UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JORGE CRISTHIAN CHAMBY DIAZ

An Incremental Gaussian Mixture Network for Data Stream Classification in Non-Stationary Environments

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof^{a}. Dra. Ana L. C. Bazzan
Coadvisor: Prof^{a}. Dra. Mariana R. Mendoza

Porto Alegre
January 2018

CIP — CATALOGING-IN-PUBLICATION

Chamby Diaz, Jorge Cristhian

An Incremental Gaussian Mixture Network for Data Stream Classification in Non-Stationary Environments / Jorge Cristhian Chamby Diaz. – Porto Alegre: PPGC da UFRGS, 2018.

117 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2018. Advisor: Ana L. C. Bazzan; Coadvisor: Mariana R. Mendoza.

1. Incremental learning. 2. Data streams classification. 3. Concept drift. 4. Gaussian mixture models. I. Bazzan, Ana L. C.. II. Mendoza, Mariana R.. III. Título.

“The people who get on in this world are the people
who get up and look for the circumstance they want,
and if they can’t find them – Make Them.”

— George Bernard Shaw

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors: firstly Prof. Paulo Engel for his continuous support with my study and research, for his patience, motivation, enthusiasm, and immense knowledge. Besides Prof. Paulo, I would like to thank my second advisor Profa. Ana Bazzan and my co-advisor Profa. Mariana Recamonde Mendoza for their valuable guidance in the final stages of my research. You definitely provided me with the tools that I needed to choose the right direction and successfully complete my dissertation.

I thank my fellow colleagues and friends, Rafael, Anderson, Fabian, Jose Luis and Abel, for the stimulating suggestions, companionship, comments and for all the fun we have had in the last two years.

Last but not the least, I especially thank my parents Jorge and Jesus, and my sisters Flor, Yenny, Guadalupe and Claudia. My hard-working parents have sacrificed their lives for my sisters and myself and provided unconditional love and care. I love them so much, and I would not have made it this far without them. I know I always have my family to count on when times are rough.

Thank you very much everyone!

ABSTRACT

Data stream classification poses many challenges for the data mining community when the environment is non-stationary. The greatest challenge in learning classifiers from data stream relates to adaptation to the concept drifts, which occur as a result of changes in the underlying concepts. Two main ways to develop adaptive approaches are ensemble methods and incremental algorithms. Ensemble method plays an important role due to its modularity, which provides a natural way of adapting to change. Incremental algorithms are faster and have better anti-noise capacity than ensemble algorithms, but have more restrictions on concept drifting data streams. Thus, it is a challenge to combine the flexibility and adaptation of an ensemble classifier in the presence of concept drift, with the simplicity of use found in a single classifier with incremental learning. With this motivation, in this dissertation we propose an incremental, online and probabilistic algorithm for classification as an effort of tackling concept drifting. The algorithm is called IGMN-NSE and is an adaptation of the IGMN algorithm. The two main contributions of IGMN-NSE in relation to the IGMN are: predictive power improvement for classification tasks and adaptation to achieve a good performance in non-stationary environments. Extensive studies on both synthetic and real-world data demonstrate that the proposed algorithm can track the changing environments very closely, regardless of the type of concept drift.

Keywords: Incremental learning. Data streams classification. Concept drift. Gaussian mixture models.

Uma Rede de Mistura de Gaussianas Incremental para Classificação de Fluxos Contínuos de Dados em Cenários não Estacionários

RESUMO

Classificação de fluxos contínuos de dados possui muitos desafios para a comunidade de mineração de dados quando o ambiente não é estacionário. Um dos maiores desafios para a aprendizagem em fluxos contínuos de dados está relacionado com a adaptação às mudanças de conceito, as quais ocorrem como resultado da evolução dos dados ao longo do tempo. Duas formas principais de desenvolver abordagens adaptativas são os métodos baseados em conjunto de classificadores e os algoritmos incrementais. Métodos baseados em conjunto de classificadores desempenham um papel importante devido à sua modularidade, o que proporciona uma maneira natural de se adaptar a mudanças de conceito. Os algoritmos incrementais são mais rápidos e possuem uma melhor capacidade anti-ruído do que os conjuntos de classificadores, mas têm mais restrições sobre os fluxos de dados. Assim, é um desafio combinar a flexibilidade e a adaptação de um conjunto de classificadores na presença de mudança de conceito, com a simplicidade de uso encontrada em um único classificador com aprendizado incremental. Com essa motivação, nesta dissertação, propomos um algoritmo incremental, online e probabilístico para a classificação em problemas que envolvem mudança de conceito. O algoritmo é chamado IGMN-NSE e é uma adaptação do algoritmo IGMN. As duas principais contribuições da IGMN-NSE em relação à IGMN são: melhoria de poder preditivo para tarefas de classificação e a adaptação para alcançar um bom desempenho em cenários não estacionários. Estudos extensivos em bases de dados sintéticas e do mundo real demonstram que o algoritmo proposto pode rastrear os ambientes em mudança de forma muito próxima, independentemente do tipo de mudança de conceito.

Palavras-chave: Aprendizado incremental. Classificação de fluxos continuos de dados. Mudança de conceito. Modelos de Mistura de Gaussianas..

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

DS	Data Stream
IGMN	Incremental Gaussian Mixture Network
NSE	Non-stationary Environment
GMM	Gaussian Mixture Model
VFDT	Very Fast Decision Tree
CVFDT	Concept-Adapting Very Fast Decision Tree
HWT	Hoeffding Window Tree
HAT	Hoeffding Adaptive Tree
SVM	Support Vector Machine
IGMM	Incremental Gaussian Mixture Model
CUSUM	Cumulated Sum
GMA	Geometric Moving Average
DDM	Drift Detection Method
EDDM	Early Drift Detection Method
EWMA	Exponentially Weighted Moving Average
ECDD	EWMA for Concept Drift Detection
DWM	Dynamic Weighted Majority
DWMIL	Dynamic Weighted Majority for Imbalance Learning
OAUE	Online Accuracy Updated Ensemble
WAOAUE	Window-Adaptive Online Accuracy Updated Ensemble
PDF	Probability Density Function
FIGMN	Fast Incremental Gaussian Mixture Network

LIST OF SYMBOLS

x^t	Data point (example)
x_i^t	Input part of the data point
x_t^t	Output part (target) of the data point
μ_j	Mean vector of the j^{th} IGMN neuron
C_j	Covariance Matrix of the j^{th} IGMN neuron
Λ_j	Precision Matrix of the j^{th} IGMN neuron
$p(j)$	Prior probability of the j^{th} IGMN neuron
sp_j	Posterior probability accumulator of the j^{th} IGMN neuron
v_j	Age of the j^{th} IGMN neuron
δ	Fraction of the overall input variables variance at IGMN
τ	Novelty criterion threshold at IGMN
sp_{min}	Minimum activation required to a IGMN neuron not to be considered noise
v_{min}	Minimum age required to a IGMN neuron not to be considered noise
σ_{ini}	The initial variance of a IGMN neuron
$p(x)$	Probability density of the input x
$p(x\|j)$	The probability density function of an observing vector x belonging to the j^{th} IGMN neuron
$p(j\|x)$	The posterior probability computed by IGMN
ϕ	Maximum overlap rate accepted
spU_j	Up-to-dated posterior probability accumulator

CONTENTS

1 INTRODUCTION

1.1 Context

Traditional data mining research mostly focused on static environments, where a complete dataset is presented to the learning algorithm and the target concepts that should be learned are fixed. However, in some of the newest applications, learning algorithms work in dynamic environments, where data are continuously generated. Some of these new applications include mining query streams (COBOS et al., 2014), network monitoring (GUPTA et al., 2016), sensor networks (SUN; CAI; HUANG, 2010), and social network streams (BARDDAL et al., 2016). In these dynamic environments, incoming data form a data stream characterized by huge volumes of instances and rapid arrival-rate, which often requires quick, real-time response. Compared to static environments, the processing of data streams implies new requirements for algorithms, such as constraints on memory usage, restricted learning, and testing time, and one scan of incoming instances (BRZEZIŃSKI; STEFANOWSKI, 2011). On top of that, problems found in a static environment setting are also present in a data stream, for example, absent values, overfitting, noise, irrelevant features, class imbalance, and others. Furthermore, due to the non-stationary nature of data streams, an additional problem is that characteristics of the data might change over time, a condition known as *concept drift*. Concept drift occurs when the concept about which data are being collected shifts from time to time after a minimum stability period (GAMA, 2010). Data streams that exhibit concept drifts are referred to as evolving or non-stationary data streams (GOMES et al., 2017).

In this dissertation, we focus on data stream classification in non-stationary environments. Data stream classification is a variation of the traditional supervised machine-learning task of classification. Both tasks are concerned with the problem of predicting a nominal value of an unlabeled instance represented by a vector of characteristics. A brief summary of those previous studies related to data stream classification follows.

The first approach is incremental learning. Incremental learning is a style of learning where the learner updates its model of the environment when a new significant experience becomes available. The challenge of learning in various environments is how to preserve all acquired knowledge, so that the learner must decide what knowledge should be replaced or retained for improving its performance (DRIES; RüCKERT, 2009). This is known as the "stability-plasticity dilemma", where "stability" means to maintain existing knowledge and "plasticity" describes the ability to learn new knowledge (ELWELL; POLIKAR, 2011).

An alternate approach to incremental learning is the use of ensemble learning. Ensemble learners are popular in the data stream setting, because besides leveraging weak learners, they can be used to handle machine challenges that arise from data streams scenarios, such as concept drift (KOLTER; MALOOF, 2007; ELWELL; POLIKAR, 2011; BRZEZINSKI; STEFANOWSKI, 2014).
Both of these approaches can handle big stream data and concept drift problems, and each of them has its own strengths. In general, incremental learning algorithms have better efficiency and ensemble learning adapts better to concept drift (ZANG et al., 2014).

1.2 Motivation

An ideal solution for non-stationary data stream classification would be to combine the modularity of an ensemble classifier in the presence of concept drift, with the simplicity of use found in a single classifier with incremental learning. With this motivation in mind, we propose the use of IGMN (Incremental Gaussian Mixture Network) (HEINEN; ENGEL; PINTO, 2011), which is a neural network based on parametric probabilistic models. IGMN meets most of the requirements we are interested in, i.e., (i) the neural network topology is defined automatically and incrementally, (ii) it does not require access to previously used data, (iii) it retains a substantial majority of the previously acquired knowledge when learning new information, (iv) the learning process can proceed perpetually, (v) it can handle the stability-plasticity dilemma, and finally (vi) it can be used in supervised, unsupervised learning tasks. However, when solving data stream classification problems, IGMN may have some issues, especially with the presence of concept drift.

1.3 Goal

Based on these facts and some preliminary results, the main goal of this dissertation is to present a new approach for data stream classification based on an adaptation of the IGMN algorithm, named IGMN-NSE, which stands for Incremental Gaussian Mixture Network for Non-Stationary Environments. The IGMN-NSE proposes a predictive power improvement for classification tasks and a policy to discard outdated concepts in contrast to the standard IGMN model. We present and test this model on some known synthetic and real datasets against classical models and the standard version of the IGMN.

1.4 Dissertation structure

This dissertation is organized as follows: Chapter 2 presents a detailed review of data streams, as a background to our proposed work. Chapter 3 describes the standard IGMN model, including is architecture and operations. Chapter 4 revises the related literature, outlining current approaches for concept drifting data stream classification. Chapter 5 describes our improvements to the IGMN algorithm and our final model. Chapter 6 describes the experiments and results obtained with IGMN-NSE, and its comparison against the standard IGMN and other classifier models. Finally, conclusions and directions for future works are discussed in Chapter 7.

2 DATA STREAMS: A REVIEW

Before describing and evaluating different approaches to mining streams with concept drift, we present basic definitions of data streams. First, in Section 2.1, we focus on the main definitions of a data stream model and how it differs from traditional data. In addition, we define and categorize concept drift and its causes. Next, in Section 2.2, we present a taxonomy of adaptive classification techniques. Since this dissertation concentrates on classification techniques, we will use the term "data stream learning" as a synonym for "data stream mining".

2.1 Important definitions

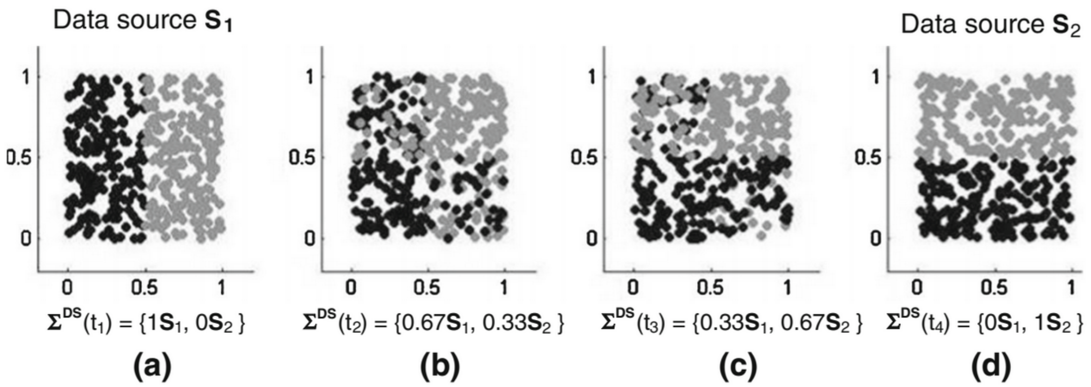
2.1.1 Data streams

A *data stream* is a sequence of elements, continuous and ordered, where each data has a specific arrival time (GOLAB; OZSU, 2003). The data streams have an unlimited size and are impossible to store in memory, resulting in a challenge for the data mining methods. The main characteristics of the data stream model imply the following constraints (BIFET, 2009):

1. Due to infinite size of a data stream, it is not possible to store it. Only small summaries of data streams can be computed and stored, and the rest of the information is thrown away.
2. The data stream flows continuously at very high speed, and the arrival of data stream tuples forces each particular element to be processed essentially in real time, and then discarded.
3. The distribution can experience changes over time. Thus, data from the past may become irrelevant or even harmful for the current summary.

Constraint 1 limits the amount of memory, since it is impossible to store streaming data, as opposed to traditional datasets which can be stored in memory. Constraint 2 limits the processing time per element that the streaming algorithm can use. In summary, data streams and sliding-window are techniques for computing fundamental functions with low memory and time-per-item.

Figure 2.1: Example of a gradual change from a data source S_1 to a data source S_2 (NARASIMHAMURTHY; KUNCHEVA, 2007). Class y_1 is depicted with *gray dots*, and class y_2 with *black dots*



Constraint 3 is the necessity of a model to adapt to time changes. Depending on the approach used in the construction of a model, this constraint can be ignored (BRZEZINSKI, 2010), since *static data stream learning* algorithms only consider the first two constraints, while *evolving data stream learning* algorithms consider constraint 3 as a key feature. Both of these approaches to stream data mining will be discussed in Section 2.1.3.

2.1.2 Concept drift

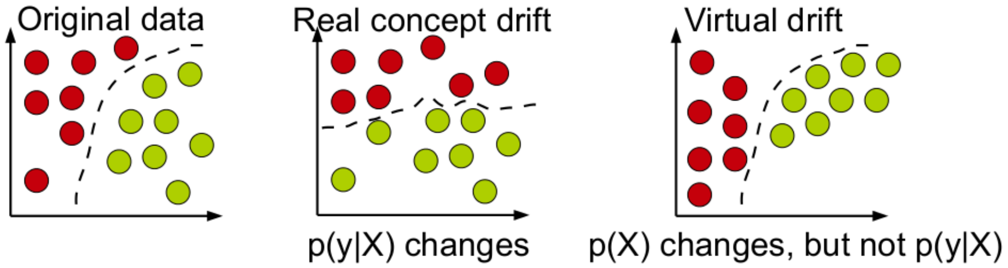
Traditional data mining works with static distributions, i.e. those which do not evolve over time. These distributions are generated from a single *concept* or a *data source*, which means that training and testing datasets share the same distribution. Meanwhile, data streams are dynamic and have many concepts (NGUYEN; WOON; NG, 2015). An example is the problem of preventing credit card fraud, where an event can be classified either as “fraudulent" or "non-fraudulent". These descriptions evolve over time into new fraud techniques, thereby changing the definition of classification "fraudulent".

The term *concept drift* indicates variation of data sources over time, i.e. the data stream has two consecutive data sources with different distributions. Figure 2.1 shows an example of a concept drift, where the data distribution of a data stream DS experiences a gradual change from a data source S_1 to a data source S_2.

Formally concept drift between time point t_0 and time point t_1 can be defined as (GAMA et al., 2014):

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y) \tag{2.1}$$

Figure 2.2: Types of drift: circles represent instances, different colors represent different classes (GAMA et al., 2014)



where p_{t_0} and p_{t_1} denote the joint distribution at time t_0 and t_1, respectively, between the set of input variables X and the target variable y. Kelly, Hand and Adams (1999) presented three ways in which concept drift may occur:

- prior probabilities of classes $P(y)$ may change over time,
- class-conditional probability distributions $P(X|y)$ may change,
- posterior probabilities $P(y|X)$ may change.

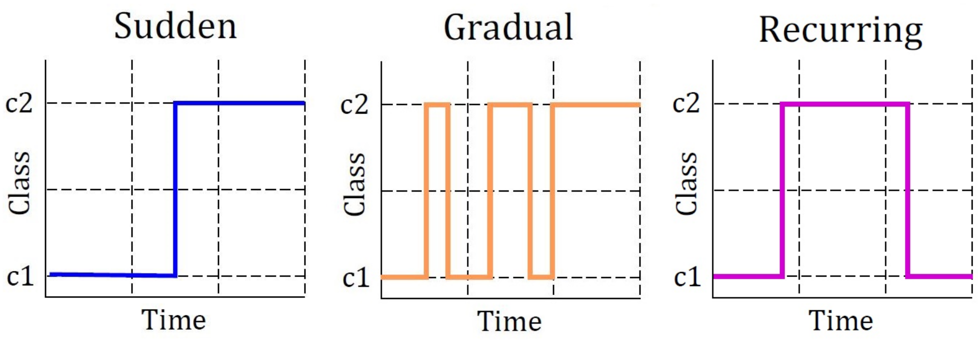
In these scenarios of concept drift, we can distinguish two different types of drifts: *virtual concept drift* and *real concept drift*. In virtual concept drift, the distributions $P(X|y)$ change in such a way that the class membership is not affected (e.g. symmetric movement to opposite directions). Alternatively, real concept drift is defined as a change in the class membership, i.e. changes in $P(y|X)$. Figure 2.2 illustrates the types of drift. From a practical point of view, the distinction between virtual and real drifts is of little importance (BRZEZINSKI, 2010), and we will not make that distinction in this dissertation.

In addition to classifying the different types of concept drift as being real or virtual, it is common to classify drift based on speed. In the next Section 2.1.2.1, we investigate the various speeds with which concept drift may occur, and discuss their relative effects.

2.1.2.1 Speed of drift

When data streams experience concept drift, we can identify different speed changes that happen over time (the interval of time necessary for change from one concept to another). Basically, the speed can occur in two main ways: sudden or gradual (HOENS;

Figure 2.3: Types of changes in streaming data (BRZEZINSKI, 2010).



POLIKAR; CHAWLA, 2012). For our explanation, we assume that f generates a first concept, and g generates a second concept of the data stream.

In *sudden concept drift* (shown in Figure 2.3), also known as *concept change*, there exists a specific point in time at which f stops being used to generate the concepts. At this point g starts to be used instead. Since sudden concept drift is defined as having an abrupt limit between generating functions, it is the simplest case of concept drift. The reason for this is that it is easy to detect future data as it is significantly different from past data.

On the other hand, *gradual concept drift* (shown in Figure 2.3) occurs when the distribution evolves slowly over time with smooth transition from sampling f to sampling g. This type of concept drift is often more difficult to detect, because the change from f to g is slow. The track left by the drift may be hidden, increasing the probability that the classifier will miss it.

Changes that are only temporary and are reverted over time, are called *recurring concept drift* and can reoccur either suddenly or gradually. Some classification models only store the current concept and must relearn each time a new concept appears. For data streams that experience recurring concept drift, the ideal classification model should retain knowledge of the previous concept.

2.1.3 Data stream mining

Data streams have intrinsic characteristics, different from traditional data mining. Table 2.1 presents a comparison between traditional and stream data mining.

Traditional data mining has only one concept and can scan datasets at any time. It is able to execute this without limit of memory or time to produce accurate results.

Table 2.1: Comparison between traditional and stream data mining (NGUYEN; WOON; NG, 2015; BRZEZINSKI, 2010)

Characteristics/domains	Traditional data mining	Data stream mining
Nro. of passes	Multiple	Single
Processing time	Unlimited	Restricted
Memory usage	Unlimited	Restricted
Type of result	Accurate	Approximate
Concept	Static	Evolving

On the other hand, data stream classifiers need to process the data sequentially with the intention of learning new class descriptions. This constraint is satisfied using "online learners". Another important constraint is that it must detect and react to changes in the environment, normally fulfilled by implementing a "forgetting method" to "unlearn" outdated knowledge (BRZEZINSKI, 2010).

2.1.3.1 Online learning

Online learning, also termed *incremental learning*, is primarily focused on processing the data in a sequential way so that, in the end, the trained classifier should ideally be equivalent to a classifier trained on the batch data (as in traditional data mining). Each incoming example is classified, and its true label is recovered at some later stage (in the simplest case, immediately). With this example and its true label, the classifier is updated, minimally if possible, to accommodate the new training point. Online learners, typically assume a static environment, so that the forgetting of learned knowledge is not considered.

A well designed online classifier should have the following qualities (DOMINGOS; HULTEN, 2003; KUNCHEVA, 2004; BRZEZINSKI, 2010):

1. **Incremental**: The classifier should read blocks of data at a time, instead of requiring all of it at the beginning.

2. **Single pass through the data**: The classifier should make only one pass through the data to learn from each example without revisiting it.

3. **Limited time and memory**: Each example should be processed in (small) constant time, and require an approximately constant amount of memory, independent of the number of examples processed in the past.

4. **Any-time learning**: If stopped at time t, before its conclusion, the classifier should provide the best possible answer as it is at than time t.

However, it is not enough that this list of qualities be learned in changing environments. As was mentioned previously, for data mining with concept drift, the classifier should be able to learn new class descriptions and implement a forgetting mechanism.

2.1.3.2 Forgetting mechanisms

Suppose the classifier has the ability to learn online and it is kept constantly up-to-date. When the data streams experience environment changes, a data stream classifier, in addition to learning online, should be able to learn new class descriptions and “forget” outdated knowledge. The main problem is how to choose the rate of forgetting, considering that different types of changes can appear (as gradual or sudden concept drift). Three forgetting strategies may be designed (KUNCHEVA, 2004):

- **Forgetting by Aging at a Constant Rate.** This strategy uses a *window* of the most recent examples (training set) to update the classifier. Since the size of the window is fixed, some problems may arise. For example, if the window is small, the system will be very responsive and will react quickly to changes but the accuracy of the classifier may be low due to insufficient training data in the window. On the other hand, a large window may lead to a sluggish but stable and well trained classifier. The decision between these two alternatives is viewed as the "stability-plasticity dilemma" (ELWELL; POLIKAR, 2011). This strategy is usually better suited for environments with gradual concept drift.

- **Forgetting by Aging at a Variable Rate.** This strategy is based on concept drift detection. When the change is detected, the window is shrunk (past examples are forgotten). For a static bout, the window is expanded to a predefined limit. This variable window allows the best balance between accuracy and flexibility to be found. However, this strategy is usually better suited for environments where the change is easy to detect, i.e. with sudden concept drift.

- **Density-Based Forgetting.** When older data points are more useful than recent points, deleting training data according to its class distribution is considered in adaptive nearest neighbors. This occurs where the classifier attaches to each data point, a weight which can decay with time or may be modified depending on the neighborhood of the most recent examples. This strategy usually is used for a recurring context.

