiting Historical Knowledge”, *IEEE Transactions on Neural Networks and Learning Systems*.
- TAVASOLI, H., OOMMEN, B. J., YAZIDI, A., 2016, “On the online classification of data streams using weak estimators”. In: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pp. 68–79. Springer.
- THRUN, S., MONTEMERLO, M., DAHLKAMP, H., et al., 2006, “Stanley: The robot that won the DARPA Grand Challenge”, *Journal of field Robotics*, v. 23, n. 9, pp. 661–692.
- TSYMBAL, A., 2004, “The problem of concept drift: definitions and related work”, *Computer Science Department, Trinity College Dublin*, v. 106, n. 2.
- UMER, M., FREDERICKSON, C., POLIKAR, R., 2016, “Learning under extreme verification latency quickly: FAST COMPOSE”. In: *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, pp. 1–8. IEEE.
- UMER, M., POLIKAR, R., FREDERICKSON, C., 2017, “LEVEL IW: Learning extreme verification latency with importance weighting”. In: *Neural Networks (IJCNN), 2017 International Joint Conference on*, pp. 1740–1747. IEEE.
- WANG, S., MINKU, L. L., YAO, X., 2018, “A systematic study of online class imbalance learning with concept drift”, *IEEE Transactions on Neural Networks and Learning Systems*.
- WEBB, G. I., HYDE, R., CAO, H., et al., 2016, “Characterizing concept drift”, *Data Mining and Knowledge Discovery*, v. 30, n. 4, pp. 964–994.
- WEBB, G. I., LEE, L. K., PETITJEAN, F., et al., 2017, “Understanding Concept Drift”, *arXiv preprint arXiv:1704.00362*.
- WIDMER, G., KUBAT, M., 1993, “Effective learning in dynamic environments by explicit context tracking”. In: *Machine learning: ECML-93*, pp. 227–243. Springer.

- WIDMER, G., KUBAT, M., 1996, “Learning in the presence of concept drift and hidden contexts”, *Machine learning*, v. 23, n. 1, pp. 69–101.
- WILSON, D. R., MARTINEZ, T. R., 2000, “Reduction techniques for instance-based learning algorithms”, *Machine learning*, v. 38, n. 3, pp. 257–286.
- WOLD, S., ESBENSEN, K., GELADI, P., 1987, “Principal component analysis”, *Chemometrics and intelligent laboratory systems*, v. 2, n. 1-3, pp. 37–52.
- WU, W., QIN, H., 2018, “Reducing Noise for PIC Simulations Using Kernel Density Estimation Algorithm”, *arXiv preprint arXiv:1804.11174*.
- XIE, C.-H., SONG, Y.-Q., CHEN, J.-M., 2011, “Fast medical image mixture density clustering segmentation using stratification sampling and kernel density estimation”, *Signal, Image and Video Processing*, v. 5, n. 2, pp. 257–267.
- XU, J., ZHU, S., GUO, H., et al., 2017, “Avoidance of Manual Labeling in Robotic Autonomous Navigation Through Multi-Sensory Semi-Supervised Learning”, *arXiv preprint arXiv:1709.07911*.
- ZHANG, T., RAMAKRISHNAN, R., LIVNY, M., 1997, “BIRCH: A new data clustering algorithm and its applications”, *Data Mining and Knowledge Discovery*, v. 1, n. 2, pp. 141–182.
- ZHOU, J., CHENG, L., BISCHOF, W. F., et al., 2008, “Prediction and Change Detection in Sequential Data for Interactive Applications.” In: *AAAI*, pp. 805–810.
- ZHU, X., GHAMRANI, Z., 2002, “Learning from labeled and unlabeled data with label propagation”, .
- ŽLIOBAITĖ, I., BIFET, A., READ, J., et al., 2015, “Evaluation methods and decision theory for classification of streaming data with temporal dependence”, *Machine Learning*, v. 98, n. 3, pp. 455–482.
- ŽLIOBAITĖ, I., PECHENIZKIY, M., GAMA, J., 2016, “An overview of concept drift applications”. In: *Big Data Analysis: New Algorithms for a New Society*, Springer, pp. 91–114.