2.2 Taxonomy of methods

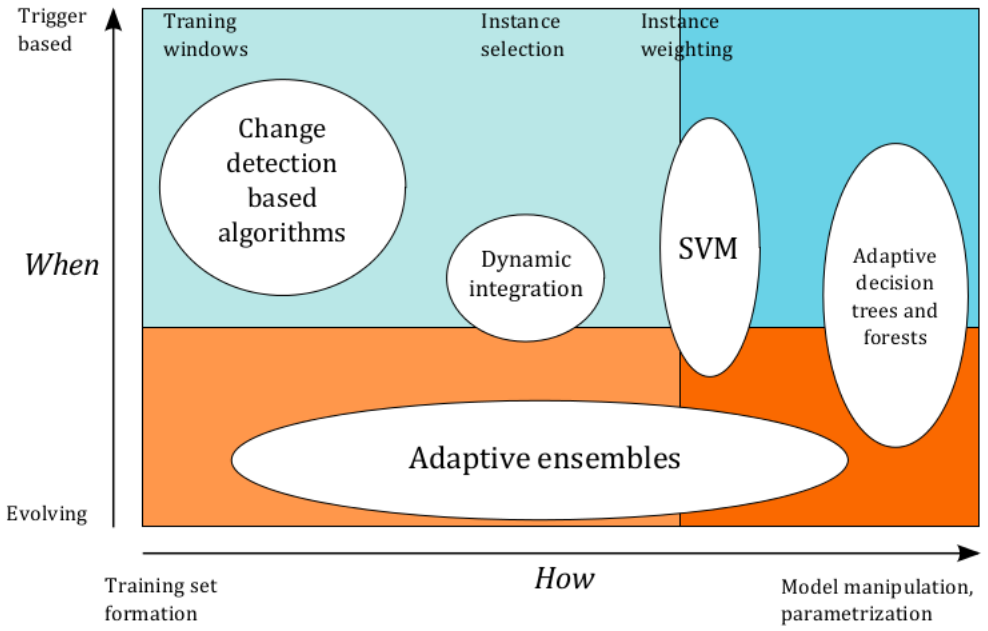
Zliobaite (2010), who focused on supervised learning under concept drift, proposed a taxonomy of learners that addressed the problem of concept drift. The taxonomy is graphically presented in Figure 2.4.
One dimension of this taxonomy is based on "when" the adaptivity is activated in the concept drift learner. In this dimension, two groups of learners can be identified:

- **Trigger based learners:** These learners incorporate a mechanism to detect when it is necessary to update the model (usually with a drift detector). Such methods work well in data streams with sudden drift, because with this type of drift it is easy to detect when the data stream experience changes.

- **Evolving learners:** These learners update the model gradually and usually do not detect changes. To adapt to changing environments, these learners keep alternative models or manipulate ensemble weights, trying to adapt their models without the necessity of rebuilding them.

Another dimension of this taxonomy is based on "how" the learners adapt to concept drift. The adaptation mechanisms mainly involve parametrization of the base learner and example selection. Basic methods of adjusting models to changing concepts were presented in Section 2.1.3.2. Detailed descriptions of some approaches presented in Figure 2.4 will be discussed in Chapter 4.

Figure 2.4: A taxonomy of adaptive supervised learning techniques (ZLIOBAITE, 2010).



3 INCREMENTAL GAUSSIAN MIXTURE NETWORK

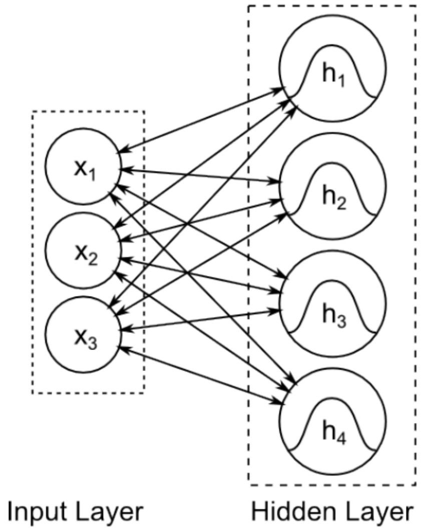
The Incremental Gaussian Mixture Network (IGMN) (HEINEN; ENGEL; PINTO, 2011; HEINEN, 2011; HEINEN; ENGEL, 2011), is a probabilistic neural network based on Gaussian Mixture Models (GMMs), inheriting, therefore, an important feature from a representation point of view: IGMN describes noisy environments in a very parsimonious way, with parameters that are readily understandable (HEINEN; ENGEL; PINTO, 2011). Differently from IGMM (ENGEL; HEINEN, 2011), however, the IGMN is capable of supervised learning, simply by assigning any of its input vector elements as outputs, i.e. any element can be used to predict any other element, like auto associative neural networks (GROSSBERG, 1987).

This chapter is structured as follows: Section 3.1 presents the operations of IGMN. Section 3.2 describes the architecture of IGMN with an overview of its mechanisms; Sections 3.3 and 3.4 present in detail the learning and recalling mode of the network, respectively. Section 3.5 presents the fast version of IGMN, named FIGMN; and, finally, Section 3.6 gives a summary of all configuration parameters of the networks and its algorithm in pseudo-code.

3.1 Operations

The network operations can be summarized in two modes: **(a) learning mode**, in which IGMN updates the neurons for new input patterns if at least one neuron can properly represent the new information, creates new neurons if there is no neuron able to represent it, and removes noisy neurons, which represent noise data. IGMN can perform the learning mode online and perpetually, without suffering from catastrophic interference. Thus, a complete retraining is not necessary when new training input is presented to the network. After at least one learning step, the network can perform the **(b) recalling mode** to estimate the missing elements at the input layer. IGMN can estimate any number of missing input elements (e.g., presenting to the network an incomplete pattern) through a weighted sum of the regression performed by its hidden neurons.

Figure 3.1: Example of an IGMN with 3 input nodes and 4 hidden neurons (PEREIRA, 2013).



3.2 Architecture

IGMN has a network topology with two layers as depicted in Figure 3.1.

The input layer has as many neurons as input variables have the input vector. Each neuron j receives one signal from the j^{th} variable of the input vector x^t.

The hidden layer, also known as association region, initially has a single neuron, but more neurons are incrementally added when necessary. At the same time, the layer can remove neurons when it considers them as noise or spurious (i.e. those which do not have a minimum activation during a certain number of steps).

The two layers are fully connected bidirectionally to each other, but not to their own neurons. It is important to notice that these connections have not weights, i.e. all IGMN parameters are stored in the neurons of the hidden layer. The bottom-up connections (from the input to the hidden layer) are used for the learning mode. On the other hand, the top-down (from the hidden to the input layer) connections return the estimations using the recalling mode.

3.3 Learning Mode

This process can be performed perpetually, there is no need for a single and exclusive training phase. When a new input pattern x is given to IGMN, it calculates the squared Mahalanobis distance $d^2_M(x, j)$ to every existing Gaussian components j of its mixture model, equivalent to neuron j of the hidden layer:

$$d^2_M(x, j) = (x - \mu_j)^T C_j^{-1} (x - \mu_j) \tag{3.1}$$

where μ_j and C_j are the mean and covariance matrix of the j^{th} neuron, respectively. IGMN assumes that the probability density of the input pattern $p(x)$ can be modeled by a linear combination of multivariate Gaussian density components, in the form:

$$p(x) = \sum_{j=1}^{M} p(x|j)p(j) \tag{3.2}$$

where M is the number of neurons, or components. The coefficients $p(j)$ are related to the prior probability of x be generated by neuron j of the hidden layer. The Probability Density Function (PDF) of an observing vector x belonging to the j^{th} neuron, $p(x|j)$, is computed as a multivariate normal distribution:

$$p(x|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|C_j|}} \exp\left(-\frac{1}{2} d^2_M(x, j)\right) \tag{3.3}$$

where D is the dimensionality of the vector x. Posteriori probabilities $p(j|x)$ are calculated for each component as follows:

$$p(j|x) = \frac{p(x|j)p(j)}{\sum_{k=1}^{K} p(x|k)p(k)} \forall j \tag{3.4}$$

3.3.1 Novelty criterion

IGMN has a novelty criterion to decide if it is necessary to add a new neuron to the model. In recent researches, three criteria were proposed:

1. **Minimum likelihood criterion**: Engel and Heinen (2011) used a minimum likelihood criterion, and a parameter τ_{nov} is necessary for achieving the required approximation level.

$$p(x|j) < \frac{\tau_{nov}}{(2\pi)^{D/2}\sqrt{|C_j|}} \tag{3.5}$$

2. **Error-driven mechanism**: Heinen, Engel and Pinto (2011) adopted an error-driven mechanism, if the instantaneous approximation error ε is larger than a user specified

threshold ε_{max}, a new neuron is added, ε is given by:

$$\varepsilon = max_{k \in z} \left\{ max_{i \in D^K} \left\{ \frac{\|k_i - \hat{k_i}\|}{max(k_i) - min(k_i)} \right\} \right\} \tag{3.6}$$

where $[min(k_i), max(k_i)]$ defines the domain of the feature k_i.

3. **Chi squared test**: Pinto and Engel (2015) proposed that new neurons would be created if any $d^2_M(x, j)$ is smaller than $\chi^2_{D,1-\tau}$ (the $1 - \tau$ percentile of a chi-squared distribution with D degrees-of-freedom, where D is the input dimensionality and τ is a user defined meta-parameter, e.g., 0.1).

$$d^2_M(x, j) < \chi^2_{D,1-\tau} \tag{3.7}$$

3.3.2 Creating neurons

When a new data point x does not meet the novelty criterion, IGMN creates a new neuron. The new neuron j is created with the following configuration:

- Mean $\mu_j = x$,
- Covariance matrix $C_j = \sigma^2_{ini} \cdot I$,
- Posteriori accumulator $sp_j = 1$ and
- Prior probability $p(j) = \frac{1}{\sum_{i=1}^{K} sp_i}$

where K is the number of neurons, including the newly created. The parameter σ_{ini} can be obtained by:

$$\sigma_{ini} = \delta \cdot std(x) \tag{3.8}$$

where δ is a manually chosen scaling factor (e.g., 0.01) and std is the standard deviation of the dataset. Note that the IGMN is an online and incremental algorithm and therefore it may be the case that we do not have the entire dataset to extract descriptive statistics. In this case the standard deviation can be just an estimation (e.g., based on sensor limits from a robotic platform), without impacting the algorithm (PINTO; ENGEL, 2015).

3.3.3 Updating neurons

When a new data point x meets the novelty criterion, x is assimilated by existing neurons and all parameters of the model are updated. Update equations are derived from the Robbins-Monro stochastic approximation (ROBBINS; MONRO, 1951) and the derivations are available in Engel and Heinen (2011) and Engel (2009). Now, parameters of the algorithm must be updated according to the following equations:

$$v_j(t) = v_j(t-1) + 1 \tag{3.9}$$

$$sp_j(t) = sp_j(t-1) + p(j|x) \tag{3.10}$$

$$e_j = x - \mu_j(t-1) \tag{3.11}$$

$$\omega_j = \frac{p(j|x)}{sp_j} \tag{3.12}$$

$$\Delta\mu_j = \omega_j e_j \tag{3.13}$$

$$\mu_j(t) = \mu_j(t-1) + \Delta\mu_j \tag{3.14}$$

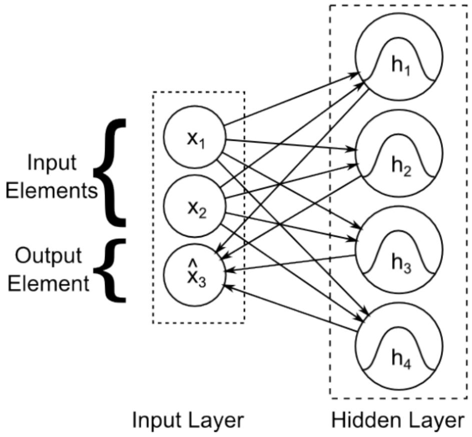
$$e_j^* = x - \mu_j(t) \tag{3.15}$$

$$C_j(t) = (1-\omega_j)C_j(t-1) + \omega_j e_j^* e_j^{*T} - \Delta\mu_j \Delta\mu_j^T \tag{3.16}$$

$$p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q} \forall j \tag{3.17}$$

where sp_j and v_j are the accumulator and the age of component j, respectively, and $p(j)$ is its prior probability.

Figure 3.2: The information flow of IGMN in recalling mode: the input layer sends the known values to the hidden layer and the hidden layer sends back an estimate of x_3 (PEREIRA, 2013).



3.3.4 Removing neurons

Optionally, a neuron j is removed whenever $v_j > v_{min}$ and $sp_j < sp_{min}$, where v_{min} and sp_{min} are manually chosen (e.g., 5.0 and 3.0, respectively). In this case, also, $p(k)$ must be adjusted for all $k \in K, k \neq j$, using the equation 3.17. In other words, each neuron is given some time v_{min} to show its importance to the model in the form of an accumulation of its posterior probabilities sp_j.

3.4 Recalling mode

During training, IGMN does not distinguish between inputs and targets, they are presented together, as an input tuple. For making an inference, IGMN interpolates the center (means) of the target variable conditioned to the posterior probabilities of the given variables. For example, if we have a tridimensional dataset, where each $x = (x_1, x_2, x_3)$. If we want to estimate x_3 given (x_1, x_2), first we split this tuple in two parts: $x_t = x_3$ (target) and $x_i = x_1, x_2$ (input). Figure 3.2 shows the information flow of IGMN in this mode. Then, the posterior probabilities are calculated as in equation 3.18. The target x_t can be estimated using the conditional mean equation presented in equation 3.19.

$$p(j|x_i) = \frac{p(x_i|j)p(j)}{\sum_{k=1}^{K} p(x_i|k)p(k)} \forall j \tag{3.18}$$

$$\hat{x}_t = \sum_{j=1}^{K} p(j|x_i)(\mu_{j,t} + C_{j,ti}C_{j,i}^{-1}(x_i - \mu_{j,i})) \tag{3.19}$$

where $C_{j,ti}$ is the sub-matrix of the jth component covariance matrix associating the unknown and known parts of the data, $C_{j,i}$ is the sub-matrix corresponding to the known part only and $\mu_{j,t}$ is the mean of the jth component of the target variable and $\mu_{j,i}$ is the mean of the jth component of the input data. The covariance matrix with the mentioned sub-matrices is show below:

$$C_j = \left(\begin{array}{c|c} C_{j,i} & C_{j,ti} \\ \hline C_{j,ti} & C_{j,t} \end{array}\right) \tag{3.20}$$

3.5 Fast IGMN

The IGMN suffers from cubic time complexity due to matrix inversion operations and determinant computations (For the equations 3.1, 3.3 and 3.19). In Pinto and Engel (2015), rank-one updates for both inverse matrices and determinants are applied to full covariance matrices, thus reducing the time complexity. Firstly, the authors denote $C^{-1} = \Lambda$, the precision matrix. Now, the task is to adapt all equations involving C to instead use Λ.

The first adaptation is for the Equation 3.1: (the squared Mahalanobis distance) allows for a direct substitution, yielding the following new equation:

$$d_M^2(x, j) = (x - \mu_j)^T \Lambda_j (x - \mu_j) \tag{3.21}$$

In the next sections, the remaining adaptations are shown.

3.5.1 Fast Learning

In the learning process, Pinto and Engel (2015) proceed to adapt the Equation 3.16 (covariance matrix update). This equation can be seen as a sequence of two rank-one updates (adding a term and then subtracting one) to the C_j matrix and applying the Sherman-Morrison formula (SHERMAN; MORRISON, 1950) Equations 3.22 and 3.23 allow us to update the precision matrix directly, eliminating the need for the covariance matrix C.

$$\bar{\Lambda}_{(t)} = \frac{\Lambda_{(t-1)}}{1-\omega} - \frac{\frac{\omega}{(1-\omega)^2}\Lambda_{(t-1)}e^* e^{*T}\Lambda_{(t-1)}}{1+\frac{\omega}{1-\omega}e^{*T}\Lambda_{(t-1)}e^*} \tag{3.22}$$

$$\Lambda_{(t)} = \bar{\Lambda}_{(t)} - \frac{\bar{\Lambda}_{(t)}\Delta\mu\Delta\mu^T\bar{\Lambda}_{(t)}}{1-\Delta\mu\bar{\Lambda}_{(t)}\Delta\mu^T} \tag{3.23}$$

Now, for the Equation 3.1 is necessary to know the determinant of the covariance matrix C, but in this point, this matrix is unknown (is only known the precision matrix Λ). Since the determinant of the inverse of a matrix is simply the inverse of the determinant, it is sufficient to invert the result. But computing the determinant itself is also a $O(N^3)$ operation, thus, the perform rank-one updates instead, using the Matrix Determinant Lemma (HARVILLE, 1997). Two new equations (3.24 and 3.25) are defined for updating the determinant, reusing the equations 3.22 and 3.23:

$$|\bar{C}_{(t)}| = (1-\omega)^D|C_{(t-1)}|\left(1+\frac{\omega}{1-\omega}e^{*T}\Lambda_{(t-1)}e^*\right) \tag{3.24}$$

$$|C_{(t)}| = |\bar{C}_{(t)}|\left(1-\Delta\mu\bar{\Lambda}_{(t)}\Delta\mu^T\right) \tag{3.25}$$

As the final adaptation in the learning part of the algorithm, it is necessary to define the initialization for Λ for each neuron. What previously was $C_j = \sigma^2_{ini}I$ now becomes $\Lambda_j = \sigma^{-2}_{ini}I$, the inverse of the variances of the dataset. Since this matrix is diagonal, there are no costly inversions involved. The determinant $|C|$, may be initialized as $\prod \sigma^2_{ini}$, which again takes advantage of the initial diagonal matrix to avoid costly operations. Note that we keep the precision matrix Λ, but the determinant of the covariance matrix C instead.

3.5.2 Fast Recall

In the recall operation, we have one equation that involve the covariance matrix C, the Equation 3.19, in fact, this equation does not involve directly the covariance matrix, but involves the product of two sub matrices of C_j: $C_{j,ti}$ and $C_{j,i}$, showed in Equation 3.20. This can be accomplished by the use of a block matrix decomposition (showed in the equation 3.26) (the i subscripts stand for "input", and refers to the input portion of the covariance matrix, i.e., the dimensions corresponding to the known variables; similarly, the t subscripts refer to the "target" portions of the matrix, i.e., the unknowns; the it and ti subscripts refer to the covariances between these variables):

$$\begin{aligned}\Lambda &= \left(\begin{array}{c|c} C_{j,i} & C_{j,it} \\ \hline C_{j,ti} & C_{j,t} \end{array}\right)^{-1} \\ &= \left(\begin{array}{c|c} \Lambda_{j,i} & \Lambda_{j,it} \\ \hline \Lambda_{j,ti} & \Lambda_{j,t} \end{array}\right) \\ &= \left(\begin{array}{c|c} (C_{j,i} - C_{j,it}C_{j,t}^{-1}C_{j,ti})^{-1} & -C_{j,i}^{-1}C_{j,it}(C_{j,t} - C_{j,ti}C_{j,i}^{-1}C_{j,it})^{-1} \\ \hline -C_{j,t}^{-1}C_{j,ti}(C_{j,i} - C_{j,it}C_{j,t}^{-1}C_{j,ti})^{-1} & (C_{j,t} - C_{j,ti}C_{j,i}^{-1}C_{j,it})^{-1} \end{array}\right)\end{aligned} \tag{3.26}$$

Looking at the decomposition, it is clear that $\Lambda_{j,it}\Lambda_{j,t}^{-1} = -C_{j,i}^{-1}C_{j,it} = -C_{j,ti} - C_{j,t}^{-1}$ (the terms between parenthesis in $\Lambda_{j,ti}$ and $\Lambda_{j,t}$ cancel each other, while due to symmetry $C_{j,ti} = C_{j,it}^{T}$). And finally the Equation 3.19 can be rewritten as:

$$\widehat{x}_t = \sum_{j=1}^{K} p(j|x_i)(\mu_{j,t} - \Lambda_{j,it}\Lambda_{j,t}^{-1}(x_i - \mu_{j,i})) \tag{3.27}$$

In Pinto and Engel (2015), authors do not clearly specify how the conditional probability $p(j|x_i)$ was calculated from the equation 3.19, since this calculation depends on the inverse of the sub matrix $C_{j,i}$ and his determinant (for compute the multivariate probability density function). Therefore, this work adopts the following strategy to implement these operations.

Since $C_{j,i} = (\Lambda_{j,i} - \Lambda_{j,it}\Lambda_{j,t}^{-1}\Lambda_{j,ti})^{-1}$ (for substitution in the Equation 3.26), we can compute directly:

$$C_{j,i}^{-1} = \Lambda_{j,i} - \Lambda_{j,it}\Lambda_{j,t}^{-1}\Lambda_{j,ti} \tag{3.28}$$

Note that $\Lambda_{j,t}$ was inverted in the Equation 3.27, we do not need to invert it again. Now, for the determinant calculation we use the method for expressing the determinants of block matrices (POWELL, 2011) (Equation 3.29):

$$\det(C_j) = \det\begin{pmatrix} C_{j,i} & C_{j,it} \\ C_{j,ti} & C_{j,t} \end{pmatrix} = \det(C_{j,i}) \cdot \det(C_{j,t} - C_{j,ti}C_{j,i}^{-1}C_{j,it}) \tag{3.29}$$

From the Equation 3.26, we have $\Lambda_{j,t} = (C_{j,t} - C_{j,ti} C_{j,i}^{-1} C_{j,it})^{-1}$, and with substitution in the Equation 3.29, we can write $|C_{j,i}|$ as:

$$|C_{j,i}| = |C_j| \cdot |\Lambda_{j,t}| \tag{3.30}$$

And we need only compute the determinant of $\Lambda_{j,t}$, since $|C_j|$ is a parameter of the model and we know it.

3.6 Summarizing

3.6.1 Configuration parameters

IGMN has a total of eight configuration parameters, where three of them are related to restarting of the accumulators sp_j:

- $\boldsymbol{sp_{max}}$: a large number to be compared with the accumulation over the time of all sp_j.
- $\boldsymbol{\beta}$: this parameter is used to get a fractional part of sp_{max}
- $\boldsymbol{\gamma}$: a fractional part of the accumulators sp_j that should be maintained after a restart.

The rest of parameters follows:

- $\boldsymbol{\delta}$: this parameter is used to set the initial radius of the covariance matrices, σ_{ini}, related to the domain range. It can take any value in $0 < \delta < 1$.
- $\boldsymbol{\tau_{nov}}$: this parameter defines the level of acceptance for new information of the existing neurons. It is used to create new neurons. It is a fraction of the maximum value of the likelihood function and can take any value in $0 < \tau_{nov} < 1$.
- $\boldsymbol{sp_{min}}$: the minimum activation of a neuron to be not considered as noise. It is used in the process of removing neurons. A natural choice for sp_{min} is $D+1$ (dimension of the input), because, according to Traven (1991), a minimum of $D+1$ samples are required to obtain a nonsingular estimate of an unconstrained covariance matrix.
- $\boldsymbol{v_{min}}$: the number of steps in which neurons can have less activation than sp_{min}. After v_{min} steps, if the neuron has a $sp_j < sp_{min}$, it is considered as noise and is removed. It can be set to any value greater than $D + 1$.

3.6.2 Neuron variables

Each neuron of IGMN has a set of variables, from which the first four are actually inherited from the mixture components. The other variables are accumulators used in the learning mode by the network.

- $\boldsymbol{p(j)}$: the prior probability of the neuron j.
- $\boldsymbol{\mu_j}$: the mean of the neuron j.
- $\boldsymbol{\Lambda_j}$: the precision matrix of the neuron j (inverse of the covariance matrix).
- $\boldsymbol{|C_j|}$: the determinant of the covariance matrix of the neuron j.
- $\boldsymbol{sp_j}$: the accumulator of $p(j|x)$ of the neuron j.
- $\boldsymbol{v_j}$: the accumulator of steps since the creation of the neuron j.

3.6.3 Learning Algorithm

Algorithm 1 presents a detailed pseudo-code description of the IGMN learning mode, which works as follows. Algorithm 3 presents the process when a new neuron is created and Algorithm 2 the process when is just necessary update the parameters of the model. Finally, Algorithm 4 present the pseudo-code description of the remotion of spurious components.

Algorithm 1: Learning Mode

```
Input  : A data point $x$
Function learnIGMN($x$)
    // Compute the squared Mahalanobis distance
    forall neuron $j \in M$ do
        $d_M^2(x,j) = (x-\mu_j)^T C_j^{-1}(x-\mu_j)$
    end
    // Create a new neuron $k$ if necessary
    if $K = 0$ or $\nexists j, d_M^2(x,j) < \chi^2_{D,1-\tau}$ then
        createNeuron($x$);
    end
    updateNeurons($x$);
    removeNeurons();
    // Reset posteriori accumulators if necessary
    if $\sum_{i=1}^{M} sp_i > \beta \cdot sp_{max}$ then
        $sp_j = \gamma \cdot sp_j;\ \forall j$
    end
end
```

Algorithm 2: Update neural network

```
Input  : A data point $x$
Function updateNeurons($x$)
    // Compute the likelihood for all neurons
    $p(x|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|C_j|}} \exp\left(-\frac{1}{2}d_M^2(x,j)\right)$
    // Compute the posteriori probabilities
    $p(j|x) = \frac{p(x|j)p(j)}{\sum_{k=1}^{K} p(x|k)p(k)} \forall j$
    // Update all neurons
    for all neuron $j$ do
        $v_j(t) = v_j(t-1) + 1;$
        $sp_j(t) = sp_j(t-1) + p(j|x);$
        $e_j = x - \mu_j(t-1);$
        $\omega_j = \frac{p(j|x)}{sp_j};$
        $\Delta\mu_j = \omega_j e_j;$
        $\mu_j(t) = \mu_j(t-1) + \Delta\mu_j;$
        $e_j^* = x - \mu_j(t);$
        $\bar{\Lambda}_j(t) = \frac{\Lambda_j(t-1)}{1-\omega} - \frac{\frac{\omega}{(1-\omega)^2}\Lambda_j(t-1)e^* e^{*T}\Lambda_j(t-1)}{1+\frac{\omega}{1-\omega}e^{*T}\Lambda_j(t-1)e^*};$
        $\Lambda_j(t) = \bar{\Lambda}_j(t) - \frac{\bar{\Lambda}_j(t)\Delta\mu\Delta\mu^T\bar{\Lambda}_j(t)}{1-\Delta\mu\bar{\Lambda}_j(t)\Delta\mu^T};$
        $p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q}$
        $|\bar{C}_j(t)| = (1-\omega)^D |C_j(t-1)| \left(1 + \frac{\omega}{1-\omega}e^{*T}\Lambda_j(t-1)e^*\right);$
        $|C_j(t)| = |\bar{C}_j(t)| \left(1 - \Delta\mu\bar{\Lambda}_j(t)\Delta\mu^T\right);$
    end
end
```

Algorithm 3: Create new neuron

```
Input  : A data point $x$
Function createNeuron($x$)
    $K = K + 1; v_k = 1; sp_k = 1.0; \mu_k = x;$
    $\Lambda_k = \sigma_{ini}^{-1} \cdot I;$
    $|C_k| = |\Lambda_k|^{-1};$
    $p(k) = \frac{1}{\sum_{i=1}^{K} sp_i};$
    $d_M^2(x, k) = 0;$
end
```

Algorithm 4: Remove neurons

```
Function removeNeurons()
    forall neuron $j \in M$ do
        if $v_j > v_{min}$ and $sp_j < sp_{min}$ then
            delete the $j$-th neuron
        end
    end
end
```

3.6.4 Recalling Algorithm

Algorithm 5 presents a detailed pseudo-code description of the IGMN recalling mode.

Algorithm 5: Recall Mode

```
Input  : A data point $x_i$
Function recallIGMN($x$)
    // Compute the squared Mahalanobis distance
    forall neuron $j \in M$ do
        $d_M^2(x_i, j) = (x_i - \mu_{j,i})^T \Lambda_{j,i}(x_i - \mu_{j,i})$
    end
    // Compute the likelihood for all neurons
    $p(x_i|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|C_{j,i}|}} \exp\left(-\frac{1}{2}d_M^2(x, j)\right)$
    // Compute the posteriori probabilities
    $p(j|x_i) = \frac{p(x_i|j)p(j)}{\sum_{k=1}^{K} p(x_i|k)p(k)} \forall j;$
    // Compute the estimate $\hat{x}_t$
    $\hat{x}_t = \sum_{j=1}^{K} p(j|x_i)(\mu_{j,t} - \Lambda_{j,it}\Lambda_{j,t}^{-1}(x_i - \mu_{j,i}));$
end
```

4 RELATED WORKS

When designing a drift handling approach, very often the choice between single learners and ensemble learners may give rise to many questions. Hence, in this chapter, the principles, challenges, and opportunities regarding both single and ensemble learners are studied.

4.1 Single classifier approaches

In a non-stationary environment, single adaptive (or dynamic) learning algorithms are widely used for handling concept changes.

An extension of *incremental algorithms*, especially when they incorporate forgetting mechanisms in order to discard data from outdated concepts. This was initially known as decremental learning (CAUWENBERGHS; POGGIO, 2000). The idea behind decremental learning is to integrate forgetting capacity with resultant reduction in weight of past data and unlearning of obsolete information due to environment changes.

Another possible extension of the incremental learning algorithms is to be self-adaptive. The objective is to monitor the learning process, adapt the model, and interpret the encountered changes in order to respond to the new environmental requirements. The development of such a self-configuring, or self-repairing learner is a major scientific and engineering challenge. The main issues are: (i) knowing how to track concept drift and (ii) knowing how to adapt the learner parameters and structure in order to react according to these new environment requirements.

In the next sections, we discuss the most popular single classifiers used to classify streaming data. In Section 4.1.1 we discuss the use of windows to model the forgetting process, where, of necessity, they react to concept drift. Drift detectors i.e. wrapper methods to allow for the rebuilding of classifiers when necessary, are presented in Section 4.1.2. Finally, in Section 4.1.3, we present the learners suggested for stationary classification tasks that can also be employed to classify data streams.

4.1.1 Windowing Techniques

Windowing techniques have been widely used for handling drift problems. They consider that the most recent observations are the most informative, and progressively estimate the change through either a time or a data window. A *window* is a short memory data structure that can store informative data or summarize statistics concerning the model behavior or the data distribution as a mean to characterizing the current concept.

In the basic windowing algorithm, each arrival example is used to update the window and this window is used to update the classifier. The main part of this algorithm is found in the definition of the window, in the way it models the forgetting process (BRZEZINSKI, 2010).

We can define the window in accordance with the following characteristics:

4.1.1.1 Size

1. **Fixed:** In early studies by Widmer and Kubat (1996), Kifer, Ben-David and Gehrke (2004), the window size was fixed a priori. However, by using a small window size, the classifier will react quickly to changes (sudden drift), but may loose on accuracy in periods of stability. By choosing a large sized window, accuracy will be enhanced in periods of stability (gradual drift), but will fail to adapt to rapidly changing concepts.

2. **Variable:** New studies by Zliobaite and Kuncheva (2009), Kuncheva and Zliobaite (2009), Khamassi and Sayed-Mouchaweh (2014), have used windows of dynamic size in order to handle different types of drift. Many methods exist for determining the window size, like statistical hypothesis testing (BIFET; GAVALDà, 2007; KHAMASSI et al., 2015) and control chart with variable change threshold (MEJRI; KHANCHEL; LIMAM, 2013).

4.1.1.2 Positioning strategy

Positioning strategy refers to how the window is evolving during the drift tracking.

1. **Sliding window:** where the learner is periodically updated according to a fixed number of instances stored in the first-in-first-out (FIFO) data structure. In this context, whenever a new instance occurs , it is saved in memory and the oldest one

is discarded (HULTEN; SPENCER; DOMINGOS, 2001).

2. **Landmark window:** starts storing instances from a given time point (timestamp) until a certain condition is reached. For example, the landmark windows in Gama and Castillo (2006) keep storing instances and enlarge their size when there is no change. Once a drift is detected, the window size is reduced by retaining only the most representative data.

4.1.2 Drift Detectors

Drift detectors is another group of algorithms for learning evolving data streams. In general, these algorithms use different strategies to analyse the results of the base classifier in order to detect when concept drifts occurs. If concept drift is detected, the base classifier should be retrained. Drift detectors usually use a confidence level (statistical test) that verifies if the class distribution or running error is constant over time.

Cumulated Sum (CUSUM) (PAGE, 1954) and the Geometric Moving Average (GMA) (ROBERTS, 1959) were two proposed tests designed for numeric sequences. The Kolmogorov - Smirnov test (KOLMOGOROV–SMIRNOV..., 2008) was proposed as a statistical test for more complex populations. CUSUM involves the calculation of a cumulative sum S, such than when this value is greater than a certain threshold value, a change in value has been found. GMA, similar to CUSUM, verifies if the weighted average of examples in a window is greater than a certain threshold value.

Below, we discuss three proposed tests designed for drifting data streams.

4.1.2.1 Drift detection method (DDM)

DDM (GAMA et al., 2004) detects changes in distribution based on the probability of a false prediction of a base classifier p_t and its corresponding standard deviation s_t as in Equation 4.1.

$$s_t = \sqrt{\frac{p_t(1-p_t)}{t}} \tag{4.1}$$

The authors argue that the error rate p_t will decrease when the number of examples increases, and, for a large number of examples, i.e. > 30, the distribution of the examples remains stationary. An increase in the error rate probably suggests changes in the data distribution, and in this case a new base learner should be created.

For each example in the data stream, DDM updates the minimum values of the probability of error (p_{min}) and standard deviation (s_{min}). These minimum values are used in the detection of a warning level presented in Equation 4.2, and a drift level presented in Equation 4.3. The values α and β represent the confidence levels at which the warning and drift signals are triggered. When the warning level is reached, the example is stored in a separate "warning" window. However, when the drift level is reached, a new base learner is created using only examples from the separate window.

$$p_t + s_t \geq p_{min} + \alpha \cdot s_{min} \tag{4.2}$$

$$p_t + s_t \geq p_{min} + \beta \cdot s_{min} \tag{4.3}$$

The authors proposed $\alpha = 2$ and $\beta = 3$, giving approximately 95% and 99% confidence intervals, respectively.

4.1.2.2 Early drift detection method (EDDM)

EDDM (BAENA-GARĆıA et al., 2006) is similar to DDM, but instead of using the classifier's error rate, the authors propose to use the distance between two consecutive error rates. They denote p'_t as the average distance between two consecutive errors and s'_t as its standard deviation. With these values, the new warning and drift conditions are given in Equations 4.4 and 4.5, respectively.

$$\frac{p'_t + 2 \cdot s'_t}{p'_{max} + 2 \cdot s'_{max}} \geq \alpha \tag{4.4}$$

$$\frac{p'_t + 3 \cdot s'_t}{p'_{max} + 3 \cdot s'_{max}} \geq \beta \tag{4.5}$$

The authors claim EDDM is better than DDM to detect gradual concept drifts, but DDM is better suited for abrupt concept drifts.

4.1.2.3 EWMA for Concept Drift Detection (ECDD)

ECDD (ROSS et al., 2012) was adapted from the Exponentially Weighted Moving Average (EWMA) (ROBERTS, 1959) to be used in data streams with concept drifts.

Given that time t, the mean (μ_t) and standard deviation of the data (δ_t) are known in advance, EWMA detects significant changes in the mean of a sequence of random variables using the EWMA estimator (Z_t). This EWMA estimator forms a "recent" estimate of (μ_t), with older data that is progressively reduced.

In addition to the above EWMA estimator (Z_t), ECDD introduces a second estimator denoted by $\hat{p}_{o,t}$. Unlike Z_t, the estimator $\hat{p}_{o,t}$ does not give more weight to recent observations from the stream. As a consequence Z_t is more sensitive to changes and should give an estimate close to its current value, while $\hat{p}_{o,t}$ is less sensitive to change and is thus intended to be an estimate of its pre-change value. When a change occurs, the Z_t estimator should react more quickly and converge towards the new value, whereas the estimator $\hat{p}_{o,t}$ should converge more slowly towards this new value. The EWMA procedure flags for a change whenever the distance between these two estimators exceeds a certain threshold β, i.e. when

$$Z_t > \hat{p}_{o,t} + \beta \tag{4.6}$$

When an object in the stream is sequentially presented to the classifier, estimators Z_t and $\hat{p}_{o,t}$ are updated, and if $Z^t > \hat{p}_{o,t} + \beta$ then it is flagged that concept drift has occurred. Normally, an action will be taken to modify the classifier in response to this, but details of which action to choose depends on the particular classifier being used.

4.1.3 Classic learners

Some classifiers that initially were proposed for static data mining were modified to have the qualities of an online learner, i.e. they are able to process data sequentially. Additionally, with the implementation of a forgetting mechanism, some of them can react to changes. In the following paragraphs we present four learners that fall into these groups:

4.1.3.1 Decision trees

Domingos and Hulten (2000) proposed a decision tree learning algorithm for streaming data, named as Very Fast Decision Tree (VFDT). This learner uses Hoeffding trees to construct the decision tree in constant time for each example, and Hoeffding bounds are used to evaluate how many examples are necessary to decide the split attribute in the decision tree (using information gain or Gini index). The authors show that VFDT gives good practical solutions to form decision trees on data stream. They are able to produce trees almost equivalent to those produced by a conventional batch tree learner.

Hulten, Spencer and Domingos (2001) adapted VFDT for continuously changing data streams, creating the Concept-Adapting Very Fast Decision Tree (CVFDT). Instead of assuming data was generated by a single concept, CVFDT assumes that the data was generated by multiple concepts. CVFDT uses a sliding window of instances, and when a fixed number of new instances arrive, each node updates its relevant statistics. CVFDT identifies concept changes when a better splitting attribute is found: in this case a new sub-tree is learned. If new instances confirm that this new sub-tree has more quality than the original sub-tree, the new sub-tree replaces the original one.

As an alternative to CVFDT, Black and Hickey (2003) proposed the CD3 algorithm. The algorithm assumes that the data will arrive in batches, where the number of examples in the batch is variable. The main idea of CD3 is that, during the learning process, a time-stamp is associated with examples and treated as an attribute. The learning algorithm evaluates the relevance of the time-stamp attribute, and if this attribute is important in the newly built tree, then drift has occurred.

Bifet and Gavaldà (2009) proposed two new methods: the Hoeffding Window Tree (HWT) and the Hoeffding Adaptive Tree (HAT). HWT is similar to CVFDT, but it has two main internal differences. (i) HWT creates sub-trees as soon as change is detected, and (ii) HWT replaces old sub-trees by new sub-trees as soon as there is evidence to support the improvements of the new sub-tree, without needing to wait for fixed number of examples. HAT, in contrast to CVFDT, uses an adaptive window at each internal tree node. With this adaptive window, HAT responds to concept drift more accurately and quickly. Another improvement that HAT brings over CVFDT is that it has theoretical guarantees of performance.

4.1.3.2 Incremental support vector machine

Support Vector Machine (SVM) methods have rarely been used for data stream classification. This is due to the challenges associated with incrementally updating the SVM classification model with an increasing number of records (AGGARWAL, 2014). However, incremental methods for SVM were proposed in the literature.

Two important concepts of the SVM algorithm are: mapping the input data into a high dimensional feature space (called support vectors) and separating the feature vectors with the "maximum margin hyperplane" using a linear optimization algorithm. Also, the SVM algorithm has an important property: it allows classification equivalence on support vectors (SV) sets and the whole training set (ZANG et al., 2014). Using this property, Cauwenberghs and Poggio (2000) proposed Incremental SVM, preserving only the SVs after each training step, and adding them to the training set for the next step. The selection of a training set at each step can be made in different ways according to different situations. This represents one of the problems in the Incremental SVM algorithm: when concept drift happens, old support vectors could be rendered useless.

4.1.3.3 Decision rules

Decision rules models have also been adjusted to the data stream context. Such proposed models are: the family FLORA, and FACIL. These systems learn rules incrementally and provide forgetting mechanisms using dynamic windows.

Kubat (1989) presented FLOating Rough Approximation (FLORA). To deal with concept drift and hidden contexts, FLORA keeps only a window of trusted examples and forms three sets of symbolic descriptions from them: expressions that are true for all positive examples in the window (ADES), expressions that are valid for some positive examples (PDES) and expressions that are valid only for negative examples (NDES). With the intention to respond to abrupt or gradual drift, Kubat (1992) proposed FLORA2, which used heuristics to adapt automatically the size of the window of trusted examples. Widmer and Kubat (1993) presented FLORA3 as an extension of FLORA2, introducing an adaptive window (of trusted examples) which attempts to vary its size to fit the current concept.

Another model based on decision rules was presented by Ferrer-Troyano, Aguilar-Ruiz and Riquelme (2006): the Fast and Adaptive Classifier by Incremental Learning (FACIL). This model uses a partial instance memory where examples are rejected if they do not define a decision boundary.

4.1.3.4 Incremental gaussian mixture models

Similar to the EM algorithm (explained by Dempster, Laird and Rubin (1977)), the Incremental Gaussian Mixture Model (IGMM) (ENGEL; HEINEN, 2011) also follows the mixture distribution modeling. However, its model can be effectively expanded with new components (i.e. concepts) as new relevant information is identified in the data flow. Moreover, IGMM adjusts the parameters of each distribution after the presentation of every single data point according to recursive equations (derived from the Robbins-Monro stochastic approximation method (ROBBINS; MONRO, 1951)). These are approximate incremental counterparts of the batch-mode update equations used by the EM algorithm.

IGMM-CD (IGMM with support to Concept Drift) was proposed by Oliveira and Batista (2015) as an extension of the IGMM, with the intention of tracking concept drift. The authors propose adding a new parameter T to the algorithm, with parameter limits the number of components allowed in the model. Therefore, when there is a very large number of components ($> T$), those ones having the smallest a priori probabilities are eliminated, since the author consider these components as outdated. For the authors, their algorithm is highly dependent of the parameter T, since it is not adaptive and needs to be defined before to start the learning process.

4.2 Ensemble Approaches

In recent decades, the ensemble learners have received much attention as they have shown better generalization abilities than single learners. The strength of ensemble learners lies in their ability to ensure a good combination of individual learners, i.e. one learner's strength can make up for the drawback of another (GOMES et al., 2017). In the same way, ensemble learners appear to provide a promising approach for tracking evolving data streams. Given their modularity, ensemble learners naturally adapt to change, either by modifying their structure, retraining ensemble members (individual classifiers), or updating decision rules (BRZEZINSKI; STEFANOWSKI, 2014).

Kuncheva (2004) proposes to group "ensemble solutions" for changing environments as follows:

$a)$ **Dynamic combiners:**
This group is composed by the "horse racing" algorithms, where individual classifiers (experts of the ensemble method) are trained in advance and changes in the environment are modeled by changing the combination rule. Following the horse racing analogy used by Brzezinski (2010) to explain this method, a person (ensemble classifier) wants to predict the outcome (class label) of each horse race (example). This person has k expert friends (ensemble members) and he can acknowledge or ignore their predictions based on a set of weights. When the outcome is known, the person notes which the experts has made correct (or wrong) predictions and

updates their trustworthiness. Such horse racing approaches include: Weighted Majority algorithm proposed by Littlestone and Warmuth (1994), Hedge by Freund and Schapire (1997), Winnow by Littlestone (1988), and Mixture of experts by Jacobs et al. (1991). This approach may not always be appropriate for changing data streams, because the individual classifiers are not re-trained at any stage and, after some time, the experts may become inadequate.

b) **Updated training data for online classifiers:**

In this group, ensemble methods use recent data to make updates in the individual classifiers. Different from the horse racing approaches, the learning process may or may not need to change the combination rule. Usually, methods in this group are variants of batch methods for stationary environments.

In this approach the use of three ways to form the training sets have been presented in literature, i.e. reusing examples (OZA, 2001), filtering examples (BREIMAN, 1999) and using data chunks (GANTI; GEHRKE; RAMAKRISHNAN, 2002).

Oza (2001) proposed two algorithms that reuse the data points. In the first one, the "online bagging" algorithm, base classifiers are incremental learners that combine their decision using a simple majority vote. To form the training set (for retraining the incremental learners), each training example is used k times in the training set, where k is defined by a Poison distribution with $\lambda = 1$.

The second one, the "online boosting" algorithm, is also guided by the Poisson distribution, but different from the previous approach, the parameter λ changes from a classifier to the next.

c) **Structural changes of the ensemble:** In this group, individual classifiers can be replaced with a new classifier trained with recent data. The choice of the classifier that will be replaced can be done using different strategies. The simplest strategy removes the oldest member of the ensemble. A more sophisticated strategy was proposed by Wang et al. (2003), which uses the most recent "chunk" of data, and uses it to evaluate all members of the ensemble. Another strategy was proposed by Street (2001), which uses a "quality score" based on the "merit to the ensemble" of each classifier. This uses, instead, a score based on its individual accuracy. This score is a soft version of the Winnow approach (LITTLESTONE; WARMUTH, 1994).

The following sections describe three specific adaptive ensemble algorithms.

4.2.1 Dynamic Weighted Majority (DWM)

Kolter and Maloof (2007) present the Dynamic Weighted Majority algorithm, an ensemble method for tracking concept drift. This is a general method based on the Weighted Majority algorithm (LITTLESTONE; WARMUTH, 1994) and can be used with any online learner in time-changing problems with unknown dynamics. The Dynamic Weighted Majority (DWM) maintains an ensemble of predictive models (the experts), each with an associated weight. Experts can be generated by the same base learning algorithm, but are generated at different time steps so that use a different training sets of examples.

The DWM algorithm dynamically creates and removes experts in response to changes in the performance of the ensemble. Whenever an example arrives, each expert is consulted and a prediction is made. The final prediction is obtained as a weighted vote of all the experts. For each class, DWM sums the weights of all the experts predicting that class, and predicts the class with the greatest weight. The learning process of DWM first predicts the classification of the training data, and then weights of all the experts that were wrong in their predictions are decreased by a multiplicative constant value. If the overall prediction of the ensemble was incorrect, a new expert is added to the ensemble with weight equal to the total weight of the ensemble. Finally, all the experts are trained by the example.

If trained on a large amount of data, DWM has the potential to create a large number of experts. Ideally, we should be able to remove, or prune the oldest experts while keeping the best experts. Doing this will not diminish accuracy. A possible pruning rule removes the oldest expert when the number of experts is greater than a constant k. Other pruning rules may be used. For example, when a new expert is added to the ensemble, if the number of experts is greater than a constant K, the expert with the lowest weight will be removed before adding the new member.

Recent research based on DWM is **Dynamic Weighted Majority for Imbalance Learning (DWMIL)** (LU YIU-MING CHEUNG, 2017) which deals with the data streams with concept drift and class imbalance problems. DWMIL utilizes an ensemble framework by dynamically weighting the base classifiers according to their performance on the current data chunk.

4.2.2 Learn++.NSE

Elwell and Polikar (2011) proposed Learn++.NSE as an extension of Learn++ (POLIKAR et al., 2002) which is applicable to Non-Stationary Environments. Learn++.NSE can accommodate a wide variety of drift settings, regardless of the type of drift, which is a challenge in most learning algorithms trying to deal with non-stationary environments. This algorithm trains one new classifier for each "chunk" of data it receives, and combines these classifiers using a dynamically weighted majority voting. According to its authors, the novelty of the approach is based on determining the voting weights for each classifier's time-adjusted accuracy on current and past environments. With the intention of tracking a possible recurrence of an earlier distribution, in Learn++.NSE all classifiers are maintained and reweighted on the most recent training data, resulting in a linear learning time increase.

In order to investigate the impact of retaining all classifiers, the authors assess the effects of pruning the ensemble (ELWELL; POLIKAR, 2009a; ELWELL; POLIKAR, 2009b). Based on experimentation, they noted that error-based pruning is always preferable to age-based pruning (e.g. removal of the oldest member), even in the presence of abrupt concept drift. It was also noted, however, that neither pruning strategy effectively dealt with situations of reoccurring concepts, since the ensemble cannot determine which concepts are going to reoccur, and thus which classifiers to retain. In consideration of this, the authors recommend retaining as many models as possible if reoccurring concepts are likely.

Ditzler and Polikar (2010) proposed a method for extending the Learn++.NSE algorithm in the case of class imbalance. The authors propose Learn++.NIE (Learning in Non-stationary and Imbalanced Environments). In Learn++.NIE, a step to the logic of the Learn++.NSE algorithm was added, using bagging instead of a single ensemble member. With this additional step, the authors claim they can both reduce error via bagging, and, more importantly, learn on a less imbalanced dataset by under-sampling the majority class when creating each bag (DITZLER; POLIKAR, 2010).

4.2.3 Online Accuracy Updated Ensemble (OAUE)

OAUE (BRZEZINSKI; STEFANOWSKI, 2014) algorithm is a ensemble method, which combines the block-based and online ensemble methods. Block-based methods were designed to work in environments were examples arrive in "blocks" or "chunks" (GAMA; GABER, 2007). The authors highlighted the comparison of three different ways to transform block-based methods to online methods and based on these comparisons and their previous research in Accuracy Updated Ensemble method (BRZEZIŃSKI; STEFANOWSKI, 2011), OAUE was presented.

OAUE maintains a weighted set of k component classifiers, such that the weighting is given by an adaptation to incremental learning of the weight function presented in (BRZEZIŃSKI; STEFANOWSKI, 2011). In contrast the old algorithm, OAUE does not use static batch learners and does not divide the stream into blocks. Meanwhile, the accuracy-based weighting method is utilized and periodically (each n examples) removes the poorest ensemble member and it is replaced by a candidate classifier, which has been trained only in the last n examples. As a result, OAUE method have a good performance in most situations.

GU et al. noted that OAUE method has a good reaction to real-time applications, but loses some accuracy when a sudden concept drift happens inside the window. Gu et al. (2014) presented WAOAUE (Window-Adaptive Online Accuracy Updated Ensemble) based on OAUE, where a change detector is added to the ensemble to decide the window size of each candidate classifier.

4.3 Conclusions

- Using single learner approaches is of interest in controlling the complexity of the system, as they were designed to be adapted in real time and with minimum computational efforts.

- Ensemble training is a costly process. It requires at least k (number of ensemble members) times more processing than the training of a single classifier, plus member example selection and weight assignment, all of which usually make the training process even longer.

- Incremental learning only can be used for some classification algorithms. In contrast, ensemble algorithms can use as base learners any classification algorithms.

- Ensemble algorithms are more flexible and naturally adapt themselves better to concept drift.

- When the data stream is relatively simple or static, i.e. there is no concept drift, an incremental algorithm is recommended, but when the data stream is continuously experiencing concept drift, ensemble algorithms are recommended to guarantee accuracy.

- Single learner approaches are not recommended for handling recurrent concept drift, as the model is continuously adapted to the current concept. When a previous concept reoccurs, these approaches relearn it from scratch without taking benefit from its previous existence.

5 PROPOSED ALGORITHM

This chapter describes in detail, the model proposed in this work. It is named Incremental Gaussian Mixture Model for Non-Stationary Environments (IGMN-NSE). This model is based on the IGMN model and improves its capacity for classification tasks while implementing a forgetting mechanism with the intention of tracking concept drift.

This chapter is structured as follows: Section 5.1 describes the difficulties of IGMN when this is used in a non-stationary environment. Section 5.2 presents relevant definitions to be considered in our proposed model. Section 5.3 identifies the main issues of IGMN in non-stationary environments and describes the characteristics that our model, IGMN-NSE, will be implementing . Finally, Sections 5.4 and 5.5 describes in detail our contributions.

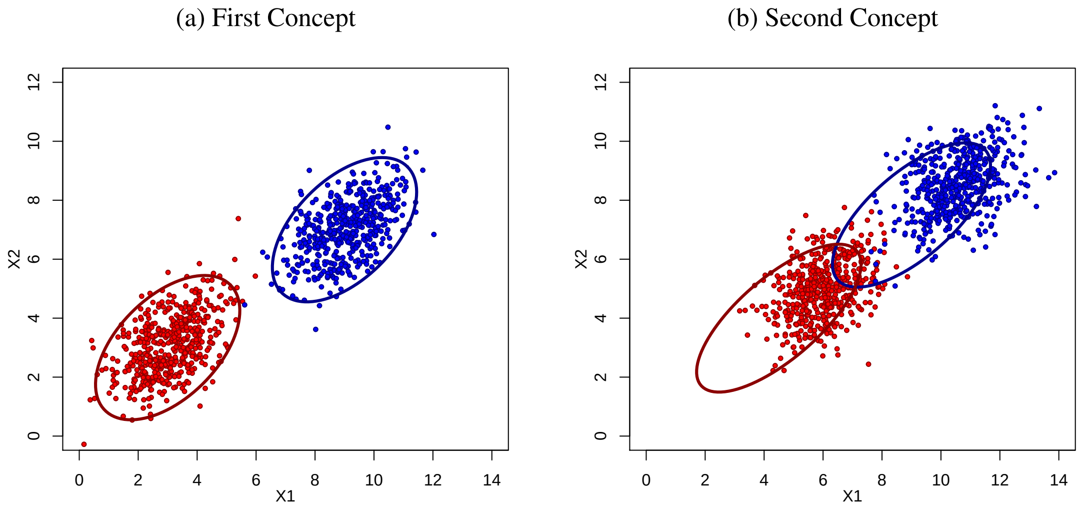
5.1 Difficulties of IGMN in non-stationary environments

IGMN was created for dealing with function approximations. In this context, the data has a static behavioral pattern and the learning process permits the creation of a stable GMM (inside IGMN). All gaussian components are of importance in the model and all parameters of the GMM have correct environment information. In non-stationary environments, the data stream presents changes in the distributions, and for a GMM, it is mandatory to react to the changes and adapt the model (all parameters should be updated with up-to-date information of the environment).

Figure 5.1 shows an example of how *outdated concepts* create a challenging problem for IGMN. Suppose the *first concept* of the environment is learned in the first training time t_1. IGMN creates two gaussian components (denoted by ellipses) in relation to two different distributions on the data: distribution 1 is denoted by red dots, and distribution 2 is denoted by blue dots as shown in Figure 5.1(a).

Now, suppose that in the second training time t_2, a *second concept* appears and both distributions experience changes in the same direction, as shown in Figure 5.1(b). IGMN in t_2 maintains complete information of the t_1 and tries to assimilate information of the time t_2. The current GMM of IGMN does not reflect the correct information of the distribution and produces misclassification of the data. Concerning these issues, the following problems are studied in this work:

Figure 5.1: An example of how outdated data creates a difficult situation for IGMN.


(a) First Concept

(b) Second Concept

1. When is it necessary to create or remove gaussian components if the data is continually changing?

2. How is up-to-date information maintained in the model without loss of information learned in the past?

3. How do you identify when a gaussian component does not reflect current information from a distribution?

Some relevant definitions used in this dissertation will be given in the next section.

5.2 Relevant Definitions

Gaussian components of IGMN experience different states in a data stream depending on their adaptation to the environment. Relevant states are defined below:

Definition 1. A *spurious component* IGMN is a component that, from v_{min} examples learned since its creation, it did not become *significant* to the model. Its significance is defined by its posteriori accumulator sp_j.

Definition 2. A *stable component* for IGMN is a component that became significant to the model. From v_{min} examples learned since its creation, its posteriori accumulator is greater than sp_{min}.

Definition 3. An *outdated component* for IGMN is a component that at some point in the data stream was considered *stable*, but for the current data, it does not represent correct environment information.

Definition 4. An *up-to-dated component* for IGMN is a component that, at some point in the data stream, was considered *stable* and for the current data, it still represents correct environment information.

Based on the Definitions 1 and 2, an important relationship was noted: the *significance* of a component for IGMN is proportional to the accumulation of information in the posteriori accumulators variables, since these variables are used to calculate the priori probabilities. In addition, based on the Definitions 3 and 4, it can be noted that a component of IGMN can be freely changed from a stable state to an outdated or up-to-date state.

5.3 Problem formulation and proposed solution

In Section 2.1.3, two important requirements for a data stream classifier were identified and we considered the use of "online learning" and implementing a "forgetting mechanism" in our IGMN-NSE model. IGMN possesses an incremental nature and can incrementally induce and adapt a model over time, according to the incoming data. However, it is not ideal for dealing with concept drift, since it does not explicitly provide a mechanism to forget outdated concepts. Outdated concepts can potentially overlap new concepts in the feature space, which, in turn, raises the uncertainty of the data. If the data distribution changes, the information from the old data needs to be adapted to fit with the new data (ZLIOBAITE, 2010). With this in mind, the following issues of IGMN will be addressed in IGMN-NSE:

1. IGMN does not adapt faster to the presence of concept drift.

2. IGMN does not implement a forgetting mechanism for recognizing when a component changes from a stable to an outdated state.

The first issue to solve relates to the capacity of IGMN to learn in the context of data stream classification. In other words, in our proposed model IGMN-NSE, we need to improve IGMN for data stream classification considering that new concepts can appear in the data stream, and it is important for us know when it is necessary to create, update or remove a component according to its situation.

The second issue to solve is the representation of the current concept in the GMM of IGMN. In IGMN-NSE, we need to implement the capacity to recognize which components are not representing current environment information (outdated components) and reduce their importance in the model. If a component was stable in a period of data streaming but currently represents outdated environment information, its posteriori probability needs to decrease in order to show that for the current data distribution, this component does not represent the up-to-date information for the current concept.

The detail of the proposed model is explained in the following sections. Section 5.4 describes the modifications in IGMN with the intention of improving its performance in classification tasks. Section 5.5 presents the algorithm for forgetting outdated concepts of IGMN-NSE with the intention of tracking concept drift.

5.4 Improving the IGMN for classification tasks

5.4.1 Supervised learning

Most applications involving classification of data streams with concept-drift assume that the true labels of the classified examples will be available immediately after classification or with a slight delay (OLIVEIRA; BATISTA, 2015). Although this is a legitimate assumption, it is in accordance with existing algorithms (KOLTER; MALOOF, 2007; ELWELL; POLIKAR, 2011; BRZEZINSKI; STEFANOWSKI, 2014). In this dissertation we are considering a supervised classification, and we assume that true labels are available during the learning process.

5.4.1.1 Classifying incoming data

Using the recall mode, IGMN can return a predicted label for each training sample. IGMN does not use the words "input" and "output" to represent the data features of a training sample such as $x^t = \{x_i^t, x_t^t\}$, for example. Instead, IGMN considers that the data vectors x_i^t and x_t^t are different sensory and/or motor modalities with distinct domains, and one modality (e.g. x_i^t) can be used to estimate another (e.g. x_t^t). In the rest of this dissertation, we considered that x_i^t is our input data and x_t^t the output data (target).

5.4.1.2 Novelty Criterion

Once a classification is provided, we assume that we will receive the true class label x_t^t, for the training data x^t and we are in the position to adjust the neural network parameters with the learning mode of IGMN. This sequence of events is very important for updating the model, in order to form the input $x^t = \{x_i^t, x_t^t\}$.

When a new training vector x^t arrives, the algorithm then decides if it is necessary to create a new neuron for the current data point x^t based on the criterion explained in Section 3.3.1. If this criterion proves false, the new training vector x^t is not considered represented by any neuron. In this case, it creates a new neuron and centres it on x^t.

In this dissertation we modify the original criterion proposed. Even when x^t meets the novelty criterion of IGMN, if the predicted label for the model $\hat{x_t}^t$ is different from true label x_t^t, a new neuron must be created. The intention of this modification is to create a more agile convergence and adaptation of the model to the concept drift since with modifications of data distributions, outdated concepts can potentially overlap new concepts in the feature space. Thus, we assume that a misclassification in IGMN probably represents changes in the data.

However, this course of action presents a difficulty. A lot of neurons will probably be created in response to the additional novelty criterion ($x_t^t \neq \hat{x_t}^t$). To take account of this, a new condition will be introduced, i.e., if the training data meets with additional novelty criterion, a new neuron is created only if all neurons of that model have an age greater than the parameter v_{min}. With this restriction, we have a stability criterion controlling the number of neurons (granularity of the model). In fact, this stability criterion was first proposed by Engel and Heinen (2011) using the original algorithm IGMM, but it was not needed in IGMN as it implement a different novelty criterion. The modified learning process is presented in the Algorithm 6.

5.4.2 Overlapping components

One problem that we detect when a new component is created (the creation of a new neuron on the IGMN represents the creation of a new component in the Gaussian Mixture Model inside IGMN), is that the initial component size depends on the parameter δ. This not only places huge responsibility and sensitivity on a single hyper parameter, but it also results in inappropriate component sizes during the process. Inappropriate, in

Algorithm 6: Learning Mode IGMN-NSE

```
    Input  : A data point $x = \{x_i, x_t\}$
Function learnIGMN($x$)
    $\hat{x}_t$ = recallIGMN($x_i$);
    // Compute the squared Mahalanobis distance
    ...
    // Create a new neuron $k$ if necessary
    if $K = 0$ or $(\nexists j, d_M^2(x, j) < \chi^2_{D,1-\tau})$ or $(x_t \neq \hat{x}_t$ and $\nexists j, v_j < v_{min})$ then
        createNeuron($x$);
    end
    updateNeurons($x$);
    removeNeurons();
    // Reset posteriori accumulators if necessary
    ...
end
```

this context, would mean overlapping areas of old components or too small components in large unoccupied spaces. While the first case can produce catastrophic forgetting, the second one leaves the model prone to overfitting.

In this dissertation, we propose to add an initial size component adaption relative to the degree of overlap between the new component created and all the old components. If we detect a considerable overlap (defined as a threshold), we will reduce the size of the new component (with a view to reducing the overlap).

In general, the definition for the overlap rate between two gaussian components implements the following principles: (1) the value of the overlap rate lies within a normalized interval such as $[0, 1]$; (2) the overlap rate tends to decrease $(\rightarrow 0)$ as the two components become more widely separated; and (3) the overlap rate increases $(\rightarrow 1)$ as the two components become more strongly overlapped.

In the literature, different approaches for measuring the degree of overlap between two gaussian components are proposed (FUKUNAGA, 1990; AITNOURI et al., 2002; SUN; WANG, 2011). In this dissertation, we studied two of them: the Bhattacharyya coefficient and the Algorithm COLR.

5.4.2.1 The Bhattacharyya coefficient

The Bhattacharyya coefficient (BHATTACHARYYA, 1943) is a divergence-type measure which has an intuitive geometric interpretation (COMANICIU; RAMESH; MEER, 2003). Moreover, it is the most popular technique used to estimate the boundary of the classification error, i.e. the overlap between two distributions (FUKUNAGA, 1990). A small Bhattacharyya coefficient means a small overlap between two distributions, which may lead to a small classification error. For two gaussian distributions with mean vectors μ_i and covariance matrices C_i, the Bhattacharyya coefficient is:

$$B_C = \frac{1}{\exp(B_D)} \tag{5.1}$$

Where B_D is the Bhattacharyya distance (FUKUNAGA, 1990), defined as:

$$B_D = \frac{1}{8}(\mu_1 - \mu_2)^T C^{-1}(\mu_1 - \mu_2) + \frac{1}{2}\ln\left(\frac{|C|}{\sqrt{|C_1||C_2|}}\right) \tag{5.2}$$

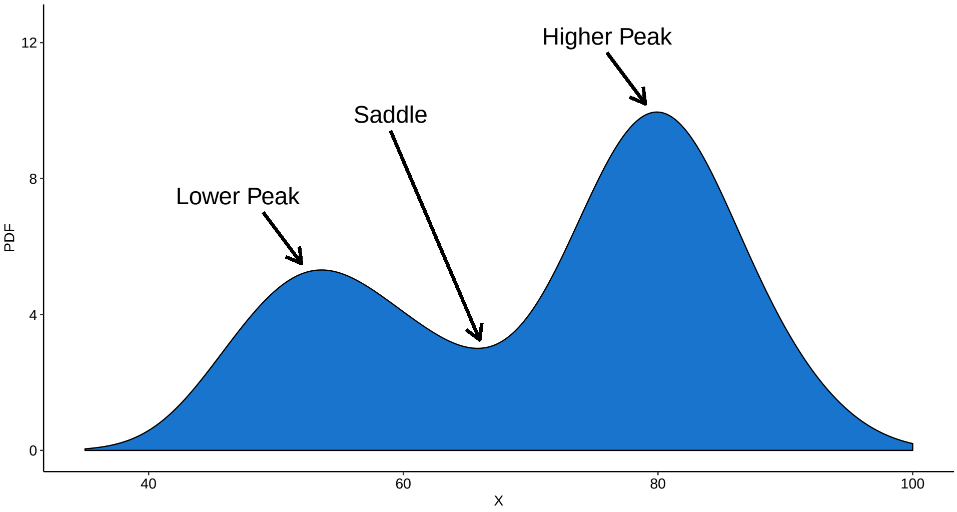
Where:

$$C = \left(\frac{C_1 + C_2}{2}\right) \tag{5.3}$$

5.4.2.2 Algorithm COLR

CORL (SUN; WANG, 2011) is a geometric method for determining the overlap rate (OLR) between two gaussian distributions. The OLR is the ratio between the PDF saddle point of a gaussian mixture distribution and the lower peak point of PDF in this distribution. Figure 5.2 depicts different elements of this definition and illustrates a case of relatively high overlap rate where the saddle point is high compared to the lower peak.

Figure 5.2: The saddle and the peaks of PDF, used in definition of OLR.



The main idea behind the algorithm is to search the segment between the means of the two components (the "ridge curve"). A local maximum point is a PDF peak, and the minimum point is a *saddle point*. Scanning the "ridge curve" is an iterative process with n steps. In each step, the PDF for the current point is calculated to find the maximum and minimum points of $p(X)$. In Sun and Wang (2011), the authors proposed $n = 1000$. Finally, the OLR is defined by:

$$OLR = \frac{p(X_{Saddle})}{p(X_{Sub\_Max})} \tag{5.4}$$

Where $p(x)$ was defined in the equation 3.2.

The authors proposed that the two components (clusters) are (visually) well separated if the value of OLR is less than 0.6; they are partially overlapping, if the OLR belongs to $(0.6, 0.8]$; and they are strongly overlapping, if the OLR is greater than 0.8.

5.4.2.3 Proposed algorithm

Our proposal is that when a new gaussian component j is created:

1. Its initial covariance matrix C_j is created with a value of $\sigma_{ini}^2 I$, where σ_{ini} can be obtained by the Equation 3.8.

2. The overlap rate is computed relative to old components to form the set $O_k, \forall\, k = 1, 2, ..., (j-1)$.

3. If the maximum overlap rate calculated O_{max},

$$O_{max} \geq O_k, \forall\, k = 1, 2, ..., (j-1) \tag{5.5}$$

is greater than a new parameter ϕ, the initial covariance matrix for this new component is recalculate as $C_j * (1 - O_{max})$. If, however, O_{max} is smaller than ϕ, the new component maintains the current covariance matrix.

This modification would leave the user free to set larger δ values, reducing their sensitivity and avoiding very small components. A visualization of this mechanism is shown in Figure 5.3.

Both Bhattacharyya coefficient and COLR can be used to determine the overlap rate between two gaussian components, but they have their own advantages and disadvantages as described in the Table 5.1:

- In terms of complexity, COLR requires more operations to calculate the overlap rate, because in each iteration, it is necessary to calculate the PDF, and this oper-

Figure 5.3: Visualizing the proposed component initial size adaptation. **Left**: the model is composed by one gaussian component and new data (the cross) arrives. **Middle**: How the current component initial size would behave, considering large enough δ. It overlaps the existing component, resulting in catastrophic forgetting in its topmost region. **Right**: How the proposed component adaptive initialization would behave. The initial size is reduced to avoid overlap.

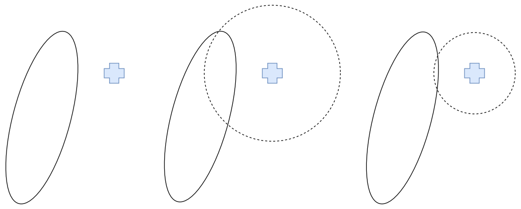


Table 5.1: Comparison of methods for measuring the overlap rate between two distributions.

Characteristic	**Bhattacharyya C.**	**COLR**
Complexity	$O(D^3)$	$O(N \cdot D^3)$
Threshold value	No	Yes

ation has complexity $O(D^3)$ (where D represents the dimensionality of the data). On the other hand, the Bhattacharyya coefficient needs only to compute the formula presented in the Equation 5.2.

- COLR has specific thresholds to determine the overlap rate (more than 0.8 represents a strong overlap while between 0.6 and 0.8 a weak overlap). On the other hand, although the Bhattacharyya coefficient does not provide a specific threshold to determine the overlap rate, we know that a coefficient near to 1 represents a strong overlap, while nearer 0 suggests that the components are more separated.

Since we will use the initial component size adaptation when a new component is created, a process of reasonable complexity is necessary and, from this point of view, the Bhattacharyya coefficient is the most suitable solution. One problem that we need to resolve is to find a threshold to determine when the components experience partial and strong overlaps.

In this dissertation, our proposed solution is to use COLR as a reference to adjust the thresholds for the Bhattacharyya coefficient. In other words, we will calculate the overlap rate for different distributions using OLR and the Bhattacharyya coefficient. With these paired data, an approximate threshold for the Bhattacharyya coefficient is calculated. We generated 12 different normalized Bi-Gaussian distributions with dimensions

Table 5.2: Overlap rate for different distributions using COLR and the Bhattacharyya coefficient.

Distance	$d = 3.0$		$d = 2.5$		$d = 2.0$		$d = 1.0$		$d = 0.5$	
Distribution	COLR	Bhat.	COLR	Bhat.	COLR	Bhat.	OLR	Bhat.	COLR	Bhat.
Bi-Gaussian_2D_1	0.158	0.080	0.472	0.235	0.981	0.504	1.000	0.797	1.000	0.928
Bi-Gaussian_2D_2	0.128	0.072	0.412	0.208	0.895	0.439	1.000	0.689	1.000	0.810
Bi-Gaussian_2D_3	0.179	0.088	0.509	0.253	1.000	0.539	1.000	0.848	1.000	0.987
Bi-Gaussian_3D_1	0.037	0.017	0.229	0.096	0.779	0.334	1.000	0.707	1.000	0.908
Bi-Gaussian_3D_2	0.011	0.005	0.112	0.052	0.578	0.250	1.000	0.644	1.000	0.882
Bi-Gaussian_3D_3	0.003	0.001	0.045	0.025	0.373	0.187	1.000	0.628	1.000	0.941
Bi-Gaussian_4D_1	0.000	0.000	0.000	0.000	0.019	0.027	0.491	0.256	1.000	0.542
Bi-Gaussian_4D_2	0.000	0.000	0.004	0.002	0.140	0.067	0.989	0.466	1.000	0.877
Bi-Gaussian_4D_3	0.020	0.009	0.148	0.063	0.649	0.259	1.000	0.604	1.000	0.800
Bi-Gaussian_5D_1	0.000	0.000	0.013	0.239	0.097	0.001	1.000	0.539	1.000	0.953
Bi-Gaussian_5D_2	0.000	0.000	0.005	0.003	0.136	0.070	1.000	0.426	1.000	0.775
Bi-Gaussian_5D_3	0.000	0.000	0.000	0.000	0.003	0.003	0.410	0.205	1.000	0.822

of $2, 3, 4$ and 5, and the centers of these distributions are modified to generate different overlap rates in our experimentation (caused by varying the distance between cluster centers). Table 5.2 summarizes the overlap rate calculated from COLR and the Bhattacharyya coefficient.

We noted that for these distributions different overlap rates are generated using COLR and the Bhattacharyya coefficient, but these results have a similar behavior, since the overlap rate tends to 1 with a short distance, and it tends to 0 for a long distance. With the tabulated data presented in Table 5.2, for the Bhattacharyya coefficient tabulated for OLR, we use interpolation to estimate the Bhattacharyya coefficient for the values 0.6 and 0.8 of COLR (thresholds that represent weak overlap and strong overlap). Figure 5.4 shows that 0.25 and 0.35 are the equivalent thresholds of the Bhattacharyya coefficient to recognize weak overlap and strong overlap.

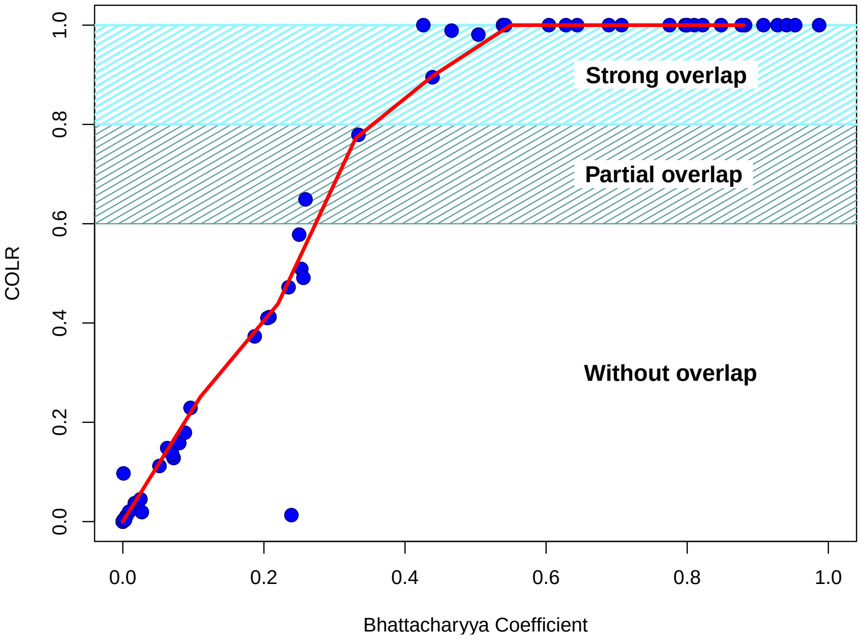
Finally the component initial size adaptation is added after the creation of a new neuron. The process is presented in the Algorithm 7.

5.4.3 Feature Selection

5.4.3.1 Beyond the recall mode

An important characteristic of IGMN is the recall mode. IGMN can estimate any feature when the rest of the features that compose the input vector are known. In Figure

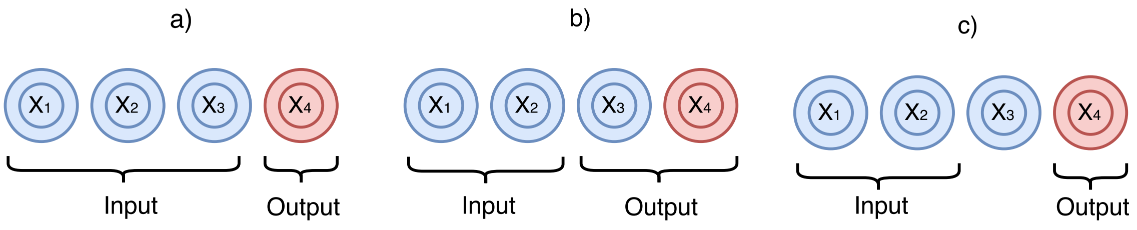
Figure 5.4: Interpolation between overlap rates calculated using CORL and the Bhattacharyya coefficient.



Algorithm 7: Create new neuron IGMN-NSE

```
Input  : A data point $x$
Function createNeuron($x$)
    $K = K + 1$;
    $v_k = 1; sp_k = 1.0; \mu_k = x$;
    $\Lambda_k = \sigma_{ini}^{-1} \cdot I$;
    $p(k) = \frac{1}{\sum_{i=1}^{K} sp_i}$;
    $d_M^2(x, k) = 0$;
    $O_{max} = computeOLR()$;
    if $O_{max} > \phi$ then
        $\Lambda_k = ((1 - O_{max}) \cdot \sigma_{ini} \cdot I)^{-1}$;
    end
    $|C_k| = |\Lambda_k|^{-1}$;
end
```

Figure 5.5: IGMN can estimate any feature from the rest of features. In (a), IGMN estimates x_4 (the feature that represent the class) using $\{x_1, x_2, x_3\}$. In (b), IGMN estimates $\{x_3, x_4\}$(the feature x_3 and the class) using $\{x_1, x_2\}$. In (c), IGMN estimates $\{x_4\}$ (the class) using $\{x_1, x_2\}$ only.



5.5, three different scenarios of estimate features are presented. The first and second ways are deduced from the recall mode, where any feature (or group of features) can be recovered from the rest of the features. In the third way, part of the "input vector" (known part) is used to deduce a part of the rest of the features (unknown part). IGMN can track this problem using one of the following approaches:

- Using the second way and ignoring the feature x_3 of the estimative.
- Using only the two features of the input vector, and estimating directly the feature output (x_4). This is possible since we can reduce the sub-matrices of the covariance matrices corresponding to the known part of the data, ignoring the third feature (x_3).

Although the first solution is more general, and uses the natural recall mode of IGMN, the second solution is more suitable, since we will estimate only the output feature and thereby reduce the complexity of the process (dimensionality is reduced when we ignore one feature). However, the estimation of the "unknown part" depends directly on the information presented in the "known part". If the known part has enough data information, IGMN can calculate a good estimate of the unknown part (HEINEN, 2011). On the other hand, if the known part does not have enough information, the recall mode probably produces a poor estimation. Moreover, in data with high dimensionality, some features of the known part could represent inadequate or irrelevant information (noise) and for IGMN, a good idea is to ignore these features for the estimation of the unknown part. We can understand this process as an internal "feature selection" of IGMN.

Feature selection is the process of identifying and removing as much irrelevant and redundant information as possible. This reduces the dimensionality of the data and may allow learning algorithms to operate faster and more effectively (HALL, 1999).

5.4.3.2 Feature Selection in IGMN

From a machine learning point of view, feature selection for a classification task offers a number of advantages. These include more powerful classification models by eliminating redundant or irrelevant features (BLUM; LANGLEY, 1997), more compact and faster models by constructing them using only a small subset of the original set of features, and the ability to focus on a subset of relevant features, which can be used for the discovery of new knowledge (GUYON; ELISSEEFF, 2003).

Feature selection techniques can be broadly characterized into three classes, depending on how they interact with the estimation of the classification model (SAEYS; INZA; nAGA, 2007). *Filter methods* are independent of the classification model, and select variables (features) as a pre-processing step using properties of the data. In contrast, *wrapper* and *embedded methods* perform feature selection by making use of a specific classification model. While wrapper methods employ a search strategy in the space of possible feature subsets, guided by the predictive performance of a classification model, embedded methods make use of the classification model's internal parameters to perform feature selection. Embedded methods show a better computational complexity than wrapper methods, especially in high-dimensional spaces.

With this in mind, in this dissertation we explore feature selection methods for IGMN. Since IGMN is an incremental algorithm and will discover properties of the data incrementally, filter and wrapper methods are inappropriate. This is because filter methods require information of the features before, to process learning, and wrapper methods require large computational complexity to determine a feature subset. Finally, in this scenario, embedded models are attractive because they use a reliable measure of goodness, similar to wrapper methods, but they avoid retraining a predictor for each feature subset explored.

In general, feature selection algorithms perform a search through the space of feature subsets, and, as a consequence, must address four basic issues affecting the nature of the search (BLUM; LANGLEY, 1997):

1. **Starting point**. The start point for a search in the feature subset space can affect the direction of the search.

2. **Search organization**. With N initial features there exist 2^N possible subsets and in this scenario an exhaustive search is not possible. Heuristic search strategies are more feasible than exhaustive ones and can give good results, although they do

not guarantee finding the optimal subset. Section 5.4.3.4 discusses some heuristic search strategies that have been used for feature selection.

3. **Evaluation strategy**. How feature subsets are evaluated is the single biggest differentiating factor among feature selection algorithms for machine learning. In the case of embedded methods, this evaluation depends on the internal parameters of the model.

4. **Stopping criterion**. Depending on the evaluation strategy, a feature selector might stop adding or removing features when none of the alternatives improves upon the "merit" of a current feature subset.

The *starting point* and the *stopping criterion* are strongly related to the heuristic search choice. However, the evaluation strategy needs to use internal parameters of IGMN to obtain information about the data. Figure 5.6 presents the methodology used in our feature selection method. Initially, IGMN is partially trained (as the learning process is a continuous procedure in IGMN, we never consider that the training process is over) using all the features of the data. In the next stage, using the internal parameters of IGMN, a heuristic search and an evaluation strategy, it choses the best feature subset. Finally, IGMN can use this feature subset for its recall mode and for estimating predictions.

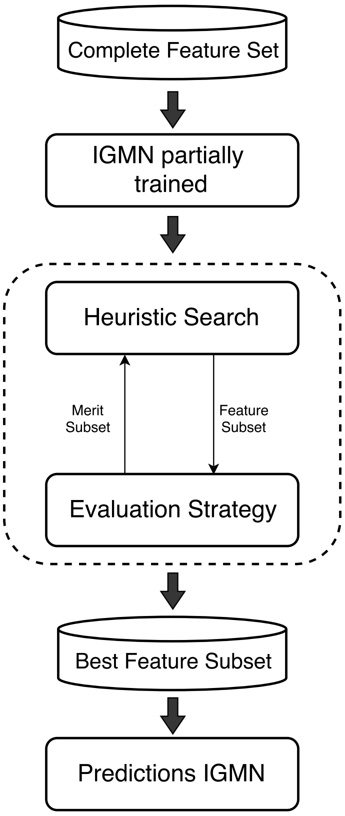
The next subsections present our evaluation strategy and the heuristic search used in the feature selection method for IGMN.

5.4.3.3 Evaluation Strategy

Potential internal parameters of IGMN used in the feature selection process are the covariance matrices. Since the covariance matrix calculates for every combination of variables (features) that tend to show similar behavior (covariance is positive) or opposite behavior (covariance is negative) (WEISSTEIN, 2017), we can choose the redundant or irrelevant variables from there.

The magnitude of the covariance is not easy to interpret because it is not normalized and hence depends on the magnitude of the variables. With this in mind, we use a normalized version of the covariance matrix, i.e. the correlation matrix, that contains the correlation coefficient (degree of dependence or predictability of one variable with another) for every combination of variables. The most common correlation coefficient is Pearson's correlation coefficient, which is obtained by dividing the covariance of the two variables

Figure 5.6: Embedded method for feature selection in IGMN.



by the product of their standard deviations (SNEDECOR; COCHRAN, 1989). Assuming that the covariance matrix of the random variables X is Σ, the correlation matrix (of Pearson's correlation coefficients) can be calculated from the covariance matrix with the Equation 5.6:

$$corr(X) = (diag(\Sigma))^{-\frac{1}{2}}\,\Sigma\,(diag(\Sigma))^{-\frac{1}{2}} \tag{5.6}$$

Where $diag(\Sigma)$ is the matrix of the diagonal elements of Σ.

Since IGMN does not distinguish between input (features) and output (class) variables, the correlation matrices contain the correlation between each feature and the class, and the inter-correlation between each pair of features. However, we need a function to evaluate the "merit" of a feature subset.

In Hall (1999) a heuristic measure of the merit of feature subsets in supervised classification tasks is proposed using correlation coefficients. This heuristic measure is based on the following definitions:

- A feature is useful if it is correlated with or predictive of the class; otherwise it is irrelevant.

- A feature is said to be redundant if one or more of the other features are highly correlated with it.

The above definitions are summary in:

A good feature subset is one that contains features highly correlated with the predicted value, yet uncorrelated with the other features.

Finally, the feature subset function evaluation is presented in the Equation 5.7 (HALL, 1999):

$$M_S = \frac{k \cdot \overline{r_{fc}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{5.7}$$

Where:

M_S : Merit of feature subset S.
k : Number of features of subset S.
r_{fc} : Mean feature-class correlation ($f \in S$).
r_{ff} : Average feature-feature inter-correlation ($f \in S$)

5.4.3.4 Heuristic Search

Searching the space of feature subsets within reasonable time constraints is necessary if a feature selection algorithm is to operate on data with a large number of features. One simple search strategy, called greedy hill climbing, considers local changes to the current feature subset. Often, a local change is simply the addition or deletion of a single feature from the subset. When the algorithm considers only additions to the feature subset it is known as **forward selection** while considering only deletions is known as **backward elimination** (MILLER, 2002). An alternative approach, called stepwise bi-directional search, uses both addition and deletion, an example being the **floating search methods** (PUDIL; KITTLER, 1994). Within each of these variations, the search algorithm may consider all possible local changes to the current subset and then select the best; or may simply choose the first change that improves the merit of the current feature subset. In either case, once a change is accepted, it is never reconsidered.

In our exploration, Forward Selection, Backward Elimination, Floating Forward Selection, Floating Backward Selection were studied.

5.4.3.5 Final Feature Subset

In our methodology, the importance of a feature can be different for each gaussian component, which implies that different feature subsets (S_i) will be obtained for each component. We explore three different approaches to evaluate the most suitable method for feature selection:

- Each component has its own subset (probably with different sizes), and this subset is used to compute the posteriori probability.
- For each component we select its best subset of size N and this subset is used to compute the posteriori probability.
- For all components we select the same subset of size N and this subset is used to compute the posteriori probability of all components.

The first approach does not require an additional process, since it has already found the best feature subset for each component. The second approach needs to define the variable N. Finally, the third approach needs to define the variable N and determine the best common subset of all components of the model.

For definition of the variable N, we compute the mean of all sizes of each subset S_i, since we cannot define a specific value as it depends on the dataset.

For definition of the best common subset of the model, we need to chose the features that represent high information in all components. We use the following procedure:

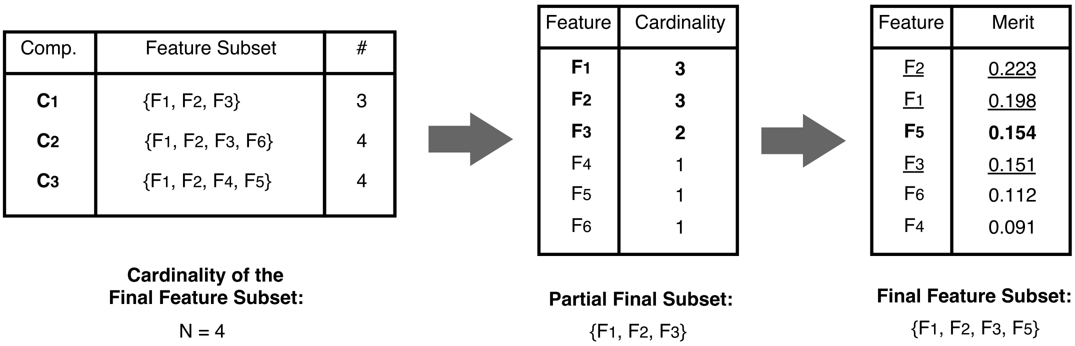
1. Each component generates one feature subset S_j of size $|S_j|$.
2. The value of N is calculated using the average of the values $|S_j|$;
3. The global merit in the model for each feature is calculated as:

$$M_f = \sum_{j=1}^{K} p(j) \cdot M_f^j \tag{5.8}$$

   where M_f^j represents the merit of the feature f in the jth component, using the Equation 5.7.

4. The order of the most common features in all subsets S_j is relative to the number of components in which they are present. Features present in less than two components are not considered.

Figure 5.7: Procedure used to form the best common subset of all components of IGMN.



Comp.	Feature Subset	#
C1	{F1, F2, F3}	3
C2	{F1, F2, F3, F6}	4
C3	{F1, F2, F4, F5}	4

Cardinality of the Final Feature Subset:
N = 4

Feature	Cardinality
F1	**3**
F2	**3**
F3	**2**
F4	1
F5	1
F6	1

Partial Final Subset:
{F1, F2, F3}

Feature	Merit
F2	0.223
F1	0.198
F5	**0.154**
F3	0.151
F6	0.112
F4	0.091

Final Feature Subset:
{F1, F2, F3, F5}

5. If the number of features obtained in the last item is greater than the value N, they are removed features with least merit. On the other hand, if the number of features obtained is less than the value N, they are added features with high merit.

6. Finally, the final feature subset is returned.

Figure 5.7 presents the procedure. In the first stage, the individual feature subsets S_j for each component are obtained, and the size for the final feature subset is calculated. In the second stage, the most common features of all components are selected. Finally in the third stage, the global merit for each feature is calculated and composes the final feature subset.

5.5 IGMN for concept-drifting data stream classification

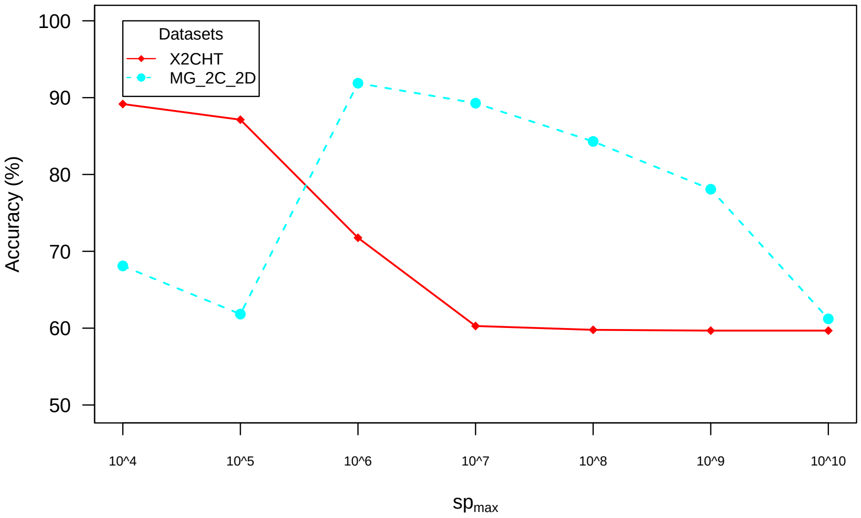
5.5.1 Reset posteriori accumulators

IGMM was designed to process data streams creating online models of eventually huge amounts of data based on accumulation of information in the posteriori accumulator variables (sp_j). This unbounded accumulation may lead to an eventual saturation of the variables that store the corresponding summations, depending on the specific limitation of the available numeric representation. To avoid this problem, the posteriori accumulators are restarted to a fraction γ of their corresponding value, when:

$$\text{if}\left(\sum_{j=1}^{M} sp_j\right) > \beta \cdot sp_{max} \text{ then } sp_j^* = \gamma \cdot sp_j;\ \forall j$$

In Engel (2009), good results were obtained using $sp_{max} = 10^8$, $\beta = 0.8$ and $\gamma = 0.5$. For the author, these parameters are not critical, because the obtained results were practically the same using several different values for sp_{max}, β and γ.

Figure 5.8: Performance of IGMN varying with values of sp_{max}.



In stationary environments, this restarting of posteriori accumulators was satisfactorily validated, but for a non-stationary environment, we employ the hypothesis that by using different values for sp_{max} (only sp_{max} because we consider it the most critical parameter), we will obtain different results. For high values, we maintain information of outdated concepts, and with small values we maintain only partial information of the non-stationary environment. If a component with a huge posteriori probability restarts at a fraction of this value, this component loses a lot of importance in relation to a component with a small posteriori probability that restarts at the same fraction.

To validate this hypothesis, we design one experiment with two synthetic data sets for the concept drift problem: *2CHT* and *MG_2C_2D*. The data sets were obtained from Souza et al. (2015). We use the original IGMN with different values for sp_{max} starting with 10^4 and finishing with 10^{10}. The results presented are expressed in terms of the mean accuracy in the Figure 5.8.

The results show that, in confirmation of our expectations, restarting the posteriori accumulators has a critical impact in non-stationary environments, since for different values of sp_{max}, IGMN produces different results. However, early or late restarting of the variables has a different effect depending on the changes in the environment. In the dataset *2CHT*, an early restarting produces a better accuracy than late restarting, while, for the data set *MG_2C_2D*, a late restarting produces a better accuracy than early restarting.

It was proposed in Section 5.3 that when a gaussian component does not represent up-to-date information of the environment, it is necessary to reduce its importance in the model i.e. reduce its posteriori probability. This action has a direct relationship with the restarting process. However, the restarting process reduces the posteriori probability of all components. An optimal solution would be to restart the posteriori probability of a component only when we detect that this component is not reflecting up-to-date information.

With this in mind, we proposed adding a new parameter to all components, an up-to-date posteriori accumulator (spU_j) with a recent accumulation of posteriori probabilities. If, after a period of time, a variable spU_j does not accumulate enough information, it is a signal that the component j is losing significance in the model, and we need to restart its posteriori probability (to a fraction of its corresponding value).

We can understand "recent information" of a component as its accumulation of posteriori probabilities in the last N training examples, and "enough information" as the minimum quantity, S, that a component needs in its "recent information" to consider the component as up-to-date. The values N and S are related to the condition of "remove spurious component" in IGMN, where a component needs v_{min} examples to show its significance in the model (with a value $sp_j > sp_{min}$). Heinen, Engel and Pinto (2011) explains that a good value for v_{min} is $D * 2$ and a good value for sp_{min} is $D + 1$ (D is the dimension of the example) because according to Traven (1991), a minimum of $D+1$ samples is required to obtain a nonsingular estimate of an unconstrained covariance matrix. However, the value $D * 2$ is important for removing spurious components. The idea is that for $D*2$ examples, these will be assimilated only by the important components of the model, and a component recently created that does not assimilate information is considered spurious. In our case, reset the accumulators and not to remove components, we consider that all components will assimilate the information of these $D * 2$ examples, but for K components, it is difficult to reach the condition of $spU_j > sp_{min}$ (recent information). Finally, we consider that a good value for N would be $K * v_{min}$, since we give to each component enough time to show its significance. Only for the components that do not meet this condition, their posteriori probabilities will be restarted to a fraction γ of its value. After the restarting process, spU_j variables will be restated to 0, and these components have $K \cdot v_{min}$ examples to show their significance in the model (again). If these components do not recover significance, their posteriori accumulators will decrease many times, and in the future these components will be removed.

Moreover, in a very long period of stability, all components will be of significance in the model and their variables sp_j will not reduce. In this context, there is the risk of an eventual saturation of the posteriori accumulators. To avoid this problem, as well as the reduction of the variables sp_j, we maintain the Equation 5.5.1.

This procedure is added to the learning process of IGMN, presented in the Algorithm 8.

Algorithm 8: Update neurons IGMN-NSE

```
Function updateNeurons()
    // Update process of all parameter IGMN
    ...
    // Reset posteriori accumulator
    updatePriors = false;
    forall neuron $j \in M$ do
        if $v_j \geq (v_{min} * K)$ then
            if $spU_j \leq sp_{min}$ then                    // Is not up-to-date
                updatePriors = true;
                $sp_j = sp_j \cdot \gamma$;          // Reset posteriori accumulator
            end
            $spu_j = 0$;
            $v_j = 0$;
        end
    end
    // Update priori probabilities
    if updatePriors = true then
        $p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q}, \forall j$;
    end
end
```

5.5.2 Removing components

With the presence of concept-drift, the data distribution changes too fast, and with the modifications in Sections 5.4.1 and 5.5.1, IGMN has a tendency to maintain a large number of components (or neurons), with the possibility of outdated components appearing. Therefore, it is important to maintain a process of component removal (spurious or outdated), although we need to make sure that this remotion is done only when necessary. The original criterion of IGMN for removal of components only considers remotion of spurious components (with the condition $v_j > v_{min}$ and $sp_j < sp_{min}$). In fact, this condition maintains only components with significance to the model. In accordance with our idea, the only problem that we found was the parameter v_{min}. This parameter represents the short period of time for a component to show its significance in the model. How this was done is explained in Section 5.5.1. $K \cdot v_{min}$ training times (training examples) of a component is enough time for it to show its significance in the model and avoid resetting its posteriori accumulator variable. For the process of removing components, we use the same period of time for a component to show that it should not be removed.

Finally, the algorithm to remove components is present in the Algorithm 9. Although simple, this modification to remove criterion allows us to have a better control of the component elimination process. In it, the recent components have more time to show their importance in the model, and components that lost importance, have probably been removed.

Algorithm 9: Remove neurons IGMN-NSE

```
Function removeNeurons()
    updatePriors = false;
    forall neuron $j \in M$ do
        if $v_j \geq (v_{min} \cdot K)$ and $sp_j \leq sp_{min}$ then
            Delete the $j$th component;
            updatePriors = true;
        end
    end
    // Update priori probabilities
    if updatePriors = true then
        $p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q}, \forall j;$
    end
end
```

6 EXPERIMENTS AND RESULTS

In this chapter we seek to validate the efficiency and effectiveness of our IGMN-NSE algorithm by conducting extensive experiments in two cases: when the scenario is stationary (static datasets) and when it is non-stationary (concept drifting datasets).

Details of experimental methods and setup are presented in Section 6.1. Section 6.2 presents results on static datasets, with the intention of evaluating the performance of IGMN-NSE for classification tasks. In these experiments, we ignore the mechanism for tracking concept drifting of IGMN-NSE and compare our algorithm with the standard IGMN and several baseline classifiers. Experiments with drifting data are presented in Section 6.3. In these experiments, we compare IGMN-NSE to state-of-the-art classifiers for data stream classification, including the standard IGMN.

6.1 Experimental settings

All experiments, for static and drifting datasets, were performed on an INTEL Core i7, 2.50 GHz PC with 12 GB of RAM. We implemented our algorithm in Java, and sources of our implementation and different tests are available from GitHub.

6.1.1 Experimental Datasets

For the experiments with the *static datasets*, nine datasets were chosen from the UCI machine learning library (LICHMAN, 2013), and they were pre-processed to have only numeric and binary values, normalized in the range of [0,1]. The data instances were shuffled randomly, to remove any unintended concept drifts already in the data. The characteristics of the datasets are shown in Table 6.1.

For the experiments with *concept drifting datasets*, we used 8 synthetic and 4 real data sets. The synthetic data sets include abrupt and gradual concept drift and one stationary data stream, while the real data sets have been consistently highlighted in the literature to assess the classification performance of data stream classifiers and exhibit multiclass, temporal dependences and imbalanced data sets. Our goal with this multitude of data sets with different characteristics is to assess how IGMN-NSE performs on each of these scenarios. Table 6.2 presents an overview of the concept drifting datasets, while

Table 6.1: Characteristics of static datasets chosen for our experimentation.

Dataset	Source	# Instances	# Attributes	# Classes
Iris	UCI	150	4	3
Diabetes	UCI	768	8	2
Glass	UCI	214	9	7
Breast Cancer	UCI	286	9	2
Vehicle	UCI	946	18	4
Segment	UCI	2310	19	7
Dermatology	UCI	366	34	6
Ionosphere	UCI	351	34	2
Sonar	UCI	208	60	2

further details may be found bellow.

Table 6.2: Concept drifting dataset configurations [A: Abrupt, G: Gradual].

Dataset	#Intances	# Features	# Classes	Type	Drifts
SEA(G)	100000	3	2	Synthetic	G
SEA(A)	100000	3	2	Synthetic	A
HYPER	100000	10	2	Synthetic	G
RTG	100000	10	2	Synthetic	-
RBF	100000	10	2	Synthetic	G
UG_2C_5D	200000	5	2	Synthetic	G
FG_2C_2D	200000	2	2	Synthetic	G
Gaussian	6000	2	4	Synthetic	G
Electricity	45312	8	2	Real	-
KDD99	49420	42	2	Real	-
Weather	19159	8	2	Real	-
ForestCover	581012	54	7	Real	-

SEA: The database of SEA was first described by Street (2001), which is a well-known data set of concept shift with numerical attributes only. It is composed of a three-dimensional feature space and two classes. All three features have values between 0 and 10, while only the first two features are relevant. Those points are partitioned into four chunks with different concepts. In each chunk, a data point belongs to class 1 if $f_1 + f_2 \leq \theta$, where f_1 and f_2 represent the first two features and θ is a threshold between a pair of concepts. In this database, there are four thresholds of 8, 9, 7 and 9.5 to divide data chunks. We used the data generator of SEA in MOA Framework (BIFET et al., 2010).

It is possible to add noise to class values, being the default value 10%, and to balance the number of instances of each class. We generated two datasets: SEA(G) simulates 3 gradual drifts each with an amplitude of 5k instances, while SEA(A) simulates 3 abrupt drifts (amplitude of 1 instance).

HYPER: This dataset simulates a gradual drift and it was generated based on the hyperplane generator (HULTEN; SPENCER; DOMINGOS, 2001). A hyperplane is a flat, $n - 1$ dimensional subset of that space that divides it into two disconnected parts. It is possible to change a hyperplane orientation and position by slightly changing its relative size of the weights w_i. This generator can be used to simulate time-changing concepts, by varying the values of its weights as the stream progresses. HYPER was generated with MOA Framework, and parametrized with 10 attributes and a magnitude of change of 0.001.

RTG: The Random Tree Generator (RTG) (DOMINGOS; HULTEN, 2000) builds a decision tree by randomly selecting attributes as split nodes and assigning random classes to each leaf. After the tree is build, new instances are obtained through the assignment of uniformly distributed random values to each attribute. The leaf reached after a traverse of the tree, according to the attribute values of an instance, determines its class value. RTG allows customizing the number of nominal and numeric attributes, as well as the number of classes. In our experiments we simulate 10 numeric attributes and 10 classes.

RBF: This dataset was generated using the Radial Basis Function (RBF) generator. This generator creates centroids at random positions and associates them with a standard deviation value, a weight and a class label. To create new instances one centroid is selected at random, where centroids with higher weights have more chances to be selected. The new instance input values are set according to a random direction chosen to offset the centroid. The extent of the displacement is randomly drawn from a Gaussian distribution according to the standard deviation associated with the given centroid. To simulate drifts, centroids move at a continuous rate, effectively causing new instances that ought to belong to one centroid to another with (maybe) a different class. RBF was parametrized with 50 centroids and all of them drift. RBF simulates a gradual drift (speed of change set to 0.0001).

UG_2C_5D: This dataset was presented in Souza et al. (2015) with two classes and five features. This dataset experiences gradual drift (for each 2000 examples), where the two classes make some circulatory movements close to each other, with overlap ranging from small to medium.

FG_2C_2D: Also presented by Souza et al. (2015), this dataset has two classes where the first class is described by three distributions (Gaussian) and the second class is described by one. The four distributions experience drift.

Gaussian: The "Gaussian drift data with class addition/removal" proposed by Elwell and Polikar (2011). In this dataset each class experiences gradual but independent drift, with class means and variances changing according to parametric equations. Class addition and removal are added to the drifting scenario, making this dataset more challenging.

The real datasets used in our experiments are as follows:

Electricity: This dataset was first used in Harries and Wales (1999) and has now become a real-world benchmark dataset for concept drift learning. It was acquired from an electricity supplier in New South Wales (NSW), Australia. It consists of 45,312 instances, 8 attributes and 2 classes. The classification goal is to predict whether the NSW price will be higher than Victoria's price. The dataset contains natural concept drift.

KDD99: This data set was used in KDD Cup 1999 Competition. The full dataset has about five million connection records, but we have used the 10 percent version of the data set, which is more concentrated, hence more challenging than the full version. The original task has 24 training attack types. The original labels of attack types are changed to label abnormal in our experiments and we keep the label normal for normal connection. This way we simplify the set to two class problem (for evaluate concept drift). Each record consists of 42 attributes, such as connection duration, the number bytes transmitted, number of root accesses, etc.

Weather: The U.S. National Oceanic and Atmospheric Administration has compiled weather measurements from over 9000 weather stations worldwide (ELWELL; POLIKAR, 2011). Records date back to the 1930s, providing a wide scope of weather trends. Daily measurements include a variety of features (temperature, pressure, wind speed, etc.) and indicators for precipitation and other weather-related events. As a meaningful real world dataset, we chose the Offutt Air Force Base in Bellevue, Nebraska, for this experiment due to its extensive range of 50 years (1949–1999) and diverse weather patterns, making it a longterm precipitation classification/prediction drift problem. Eight features were selected based on their availability, eliminating those with a missing feature rate above 15%.

ForestCover The forest cover type contains geospatial descriptions of different types of forests obtained from the US Forest Service Region 2 Resource information system (RIS) data. Each class corresponds to a different cover type. This data set contains 581,012 instances, 54 attributes (10 numeric and 44 binary) and 7 imbalanced class labels.

6.1.2 Other Classifiers

For the experiments in the static datasets, in order to perform a more thorough evaluation of our improvements on the standard IGMN, we compare it with four popular classifiers in machine learning: (i) J48, an open source Java implementation of the C4.5 algorithm for building decision trees; (ii) PART (FRANK; WITTEN, 1998), a method that combines the rule learning paradigms C4.5 and RIPPER; (iii) Naive Bayes (NB), a probabilistic classifier; and (iv) Random Forest (RF), an ensemble method that operates by constructing a multitude of decision trees. These algorithms are distributed with the Weka Software (FRANK; HALL; WITTEN, 2016). In addition, since different classifiers require different levels of parameter tuning, we maintain default values provided by the Weka Software.

For the experiments in concept-drifting datasets, we include three groups of algorithms: ensemble learners, online learners with drift detector methods, and adaptive learners. As representatives of ensemble methods, Learn++.NSE and Online Accuracy Updated Ensemble (OAUE) were chosen (algorithms described in Section 4.2). Using as the base learner, the Incremental Naive Bayes Classifier (GONG; LIU; SHI, 2002), the following drift detectors were used: Drift Detection Method (DDM) and EWMA for Concept Drift Detection (ECDD) (algorithms described in Section 4.1.2), as representative of active drift detection methods. Finally, as representative of adaptive learners, we chose the Hoeffding Adaptive Tree (HAT) (described in Section 4.1.3). All these algorithms are available in the MOA Framework (BIFET et al., 2010). We conserve their default value parameters provided for MOA in our experimentation.

6.1.3 Evaluation Criteria

6.1.3.1 Static datasets

For experiments on static datasets, we evaluated the method results in terms of classification performance, using accuracy (ACC) based on the confusion matrix. This matrix quantifies the number of instances in the test set classified as false positive (FP), true positive (TP), false negative (FN) and true negative (TN). The accuracy is defined as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

Additionally, we compute the area under the ROC curve (AUC score). The AUC score is interpreted as the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. Thus, a higher AUC score means a better classification result and a more accurate classifier. However, the AUC score is only applicable for two-class problems (FAWCETT, 2004). Since we have multi-class datasets (Iris, Glass, Vehicle, Segment and Dermatology), we used the approach for calculating multi-class AUCs proposed by Hand and Till (2001). The authors derived a formulation that measures the unweighted pairwise discriminability of classes. Their measure is equivalent to:

$$AUC_{total} = \frac{2}{n\ (n-1)} \sum_{\{c_i, c_j\} \in C} AUC(c_i, c_j) \tag{6.2}$$

where n is the number of classes and $AUC(c_i, c_j)$ is the area under the two-class ROC curve involving classes c_i and c_j. The summation is calculated over all pairs of distinct classes, irrespective of order.

Since these datasets are static, a repeated 10-fold cross-validation (resulting in training sets with 90% of the data, and test sets with the remaining 10%) was performed, averaging results over five repetitions. Statistical significances came from paired t-tests with 5% significance level ($p <$ 0.05).

6.1.3.2 Concept drifting datasets

For experiments with drifting data, method results were evaluated in terms of their execution time and classification performance. Classification performance was assessed using 10-fold distributed cross-validation, proposed by Bifet et al. (2015). This evaluation ought not to be confused with the standard cross-validation from batch learning, which is not applicable to data stream classification mainly because this strategy treats each fold of the stream independently, and therefore may miss concept drift occurring in the data stream. In k-fold distributed cross-validation, each instance of the data stream is used for testing in one randomly selected model and for training by all others.

Since accuracy can be misleading on datasets with class imbalance or temporal dependencies, we compared our proposal model against two baselines: one that always outputs the majority class (Majority Class classifier), and one that outputs the last true class that was acknowledged (Persistent classifier). To capture the performance with respect to the baseline classifiers, instead of the accuracy, we also used the Kappa-M statistic and the Kappa-Temporal statistic.

The Kappa-M statistic proposed by Bifet et al. (2010) is a measure that indicates when we are doing better than a Majority Class classifier. This is defined as:

$$\kappa_m = \frac{p - p_m}{1 - p_m} \tag{6.3}$$

where p is the accuracy of the classifier and p_m is the accuracy of the Majority Class classifier, i.e. the prior probability of the class that has the maximum prior probability.

The Kappa-Temporal statistic proposed by Zliobaite et al. (2015) is a measure that indicates when we are doing better than a Persistent classifier. This is defined as:

$$\kappa_{per} = \frac{p - p_{per}}{1 - p_{per}} \tag{6.4}$$

where p_{per} the accuracy of the Persistent classifier, i.e. the prior weighted probability of observing the same class in two consecutive observations.

The Kappa-M statistic may take values from 1 down to $-\infty$. If the classifier is perfectly correct then $\kappa_m = 1$. If the classifier is achieving the same accuracy as the Majority Class classifier, then $\kappa_m = 0$. Classifiers that outperform the Majority Class classifier fall between 0 and 1. Sometimes it may happen that $\kappa_m < 0$, which means that the reference classifier is performing worse than the Majority Class baseline. The Kappa-Temporal statistic may also take values from 1 down to $-\infty$ and its interpretation is similar to that of κ_m.

Experiments focusing on resource usage were run individually and repeated 10 times to diminish perturbations on the results. We evaluate algorithms using immediate setting and delayed setting. In immediate setting, the true labels are presented to the classifier before the next instance arrives (GOMES et al., 2017). In the delayed setting, the delay was set to 1000 instances and the classification performance estimates are calculated in the same way as they are in the immediate setting, i.e. a k-fold distributed cross-validation. The only difference with immediate setting is "when" true labels become available to train the classifier (GOMES et al., 2017), i.e. 1000 instances after the instance is used for prediction.

To verify if there were statistically significant differences between algorithms, we employ the Friedman test with $\alpha = 0.05$ and the null hypothesis “there were no statistical difference between given algorithms”, if it is rejected, then we proceed with the Nemenyi post-hoc test to identify these differences.

Finally, all experiments with concept drifting datasets were configured and executed within the Massive Online Analysis (MOA) framework (BIFET et al., 2010).

6.2 Experimental results of static datasets

This section presents the experimental evaluation of IGMN-NSE for classification tasks in stationary environments, i.e. static dataset. Firstly, the proposal of the initial component size adaptation was evaluated. Secondly, we make an evaluation of the effects of feature selection in IGMN. Finally, the performance of IGMN-NSE was evaluated comparing it against several popular classifiers.

6.2.1 Initial component size adaptation

In this experiment, our intention is to evaluate the effect of our algorithm in the initial component size adaptation on the standard IGMN algorithm. With the intention to create components with more frequency, we set the parameter τ with the value 0.5. The parameter δ is used with different values for exploring the effects of high values with initial component size adaptation.

Results presented in Table 6.3 show that for high values of δ, the initial component size adaption have a positive impact. This occurs because accuracy increases in comparison to the standard IGMN. The principal reason for this effect is that when δ tends to 1,

Table 6.3: Accuracy of IGMN and IGMN-NSE with different δ values on static datasets. Values represent the mean and standard deviation computed from 10-fold cross-validation.

	$\delta = 0.1$		$\delta = 0.4$		$\delta = 0.7$	
	IGMN	IGMN-NSE	IGMN	IGMN-NSE	IGMN	IGMN-NSE
Iris	94.67 ± 3.33	94.67 ± 3.33	97.33 ± 4.42	97.33 ± 4.42	95.33 ± 6.00	**98.00 ± 3.06**
Diabetes	68.36 ± 3.74	68.36 ± 3.74	**73.97 ± 5.35**	73.32 ± 3.22	73.84 ± 6.44	73.84 ± 6.44
Glass	67.79 ± 8.30	67.79 ± 8.30	66.32 ± 12.05	66.32 ± 12.05	58.46 ± 9.61	**62.23 ± 7.60**
Breast Cancer	61.81 ± 21.03	61.81 ± 21.03	61.51 ± 21.86	61.51 ± 21.86	61.12 ± 21.60	61.12 ± 21.60
Vehicle	71.99 ± 5.70	71.99 ± 5.70	78.24 ± 4.72	78.24 ± 4.72	75.66 ± 4.17	**76.52 ± 3.81**
Segment	94.42 ± 1.77	94.42 ± 1.77	91.00 ± 2.95	**92.14 ± 2.44**	82.47 ± 3.83	**85.45 ± 3.61**
Dermatology	90.20 ± 5.38	90.20 ± 5.38	96.37 ± 3.29	96.37 ± 3.29	96.94 ± 2.31	96.94 ± 2.31
Ionosphere	92.89 ± 3.63	92.89 ± 3.63	85.50 ± 5.50	**87.71 ± 4.91**	93.74 ± 3.06	**94.02 ± 3.12**
Sonar	83.64 ± 8.75	83.64 ± 8.75	87.90 ± 6.02	87.90 ± 6.02	77.76 ± 8.54	77.76 ± 8.54
Average	80.64	80.64	82.02	**82.32**	79.48	**80.65**

Parameters: $\tau = 0.5$ and $\phi = 0.25$

Bold values indicate the best results per data set

the initial size of the components produces overlap (with the old components) with more frequency, resulting in catastrophic forgetting, and IGMN looses accuracy. This additional step, in confirmation of our expectations, reduces the sensitivity of the parameter δ and avoids very small components (with δ tends to 0).

6.2.2 Feature Selection

In this section, our experiments evaluate the effects of using feature selection in IGMN. We explore different heuristic search methods to find the adequate feature subset in accordance with the dataset, namely Forward Selection (FS), Backward Elimination (BE), Floating Forward Selection (FFS), and Floating Backward Selection (FBS). All heuristics were evaluated using the same experimental strategy, adopting the feature subset function described in Equation 5.7. Finally, the same feature subset was used for all components generated in the model after the training of each dataset (as is explained in Section 5.4.3.5).

Table 6.4 depicts the performance of IGMN with the different strategies for feature selection (different heuristic searches). Forward Selection and Floating Forward Selection methods were significantly inferior to the results obtained with the standard IGMN without feature selection. On the other hand, Backward Elimination in three datasets (Segment, Dermatology and Ionosphere), shows better results than the standard IGMN. The results show that, for datasets with a large number of attributes (Segment, Dermatol-

Table 6.4: Accuracy (# of features used) of different algorithms on static datasets. Values represent the mean and the average of the number of features computed from 10-fold cross-validation

	Heuristic Search Method				
Dataset	Without	FS	BE	FFS	FBS
Iris	**98.00 (4)**	95.33 (3.2)	95.33 (3.2)	97.33 (2.6)	97.33 (2.6)
Diabetes	**77.74 (8)**	73.96 (5.9)	73.45 (6.2)	74.49 (6.6)	75.79 (7.1)
Glass	64.00 (9)	**65.87 (6.9)**	63.48 (5.2)	63.55 (7.2)	63.55 (5.3)
Breast Cancer	61.76 (9)	**65.82 (7.9)**	61.44 (8.7)	63.24 (8.1)	61.49 (8.8)
Vehicle	**76.95 (18)**	61.46 (6.2)	75.07 (9.8)	66.56 (7.6)	75.07 (9.8)
Segment	87.45 (19)	39.09 (6.3)	**88.02 (14.1)**	54.29 (9.3)	86.02 (13.2)
Dermatology	95.25 (34)	89.93 (13.4)	**95.53 (31.2)**	84.29 (15.6)	94.29 (29.8)
Ionosphere	94.02 (34)	92.90 (14.1)	**94.22 (29.1)**	76.94 (19.2)	92.66 (30.4)
Sonar	**74.45 (60)**	73.48 (23.4)	**74.45 (60)**	72.50 (29.4)	**74.45 (60)**
Average Acc.	81.07	73.09	80.11	72.58	80.07

Parameters: $\tau = 0.001$ and $\delta = 0.5$

Bold values indicate the best results per data set

ogy and Ionosphere), i.e.: > 12, the features selection in IGMN produces good results.

We can explain that the Backward Elimination Heuristic produces the best results, since this strategy only removes characteristics that it does not consider relevant for the model. This is a significant advantage in our model, since initially, all of the characteristics are considered, then, using our evaluation strategy (based on parameters of IGMN), characteristics that the algorithm considers to be non-relevant are removed, thereby maintaining the maximum possible number of characteristics.

6.2.3 Comparison with other classifiers

The next step in our experiments, is a comparison between IGMN-NSE, the standard IGMN and other classifiers (J48, PART, NB and RF), to evaluate their overall classification performance according to accuracy and AUC score. For consistency, we use the same parameters for the standard IGMN and IGMN-NSE, i.e. $\delta = 0.5$ and $\tau = 0.001$. We group experiments per evaluation metric in Tables 6.5 and 6.6.

Results in terms of accuracy are shown in Table 6.5. In comparison to the standard IGMN, IGMN-NSE shows better or equal results in all datasets with the exception of the

Table 6.5: Mean and standard deviation for the accuracies of different classifiers for a 10-fold cross-validation - Static datasets.

Dataset	IGMN	IGMN-NSE	J48	PART	Naive Bayes	Random Forest
Iris	**98.00 ± 3.06**	**98.00 ± 3.06**	94.00 ± 6.96	94.00 ± 6.96	96.00 ± 4.42	94.67 ± 7.18
Diabetes	77.74 ± 2.69	**77.87 ± 3.89**	74.48 ± 3.46	72.27 ± 3.92	75.66 ± 5.75	76.95 ± 3.01
Glass	64.00 ± 7.30*	62.60 ± 7.32*	63.48 ± 8.04*	66.34 ± 9.22*	48.61 ± 11.28*	**78.90 ± 6.99**
Breast Cancer	61.76 ± 21.75*	61.76 ± 21.75*	74.38 ± 6.85	68.97 ± 7.71*	73.29 ± 3.90	**75.04 ± 6.89**
Vehicle	76.95 ± 3.32	**77.31 ± 3.32**	73.16 ± 3.64	71.53 ± 4.07*	44.20 ± 6.02*	74.71 ± 3.99
Segment	87.45 ± 3.67*	87.62 ± 2.62*	96.75 ± 1.15	96.54 ± 0.82	80.09 ± 3.01*	**98.31 ± 0.68**
Dermatology	95.25 ± 4.13	95.25 ± 4.13	95.25 ± 1.79	94.71 ± 3.15	97.50 ± 3.39	**97.77 ± 2.08**
Ionosphere	94.02 ± 2.67	94.02 ± 2.67	90.03 ± 4.64	91.75 ± 3.69	82.34 ± 8.82*	**94.03 ± 3.67**
Sonar	74.45 ± 10.42*	74.45 ± 10.42*	68.19 ± 9.96*	72.48 ± 9.07*	66.83 ± 11.72*	**85.98 ± 8.04**
Average	81.07	81.10	81.08	80.95	73.84	**86.26**

* statistically significant degradation ($p < 0.05$)

Bold values indicate the best results per data set

Parameters: $\tau = 0.001$ and $\delta = 0.5$

Glass dataset. However, for this specific dataset, IGMN-NSE has a better AUC score than the standard IGMN. These results confirm that our proposed IGMN-NSE has a more suitable performance for classification tasks than the standard IGMN.

Results against the other classifiers show that the improved IGMN algorithm produced competitive results in all datasets, with just four of them (Glass, Breast Cancer, Segment and Sonar) being, with statistically significant difference below the accuracy produced by the Random Forest algorithm. Two of these values were also significantly inferior for all other algorithms (Glass and Sonar).

Results in terms of the AUC score are shown in Table 6.6. These results show that the improved IGMN algorithm was the second best from among all algorithms compared, losing only to the Random Forest. Note, however, that the Random Forest is a batch and ensemble algorithm, while the IGMN learns incrementally from each data point.

Table 6.6: Mean and standard deviation for the AUC scores of different classifiers for a 10-fold cross-validation - Static datasets.

Dataset	IGMN	IGMN-NSE	J48	PART	NB	RF
Iris	**1.000** ± **0.00**	**1.000** ± **0.00**	0.958 ± 0.05	0.958 ± 0.05	0.995 ± 0.01	0.993 ± 0.02
Diabetes	0.812 ± 0.04	0.828 ± 0.03	0.758 ± 0.05*	0.786 ± 0.04*	0.816 ± 0.05	**0.824** ± **0.04**
Glass	0.859 ± 0.05*	0.862 ± 0.04*	0.785 ± 0.07*	0.798 ± 0.11*	0.806 ± 0.08*	**0.947** ± **0.06**
Breast Cancer	0.616 ± 0.11*	0.616 ± 0.11*	0.650 ± 0.09*	0.607 ± 0.12*	**0.729** ± **0.06**	0.723 ± 0.10
Vehicle	0.924 ± 0.01	0.925 ± 0.01	0.861 ± 0.03*	0.856 ± 0.04*	0.770 ± 0.02*	**0.923** ± **0.01**
Segment	0.971 ± 0.01	0.971 ± 0.01	0.985 ± 0.01	0.983 ± 0.01	0.971 ± 0.01	**1.000** ± **0.00**
Dermatology	0.985 ± 0.01	0.985 ± 0.01	0.976 ± 0.01	0.971 ± 0.02	0.997 ± 0.00	**0.999** ± **0.00**
Ionosphere	0.974 ± 0.02	0.974 ± 0.02	0.867 ± 0.09*	0.918 ± 0.05*	0.925 ± 0.07*	**0.981** ± **0.02**
Sonar	0.831 ± 0.11*	0.831 ± 0.11*	0.724 ± 0.12*	0.728 ± 0.07*	0.805 ± 0.11*	**0.948** ± **0.05**
Average	0.886	0.888	0.841	0.845	0.868	**0.927**

* statistically significant degradation ($p < 0.05$)

Bold values indicate the best results per data set

6.3 Experimental results of concept drift datasets

In this section, as the main focus of the dissertation, we evaluate IGMN-NSE for classification tasks in non-stationary environments. Experimentation was initiated by comparing IGMN-NSE against other state-of-the-art classifiers with support to concept drift in terms of accuracy, Kappa M and Kappa T, with immediate setting. In the second part, the experiments were repeated with delayed setting. One important detail is that in these experiments we do not consider feature selection in IGMN-NSE, since only two datasets have a "considerable" number of features (> 12).

6.3.1 Immediate Setting

The first step in our comparison of IGMN-NSE to other algorithms is the evaluation of its overall classification performance according to Accuracy, Kappa M, Kappa Temporal and Processing Time with immediate setting configuration. For consistency, we used the same parameters for IGMN-NSE and IGMN, i.e. $\delta = 0.5$ and $\tau = 0.001$. We group experiments per evaluation metric in Tables 6.7, 6.8, 6.9 and 6.10. Figure 6.2 presents the graphical results for Electricity, Weather and Forest Cover datasets.

Table 6.7: Accuracy mean - Immediate Setting.

Dataset	IGMN	IGMN-NSE	HAT	DDM	ECDD	LearnNSE	OAUE
SEA(G)	83.24	82.60	86.31	**87.26**	84.78	85.28	68.63
SEA(A)	83.25	82.78	86.97	**87.58**	85.13	85.43	87.05
HYPER	81.33	85.09	85.67	**87.89**	86.89	86.21	87.21
RTG	77.93	77.80	94.87	79.82	78.83	79.84	**97.28**
RBF	35.06	38.88	57.95	48.89	36.78	52.96	**67.72**
UG_2C_5D	80.20	**93.32**	92.74	92.26	92.82	90.40	93.29
FG_2C_2D	79.65	91.18	94.97	92.73	93.09	93.65	**95.29**
Gaussian	59.39	**85.80**	77.47	84.48	85.17	81.49	85.50
Synthetic AVG	72.51	79.68	84.62	82.61	80.44	81.91	**85.25**
Electricity	71.60	78.54	82.05	78.26	**83.68**	67.06	77.20
KDD99	98.77	99.80	99.75	**99.87**	99.87	19.93	22.00
Weather	77.02	**77.68**	73.12	71.09	72.85	69.27	74.75
ForestCover	68.70	83.47	80.99	84.94	**87.63**	66.02	84.28
Real AVG	79.19	84.71	83.98	83.54	**86.01**	55.57	64.56
Overall AVG	74.73	81.36	**84.41**	82.92	82.29	73.13	78.35

Bold values indicate the best results per data set

Table 6.8: Kappa M - Immediate Setting.

Dataset	IGMN	IGMN-NSE	HAT	DDM	ECDD	LearnNSE	OAUE
SEA(G)	54.60	53.37	63.88	**66.48**	60.00	61.38	64.64
SEA(A)	54.26	53.64	65.46	**67.00**	60.86	61.73	65.68
HYPER	62.20	69.84	71.00	75.50	73.48	72.09	**74.12**
RTG	48.88	48.58	88.10	53.27	50.73	53.30	**93.67**
RBF	7.11	12.57	39.84	26.88	9.58	32.70	**53.84**
UG_2C_5D	59.45	**86.32**	85.14	84.14	85.30	80.31	86.27
FG_2C_2D	18.62	64.74	79.89	70.91	72.36	74.60	**81.16**
Gaussian	38.31	**78.26**	65.61	76.36	77.34	71.56	77.78
Synthetic AVG	42.93	58.42	69.87	65.07	61.21	63.46	**74.65**
Electricity	32.99	49.60	57.53	48.72	**61.78**	22.33	45.72
KDD99	92.96	98.30	98.22	**98.98**	98.82	-2761.71	-2736.66
Weather	**28.60**	26.18	13.92	7.37	12.81	1.03	19.08
ForestCover	28.28	61.39	57.14	64.37	**70.33**	27.51	64.99
Real AVG	45.71	58.87	56.70	54.86	60.94	-677.71	-651.72
Overall AVG	43.86	58.57	**65.48**	61.67	61.12	-183.60	-167.48

Bold values indicate the best results per data set

Table 6.9: Kappa Temporal - Immediate Setting.

Dataset	IGMN	IGMN-NSE	HAT	DDM	ECDD	LearnNSE	OAUE
SEA(G)	63.87	62.66	70.81	**72.85**	67.60	68.69	71.47
SEA(A)	63.73	62.95	72.15	**73.43**	68.31	68.97	72.32
HYPER	62.54	70.10	71.25	75.71	73.70	72.33	**74.34**
RTG	55.09	54.82	89.53	58.95	56.71	58.98	**94.44**
RBF	16.02	20.96	45.65	33.90	18.24	39.17	**58.29**
UG_2C_5D	60.27	**86.58**	85.43	84.45	85.58	80.75	86.52
FG_2C_2D	45.88	76.55	86.62	80.65	81.62	83.11	**87.47**
Gaussian	33.37	**76.64**	63.02	74.52	75.63	69.46	76.14
Synthetic AVG	50.10	63.91	73.06	69.31	65.92	67.68	**77.62**
Electricity	-95.56	-47.16	-25.36	-50.13	**-11.94**	-125.96	-58.01
KDD99	-1060.35	-143.07	-104.75	**-30.58**	-45.55	-553279.26	-550800.57
Weather	**30.23**	28.13	15.94	9.53	15.00	3.72	21.04
ForestCover	-646.20	-286.82	-337.07	-242.15	**-186.00**	-678.99	-250.18
Real AVG	-442.97	-112.23	-112.81	-78.33	-57.12	-138520.12	-137771.93
Overall AVG	-114.26	5.19	11.10	20.09	**24.91**	-46128.25	-45872.23

Bold values indicate the best results per data set

Table 6.10: Execution Time (seconds) - Immediate Setting.

Dataset	IGMN	IGMN-NSE	HAT	DDM	ECDD	LearnNSE	OAUE
SEA(G)	3.31	3.57	2.47	**0.73**	0.90	40.68	11.19
SEA(A)	3.32	3.48	2.45	**0.70**	0.89	40.49	11.10
HYPER	14.46	6.66	5.85	**1.88**	2.09	117.58	36.29
RTG	13.49	6.60	**5.70**	102.87	101.77	99.11	29.64
RBF	15.12	9.20	8.97	**4.00**	4.25	222.44	72.78
UG_2C_5D	7.80	8.59	6.51	**2.06**	2.41	335.12	35.38
FG_2C_2D	6.82	5.58	5.61	**1.05**	1.48	128.65	19.75
Gaussian	0.48	0.65	0.12	**0.05**	0.06	0.32	0.91
Electricity	31.46	10.02	2.59	0.97	0.93	31.20	16.34
KDD99	38.02	31.52	24.58	13.76	12.82	77.08	110.64
Weather	1.32	1.14	0.59	**0.28**	0.32	3.40	3.24
ForestCover	137.18	114.90	101.53	35.34	**27.06**	338.85	643.10
Average	22.73	16.83	13.91	13.64	**12.92**	119.58	82.53

Bold values indicate the best results per data set

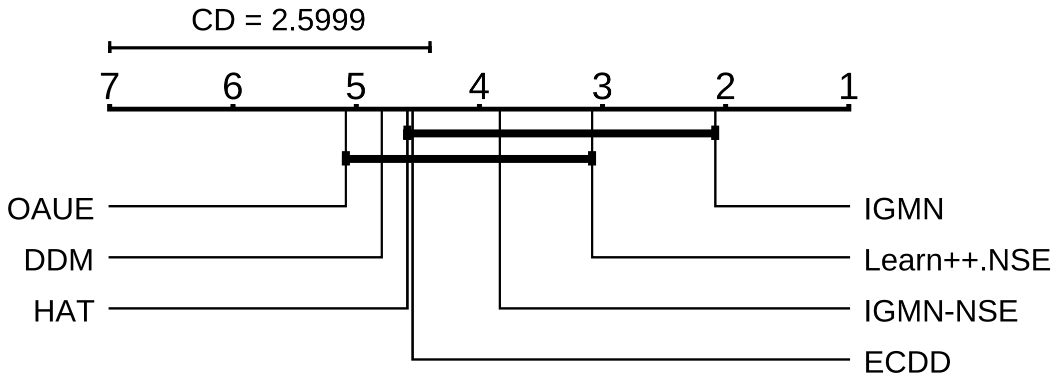
It is obvious that the proposed IGMN-NSE algorithm successfully achieved higher average accuracy than the standard IGMN. This is because the IGMN-NSE algorithm has the capability of gradually and temporally adjusting the neural parameters according to the new incoming data during the learning process, whereas IGMN simply adjusts its neural parameters based on the whole data set. This implies that the local information of how the data is distributed in the space can affect the increase in learning time and accuracy. Therefore, the standard IGMN has better accuracy than IGMN-NSE in three datasets - the two SEA datasets and the RTG dataset. We can explain this result in that the SEA datasets have only four concepts, i.e. only three points of change. The results show that the standard IGMN can be adapted in these scenarios with few concepts in the data stream. In relation to RTG, this is our only synthetic dataset that does not experience concept drift, and in this context, standard IGMN can learn without the necessity of a forgetting mechanism.

In comparison with the other algorithms, we highlight IGMN-NSE performance in two synthetic and one real dataset. Focusing on real world data sets, it is clear that IGMN-NSE consistently obtains competitive results or at least results that could be considered reasonable, in contrast with other algorithms that, even though they achieve very good results, sometimes fail to obtain a reasonable model (e.g. OAUE and Learn++.NSE on KDD99).

Analyzing the results from Kappa Temporal in Table 6.9, we observed that none of the classifiers were able to surpass the baseline (NoChange classifier (ZLIOBAITE et al., 2015)) on three real data sets (Electricity in Figure 6.2b, KDD99 and ForestCover). Probably using a temporally augmented wrapper, as suggested in Zliobaite et al. (2015), would aid this problem for the immediate setting. In our analysis of Kappa M on Table 6.8, we observed that differences in accuracy that did not appear to be very large were actually highlighted in Kappa M. For example, HAT and IGMN-NSE in data set FG_2C_2D achieved 94.97 and 91.18% accuracy, respectively. In terms of Kappa M, HAT achieves 79.89% while IGMN-NSE only 64.74%.

We report statistical tests based on the overall rankings of Table 6.7 (both synthetic and real data sets). The Friedman test indicates that there were significant differences between the classifiers (with $p = 0.006$), the follow-up posthoc Nemenyi test with 95% confidence level, presented in Figure 6.1, indicates that there are significant differences between the standard IGMN and the classifiers DDM and OUAE. IGMN-NSE did not show significant differences with the rest of classifiers, confirming their competitive re-

Figure 6.1: Nemenyi test with 95% confidence level based on the results presented in Table 6.7.

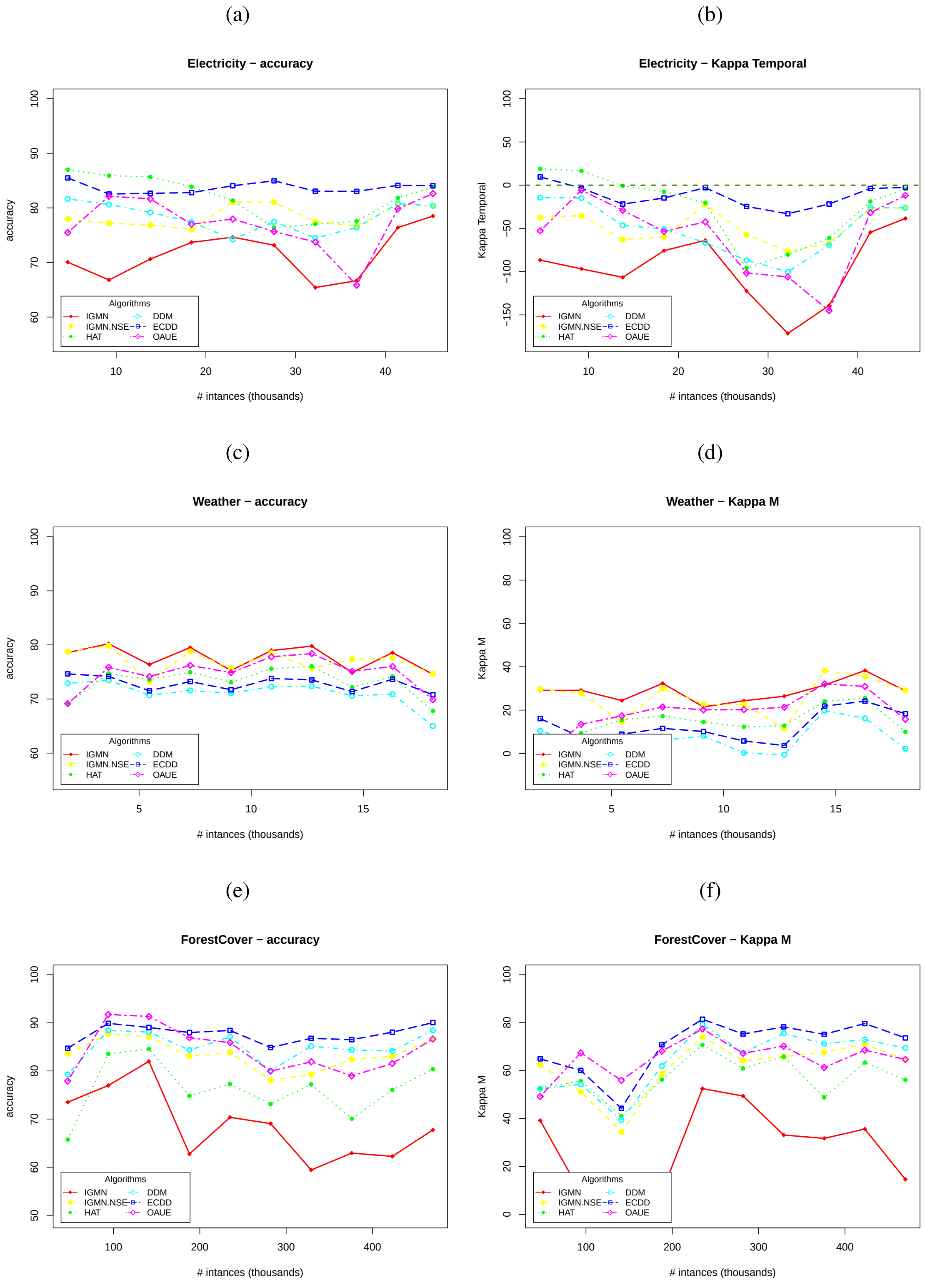


sults.

Moreover, we report average processing time used to process all data sets, presented in Table 6.10. It is expected that IGMN-NSE requires less processing time than standard IGMN, since it can remove outdated components whereas standard IGMN maintains all components that were stable at a point in the data stream, for a longer period of time. In comparison with the other algorithms, it is clear than the ensemble methods require more processing time, since each ensemble member has its own learning process. On the other hand, adaptive methods and drift detector methods have similar results to IGMN-NSE.

In Figure 6.2 some results of the experiments (Tables 6.7, 6.8 and 6.9) are presented. IGMN-NSE is able to consistently achieve superior accuracy on Electricity (Fig. 6.2a) and ForestCover (Fig. 6.2e), in comparison to the standard IGMN. In Weather (Fig. 6.2c), IGMN-NSE obtain similar results that the standard IGMN, but this was the method with highest accuracy in comparison to the other classifiers. In Figure 6.2b, observing the Kappa Temporal plot it is clear that classifiers are not actually improving relatively to the NoChange classifier. Weather and ForestCover plots in Fig. 6.2d and Fig. 6.2f show that by using Kappa M on imbalanced datasets, the methods are actually improving relatively to the the majority class classifier.

Figure 6.2: Sorted plots of Accuracy, Kappa M and Kappa T over time.



(a)

(b)

(c)

(d)

(e)

(f)

Table 6.11: Accuracy mean - Delayed Setting.

Dataset	IGMN	IGMN-NSE	HAT	DDM	ECDD	LearnNSE	OAUE
SEA(G)	82.69	82.02	85.37	**86.41**	84.16	84.72	85.73
SEA(A)	82.69	82.08	86.01	**86.62**	84.31	84.64	86.11
HYPER	80.70	84.34	84.63	**86.72**	85.70	85.16	86.12
RTG	77.70	77.59	94.44	81.08	78.41	79.53	**96.79**
RBF	33.49	48.50	54.53	46.46	35.31	52.15	**64.56**
UG_2C_5D	79.52	**92.97**	92.27	91.68	92.49	90.09	92.86
FG_2C_2D	79.35	91.03	94.79	92.47	92.99	93.54	**95.11**
Gaussian	47.23	**51.63**	49.05	50.44	51.08	47.63	49.83
Synthetic AVG	70.42	76.27	80.14	77.74	75.56	77.18	**82.14**
Electricity	57.32	65.64	68.72	58.48	42.77	47.57	**74.75**
KDD99	94.12	**98.00**	95.57	95.03	95.04	19.52	22.20
Weather	73.94	**76.60**	70.90	68.83	68.69	67.17	72.42
ForestCover	67.74	71.91	76.28	61.93	52.30	71.23	**79.78**
Real AVG	73.28	**78.04**	77.87	71.07	64.70	51.37	62.29
Overall AVG	71.37	76.86	**79.38**	75.51	71.94	68.58	75.52

Bold values indicate the best results per data set

6.3.2 Delayed Setting

The variations in results of delayed and immediate settings, presented in Tables 6.7 and 6.11 respectively, suggest that IGMN-NSE is more suitable to the immediate setting. However, IGMN-NSE has better performance in the real dataset, in terms of accuracy, compared to with the other classifiers, mainly because ensemble methods fail in the KDD99 dataset. In comparison to the adaptive and drift detection methods, IGMN-NSE consistently obtains better results.

The statistical tests for the delayed settings indicates that there are statistically significant differences between each classifier in the results produced with immediate setting, the OAUE learner being the exception. This suggests that sometimes drift detection methods and adaptive classifiers are more sensitive to the delayed setting such as DDM and ECDD, where their overall performance is clearly degraded when training is delayed. This is especially true for gradual drifts, as "drift signals" are delayed and action is taken to accommodate a potentially already outdated concept. This is observed by analyzing the accuracy drop from the immediate to delayed setting for all algorithms in Gaussian dataset (Tables 6.7 and 6.11).

7 CONCLUSIONS AND FUTURE WORK

In this dissertation, we study the use of the Incremental Gaussian Mixture Network (IGMN) for data stream classification in scenarios where concept-drift is present. This study resulted in the implementation of a new model, called Incremental Incremental Gaussian Mixture Network for Non-Stationary Environment (IGMN-NSE). This is the main goal of the current study with its principal contribution being an adaptive classifier that learns incrementally and adapts to changing environments using a forgetting mechanism to remove outdated components.

Beyond the characteristics inherited from IGMN, IGMN-NSE also presents improvements for classification tasks, with the implementation of a novelty criterion to create new components and an initial component size adaptation for the component created. We also introduce the use of an embedded method for feature selection in IGMN. With this method our algorithm identifies the most representative features for a dataset.

In relation to non-stationary environments, we recognized the weaknesses of IGMN in this scenario, and we implemented a mechanism for recognizing when components lost significance in the model. This process was transformed into a forgetting mechanism for removal of the outdated components of the model.

Nevertheless, IGMN-NSE had problems when the experiments were realized with delayed setting. Our model lost performance, in terms of accuracy, when the true labels arrived with delay. Another problem was found when the datasets presented a lot of nominal features, since IGMN-NSE considers each feature as numeric, nominal features represent a challenge in the learning process. Our practical solution was to convert these nominal features to binary features, but this change increments the number of features (one nominal feature with 3 different values is converted to three binary features), and IGMN-NSE lost performance in comparison with other algorithms.

Finally, besides the related issues, the IGMN-NSE may be considered a good classifier for concept-drifting data streams, based on the experiments and the results presented. The model performed well on static and drifting dataset (real and synthetics), surpassing, in some cases, other similar classifiers.

7.1 Future work

In order to verify the benefits offered by IGMN-NSE, new experiments with more complex environments would be necessary, e.g. text stream classification, feature evolution, etc. Unfortunately, the time window left for the conclusion of this research is not large enough, and this will be left for future work. Future studies may involve:

- *IGMN as a feature selector.* We proposed a simple embedded method for feature selection using IGMN, which gave better results in preliminary experiments when this subset was used on IGMN. In a future work, we propose to evaluate whether the feature subset selected for IGMN represents a subset with high quality and if these subsets can be used in other classifiers.
- *IGMN-NSE as a member of an ensemble.* In this dissertation, we presented IGMN-NSE as a single adaptive learner, i.e. it uses online learning and implements a forgetting mechanism. However, this classifier can be used as a member of an ensemble method and works together with other classifiers or constructs an ensemble method with only IGMN-NSE classifiers with different configurations.
- *Time series analysis.* In IGMN-NSE, the main characteristic for recognizing changes in the data, is the removal of outdated components. With the use of time series analysis, we can identify how and why a component lost significance in the model, since each component maintains representative data information. For example, in recommendation systems, when certain suggestions are no longer recommended, this information could be represented by a component, and from this we could identify which characteristics of that data produced that particular loss of significance in the model.

REFERENCES

AGGARWAL, C. C. Data classification: Algorithms and applications. **A survey of stream classification algorithms**, CRC Press, p. 245–268, 2014. Available from Internet: <https://pdfs.semanticscholar.org/1e94/fdfd783c553ed04aacd03e17bfc6aa69fded.pdf>.

AITNOURI, E. et al. Controlling mixture component overlap for clustering algorithms evaluation. **Pattern Recognition and Image Analysis**, v. 12, p. 331–346, 2002. Available from Internet: <https://elibrary.ru/item.asp?id=15318978>.

BAENA-GARĆıA, M. et al. Early drift detection method. **In Fourth International Workshop on Knowledge Discovery from Data Streams**, p. 77–86, 2006. Available from Internet: <http://www.lsi.upc.edu/~abifet/EDDM.pdf>.

BARDDAL, J. P. et al. Sncstream+. **Inf. Syst.**, Elsevier Science Ltd., v. 62, n. C, p. 60–73, 2016. Available from Internet: <https://doi.org/10.1016/j.is.2016.06.007>.

BHATTACHARYYA, A. On a measure of divergence between two statistical populations defined by their probability distributions. **Bulletin of the Calcutta Mathematical Society**, v. 35, n. 1, p. 99–109, 1943. Available from Internet: <http://www.citeulike.org/user/giampierosalvi/article/7675294>.

BIFET, A. Adaptive learning and mining for data streams and frequent patterns. **SIGKDD Explorations Newsletter**, ACM, v. 11, p. 55–56, 2009. Available from Internet: <http://doi.acm.org/10.1145/1656274.1656287>.

BIFET, A.; GAVALDà, R. Adaptive learning from evolving data streams. **International Symposium on Intelligent Data Analysis**, Springer-Verlag, p. 249–260, 2009. Available from Internet: <http://dx.doi.org/10.1007/978-3-642-03915-7_22>.

BIFET, A.; GAVALDà, R. Learning from time-changing data with adaptive windowing. **International Conference on Data Mining**, p. 443–448, 2007. Available from Internet: <http://epubs.siam.org/doi/abs/10.1137/1.9781611972771.42>.

BIFET, A. et al. Moa: Massive online analysis. **The Journal of Machine Learning Research**, JMLR.org, v. 11, p. 1601–1604, 2010. Available from Internet: <http://dl.acm.org/citation.cfm?id=1756006.1859903>.

BIFET, A. et al. Efficient online evaluation of big data stream classifiers. **Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, ACM, p. 59–68, 2015. Available from Internet: <http://doi.acm.org/10.1145/2783258.2783372>.

BLACK, M.; HICKEY, R. Learning classification rules for telecom customer call data under concept drift. **Soft Computing**, v. 8, n. 2, p. 102–108, 2003. Available from Internet: <https://doi.org/10.1007/s00500-002-0250-2>.

BLUM, A. L.; LANGLEY, P. Selection of relevant features and examples in machine learning. **Artificial Intelligence - Special issue on relevance**, Elsevier Science Publishers Ltd., v. 97, n. 1-2, p. 245–271, 1997. Available from Internet: <http://dx.doi.org/10.1016/S0004-3702(97)00063-5>.

BREIMAN, L. Pasting small votes for classification in large databases and on-line. **Machine Learning**, v. 36, n. 1, p. 85–103, 1999. Available from Internet: <https://doi.org/10.1023/A:1007563306331>.

BRZEZINSKI, D. Mining data streams with concept drift. **Thesis for MSc**, 2010. Available from Internet: <https://doi.org/10.13140/RG.2.1.4634.6086>.

BRZEZIŃSKI, D.; STEFANOWSKI, J. Accuracy updated ensemble for data streams with concept drift. **Hybrid Artificial Intelligent Systems: 6th International Conference**, Springer Berlin Heidelberg, p. 155–163, 2011. Available from Internet: <https://doi.org/10.1007/978-3-642-21222-2_19>.

BRZEZINSKI, D.; STEFANOWSKI, J. Combining block-based and online methods in learning ensembles from concept drifting data streams. **Information Sciences**, v. 265, n. Supplement C, p. 50–67, 2014. Available from Internet: <https://doi.org/10.1016/j.ins.2013.12.011>.

CAUWENBERGHS, G.; POGGIO, T. Incremental and decremental support vector machine learning. **Proceedings of the 13th International Conference on Neural Information Processing Systems**, MIT Press, p. 388–394, 2000. Available from Internet: <http://dl.acm.org/citation.cfm?id=3008751.3008808>.

COBOS, C. et al. Clustering of web search results based on the cuckoo search algorithm and balanced bayesian information criterion. **Inf. Sci.**, Elsevier Science Inc., v. 281, p. 248–264, 2014. Available from Internet: <http://dx.doi.org/10.1016/j.ins.2014.05.047>.

COMANICIU, D.; RAMESH, V.; MEER, P. Kernel-based object tracking. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE Explore Digital Library, v. 25, n. 5, p. 564–577, 2003. Available from Internet: <https://doi.org/10.1109/TPAMI.2003.1195991>.

DEMPSTER, A.; LAIRD, N.; RUBIN, D. Maximum likelihood from incomplete data via the em algorithm. **Journal of the Royal Statistical Society Series B (Methodological)**, v. 39, n. 1, p. 1–38, March 1977. Available from Internet: <http://www.jstor.org/stable/2984875>.

DITZLER, G.; POLIKAR, R. An ensemble based incremental learning framework for concept drift and class imbalance. **International Joint Conference on Neural Network**, 2010. Available from Internet: <https://doi.org/10.1109/IJCNN.2010.5596764>.

DOMINGOS, P.; HULTEN, G. Mining high-speed data streams. In: **International Conference on Knowledge Discovery and Data Mining**. ACM, 2000. p. 71–80. Available from Internet: <http://doi.acm.org/10.1145/347090.347107>.

DOMINGOS, P.; HULTEN, G. A general framework for mining massive data streams. **Journal of Computational and Graphical Statistics**, American Statistical Association, v. 12, p. 945–949, 12 2003.

DRIES, A.; RüCKERT, U. Adaptive concept drift detection. **Statistical Analysis and Data Mining**, John Wiley & Sons, Inc., v. 2, n. 5-6, p. 311–327, 2009. Available from Internet: <http://dx.doi.org/10.1002/sam.v2:5/6>.

ELWELL, R.; POLIKAR, R. Incremental learning in nonstationary environments with controlled forgetting. **International Joint Conference on Neural Networks**, p. 771–778, 2009. Available from Internet: <https://doi.org/10.1109/IJCNN.2009.5178779>.

ELWELL, R.; POLIKAR, R. Incremental learning of variable rate concept drift. **Multiple Classifier Systems**, Springer Berlin Heidelberg, p. 142–151, 2009. Available from Internet: <https://doi.org/10.1007/978-3-642-02326-2_15>.

ELWELL, R.; POLIKAR, R. Incremental learning of concept drift in nonstationary environments. **IEEE Transactions on Neural Networks**, IEEE Explore Digital Library, v. 22, n. 10, p. 1517–1531, 2011. Available from Internet: <https://doi.org/10.1109/TNN.2011.2160459>.

ENGEL, P. INBC: an incremental algorithm for dataflow segmentation based on a probabilistic approach. **Universidade Federal do Rio Grande do Sul**, 2009. Available from Internet: <http://hdl.handle.net/10183/126650>.

ENGEL, P.; HEINEN, M. Incremental learning of multivariate gaussian mixture models. **Advances in Artificial Intelligence–SBIA**, p. 82–91, 01 2011. Available from Internet: <https://www.researchgate.net/publication/215385274>.

FAWCETT, T. **ROC Graphs: Notes and Practical Considerations for Researchers**. [S.l.], 2004. Available from Internet: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.9777>.

FERRER-TROYANO, F.; AGUILAR-RUIZ, J. S.; RIQUELME, J. C. Data streams classification by incremental rule learning with parameterized generalization. **Proceedings of the 2006 ACM Symposium on Applied Computing**, ACM, p. 657–661, 2006. Available from Internet: <http://doi.acm.org/10.1145/1141277.1141428>.

FRANK, E.; HALL, M. A.; WITTEN, I. H. **Data Mining: Practical Machine Learning Tools and Techniques**. 4. ed. [S.l.]: Morgan Kaufmann, 2016.

FRANK, E.; WITTEN, I. H. Generating accurate rule sets without global optimization. **Proceedings of the Fifteenth International Conference on Machine Learning**, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 144–151, 1998. Available from Internet: <http://dl.acm.org/citation.cfm?id=645527.657305>.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of Computer and System Sciences - Special issue**, Academic Press, Inc., v. 55, n. 1, p. 119–139, 1997. Available from Internet: <http://dx.doi.org/10.1006/jcss.1997.1504>.

FUKUNAGA, K. **Introduction to statistical pattern recognition**. 2nd. ed. Academic Press Professional, Inc., 1990. Available from Internet: <https://dl.acm.org/citation.cfm?id=92131>.

GAMA, J.; CASTILLO, G. Learning with local drift detection. **Advanced Data Mining and Applications**, Springer Berlin Heidelberg, p. 42–55, 2006. Available from Internet: <https://doi.org/10.1007/11811305_4>.

GAMA, J. et al. Learning with drift detection. **Brazilian Symposium on Artificial Intelligence**, Springer Berlin Heidelberg, p. 286–295, 2004. Available from Internet: <https://doi.org/10.1007/978-3-540-28645-5_29>.

GAMA, J. a. **Knowledge Discovery from Data Streams**. 1st. ed. Chapman & Hall/CRC, 2010. ISBN 1439826110, 9781439826119. Available from Internet: <https://dl.acm.org/citation.cfm?id=1855075>.

GAMA, J. a.; GABER, M. M. **Learning from Data Streams: Processing Techniques in Sensor Networks**. 1. ed. [S.l.: s.n.], 2007.

GAMA, J. a. et al. A survey on concept drift adaptation. **ACM Comput. Surv.**, ACM, v. 46, n. 4, p. 44:1–44:37, 2014. Available from Internet: <http://doi.acm.org/10.1145/2523813>.

GANTI, V.; GEHRKE, J.; RAMAKRISHNAN, R. Mining data streams under block evolution. **SIGKDD Explor. Newsl.**, ACM, v. 3, n. 2, p. 1–10, 2002. Available from Internet: <http://doi.acm.org/10.1145/507515.507517>.

GOLAB, L.; OZSU, M. T. Issues in data stream management. **ACM SIGMOD Record**, ACM, New York, NY, USA, v. 32, n. 2, p. 5–14, jun. 2003. ISSN 0163-5808. Available from Internet: <http://doi.acm.org/10.1145/776985.776986>.

GOMES, H. M. et al. A survey on ensemble learning for data stream classification. **ACM Computing Survey**, ACM, v. 50, n. 2, p. 23:1–23:36, 2017. Available from Internet: <https://doi.org/10.1145/3054925>.

GONG, X.-J.; LIU, S.-H.; SHI, Z.-Z. An incremental bayes classifical model. **Chinese Journal of Computers**, 2002. Available from Internet: <http://en.cnki.com.cn/Article_en/CJFDTOTAL-JSJX200206013.htm>.

GROSSBERG, S. Competitive learning: From interactive activation to adaptive resonance. **Cognitive Science**, Wiley Online Library, v. 11, n. 1, p. 23–63, 1987. ISSN 0364-0213. Available from Internet: <https://doi.org/10.1016/S0364-0213(87)80025-3>.

GU, X. F. et al. An improving online accuracy updated ensemble method in learning from evolving data streams. **International Computer Conference on Wavelet Actiev Media Technology and Information Processing**, p. 430–433, 2014. Available from Internet: <https://doi.org/10.1109/ICCWAMTIP.2014.7073443>.

GUPTA, A. et al. Network monitoring as a streaming analytics problem. **Proceedings of the 15th ACM Workshop on Hot Topics in Networks**, ACM, p. 106–112, 2016. Available from Internet: <http://doi.acm.org/10.1145/3005745.3005748>.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. **The Journal of Machine Learning Research**, JMLR.org, v. 3, p. 1157–1182, 2003. Available from Internet: <http://dl.acm.org/citation.cfm?id=944919.944968>.

HALL, M. A. Correlation-based feature selection for machine learning. **Ph.D. Thesis, Department of Computer Science - The University of Waikato**, 1999. Available from Internet: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.149.3848>.

HAND, D. J.; TILL, R. J. A simple generalisation of the area under the roc curve for multiple class classification problems. **Machine Learning**, v. 45, n. 2, p. 171–186, 2001. Available from Internet: <https://doi.org/10.1023/A:1010920819831>.

HARRIES, M.; WALES, N. S. Splice-2 comparative evaluation: Electricity pricing. **Technical Report - University of New South Wales**, 1999. Available from Internet: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.9013>.

HARVILLE, D. A. **Matrix Algebra From a Statistician's Perspective**. 1st. ed. Springer, 1997. Available from Internet: <http://www.springer.com/la/book/9780387949789>.

HEINEN, M. A connectionist approach for incremental function approximation and on-line tasks. **Ph.D. Thesis, Informatics Institute - Universidade Federal do Rio Grande do Sul**, May 2011. Available from Internet: <http://hdl.handle.net/10183/29015>.

HEINEN, M.; ENGEL, P. Igmn: An incremental connectionist approach for concept formation, reinforcement learning and robotics. **Journal of Applied Computing Research**, v. 1, p. 2–19, 2011. Available from Internet: <http://revistas.unisinos.br/index.php/jacr/article/view/jacr.2011.11.01/0>.

HEINEN, M.; ENGEL, P.; PINTO, R. IGMN: An incremental gaussian mixture network that learns instantaneously from data flows. **VIII Encontro Nacional de Inteligencia Artificial**, p. 1–12, 01 2011. Available from Internet: <https://www.researchgate.net/publication/215455099>.

HOENS, T. R.; POLIKAR, R.; CHAWLA, N. V. Learning from streaming data with concept drift and imbalance: an overview. **Progress in Artificial Intelligence**, v. 1, n. 1, p. 89–101, 2012. Available from Internet: <https://doi.org/10.1007/s13748-011-0008-0>.

HULTEN, G.; SPENCER, L.; DOMINGOS, P. Mining time-changing data streams. **Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, ACM, p. 97–106, 2001. Available from Internet: <http://doi.acm.org/10.1145/502512.502529>.

JACOBS, R. A. et al. Adaptive mixtures of local experts. **Neural Computing**, MIT Press, v. 3, n. 1, p. 79–87, 1991. Available from Internet: <http://dx.doi.org/10.1162/neco.1991.3.1.79>.

KELLY, M. G.; HAND, D. J.; ADAMS, N. M. The impact of changing populations on classifier performance. **International Conference on Knowledge Discovery and Data Mining**, ACM, p. 367–371, 1999. Available from Internet: <http://doi.acm.org/10.1145/312129.312285>.

KHAMASSI, I.; SAYED-MOUCHAWEH, M. Drift detection and monitoring in non-stationary environments. **IEEE Conference on Evolving and Adaptive Intelligent Systems**, p. 1–6, 2014. Available from Internet: <https://doi.org/10.1109/EAIS.2014.6867461>.

KHAMASSI, I. et al. Self-adaptive windowing approach for handling complex concept drift. **Cognitive Computation**, v. 7, n. 6, p. 772–790, 2015. Available from Internet: <https://doi.org/10.1007/s12559-015-9341-0>.

KIFER, D.; BEN-DAVID, S.; GEHRKE, J. Detecting change in data streams. **International Conference on Very Large Data Bases**, VLDB Endowment, p. 180–191, 2004. Available from Internet: <http://dl.acm.org/citation.cfm?id=1316689.1316707>.

KOLMOGOROV–SMIRNOV Test. In: THE Concise Encyclopedia of Statistics. Springer New York, 2008. p. 283–287. Available from Internet: <https://doi.org/10.1007/978-0-387-32833-1_214>.

KOLTER, J. Z.; MALOOF, M. A. Dynamic weighted majority: An ensemble method for drifting concepts. **The Journal of Machine Learning Research**, JMLR.org, v. 8, p. 2755–2790, 2007. Available from Internet: <http://dl.acm.org/citation.cfm?id=1314498.1390333>.

KUBAT, M. Floating approximation in time-varying knowledge bases. **Pattern Recognition Letters**, v. 10, n. 4, p. 223–227, 1989. Available from Internet: <http://www.sciencedirect.com/science/article/pii/0167865589900925>.

KUBAT, M. A machine learning-based approach to load balancing in computer networks. **Cybernetics and Systems - Special issue**, Taylor & Francis, Inc., v. 23, n. 3-4, p. 389–400, 1992. Available from Internet: <http://dx.doi.org/10.1080/01969729208927471>.

KUNCHEVA, L. Classifier ensembles for changing environments. **Multiple Classifier Systems: 5th International Workshop**, Springer Berlin Heidelberg, v. 1, n. 1, p. 1–15, 2004. Available from Internet: <https://doi.org/10.1007/978-3-540-25966-4_1>.

KUNCHEVA, L. I.; ZLIOBAITE, I. On the window size for classification in changing environments. **Intelligent Data Analysis**, IOS Press, v. 13, n. 6, p. 861–872, 2009. Available from Internet: <http://dl.acm.org/citation.cfm?id=1662648.1662650>.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. Available from Internet: <http://archive.ics.uci.edu/ml>.

LITTLESTONE, N. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. **Machine Learning**, v. 2, n. 4, p. 285–318, 1988. Available from Internet: <https://doi.org/10.1023/A:1022869011914>.

LITTLESTONE, N.; WARMUTH, M. The weighted majority algorithm. **Information and Computation**, v. 108, n. 2, p. 212–261, 1994. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0890540184710091>.

LU YIU-MING CHEUNG, Y. Y. T. Y. Dynamic weighted majority for incremental learning of imbalanced data streams with concept drift. **International Joint Conference on Artificial Intelligence**, p. 2393–2399, 2017. Available from Internet: <https://doi.org/10.24963/ijcai.2017/333>.

MEJRI, D.; KHANCHEL, R.; LIMAM, M. An ensemble method for concept drift in nonstationary environment. **Journal of Statistical Computation and Simulation**, Taylor & Francis, v. 83, n. 6, p. 1115–1128, 2013. Available from Internet: <https://doi.org/10.1080/00949655.2011.651797>.

MILLER, A. J. **Subset Selection in Regression**. 2nd. ed. Chapman and Hall/CRC, 2002. Available from Internet: <https://www.crcpress.com/Subset-Selection-in-Regression/Miller/p/book/9781584881711>.

NARASIMHAMURTHY, A.; KUNCHEVA, L. I. A framework for generating data to simulate changing environments. **International Multi-Conference: Artificial Intelligence and Applications**, ACTA Press, p. 384–389, 2007. Available from Internet: <http://dl.acm.org/citation.cfm?id=1295303.1295369>.

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. **Knowledge and Information Systems**, Springer-Verlag New York, Inc., v. 45, n. 3, p. 535–569, 2015. Available from Internet: <https://doi.org/10.1007/s10115-014-0808-1>.

OLIVEIRA, L. S.; BATISTA, G. E. A. P. A. IGMM-CD: A gaussian mixture classification algorithm for data streams with concept drifts. **Brazilian Conference on Intelligent Systems (BRACIS)**, IEEE Explore Digital Library, p. 55–61, 2015. Available from Internet: <https://doi.org/10.1109/BRACIS.2015.61>.

OZA, N. C. **Online Ensemble Learning**. Thesis (PhD), 2001.

PAGE, E. Continuous inspection schemes. **Biometrika**, JSTOR, p. 100–115, 1954. Available from Internet: <https://doi.org/10.2307/2333009>.

PEREIRA, R. d. P. Higmn : an igmn-based hierarchical architecture and its applications for robotic tasks. **Universidade Federal do Rio Grande do Sul**, May 2013. Available from Internet: <http://www.lume.ufrgs.br/handle/10183/80752>.

PINTO, R.; ENGEL, P. A fast incremental gaussian mixture model. **PLOS ONE**, Public Library of Science, v. 10, p. 1–12, 2015. Available from Internet: <https://doi.org/10.1371/journal.pone.0139931>.

POLIKAR, R. et al. Learn++: a classifier independent incremental learning algorithm for supervised neural networks. **International Joint Conference on Neural Networks (IJCNN)**, IEEE Explore Digital Library, n. 52, p. 1742–1747, 2002. Available from Internet: <https://doi.org/10.1109/IJCNN.2002.1007781>.

POWELL, P. D. Calculating determinants of block matrices. **Rings and Algebras (math.RA)**, 2011. Available from Internet: <https://arxiv.org/abs/1112.4379>.

PUDIL, P.; KITTLER, J. Floating search methods in feature selection. **Pattern Recogn. Lett.**, Elsevier Science Inc., v. 15, n. 11, p. 1119–1125, 1994. Available from Internet: <https://doi.org/10.1016/0167-8655(94)90127-9>.

ROBBINS, H.; MONRO, S. A stochastic approximation method. **The Annals of Mathematical Statistics**, JSTOR, v. 22, n. 3, p. 400–407, 1951. Available from Internet: <www.jstor.org/stable/2236626>.

ROBERTS, S. W. Control chart tests based on geometric moving averages. **Technometrics**, JSTOR, v. 42, n. 1, p. 97–101, 1959. Available from Internet: <https://doi.org/10.2307/1271439>.

ROSS, G. J. et al. Exponentially weighted moving average charts for detecting concept drift. **Pattern Recognition Letters**, v. 33, n. 2, p. 191–198, 2012. Available from Internet: <https://doi.org/10.1016/j.patrec.2011.08.019>.

SAEYS, Y.; INZA, I. n.; nAGA, P. L. A review of feature selection techniques in bioinformatics. **Bioinformatics**, Oxford University Press, v. 23, n. 19, p. 2507–2517, 2007. Available from Internet: <http://dx.doi.org/10.1093/bioinformatics/btm344>.

SHERMAN, J.; MORRISON, W. J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. **The Annals of Mathematical Statistics**, v. 21, n. 1, p. 124–127, 1950. Available from Internet: <http://www.jstor.org/stable/2236561>.

SNEDECOR, G. W.; COCHRAN, W. G. **Statistical Methods**. 8th. ed. Iowa State University Press, 1989. Available from Internet: <https://archive.org/details/in.ernet.dli.2015.5515>.

SOUZA, V. M. A. et al. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. p. 873–881, 2015. Available from Internet: <http://epubs.siam.org/doi/abs/10.1137/1.9781611974010.98>.

STREET, W. N. A streaming ensemble algorithm (SEA) for large-scale classification. **Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, ACM, p. 377–382, 2001. Available from Internet: <https://doi.org/10.1145/502512.502568>.

SUN, H.; WANG, S. Measuring the component overlapping in the gaussian mixture model. **Data Min. Knowl. Discov.**, Kluwer Academic Publishers, v. 23, n. 3, p. 479–502, 2011. Available from Internet: <http://dx.doi.org/10.1007/s10618-011-0212-3>.

SUN, L. Y.; CAI, W.; HUANG, X. X. Data aggregation scheme using neural networks in wireless sensor networks. **International Conference on Future Computer and Communication**, IEEE Explore Digital Library, v. 1, p. 725–729, 2010. Available from Internet: <https://doi.org/10.1109/ICFCC.2010.5497335>.

TRAVEN, H. G. C. A neural network approach to statistical pattern classification by 'semiparametric' estimation of probability density functions. **IEEE Transactions on Neural Networks**, IEEE Explore Digital Library, v. 2, n. 3, p. 366–377, 1991. Available from Internet: <https://doi.org/10.1109/72.97913>.

WANG, H. et al. Mining concept-drifting data streams using ensemble classifiers. **Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, ACM, p. 226–235, 2003. Available from Internet: <http://doi.acm.org/10.1145/956750.956778>.

WEISSTEIN, E. W. Statistical correlation. **MathWorld - A Wolfram Web Resource**, 2017. Available from Internet: <http://mathworld.wolfram.com/StatisticalCorrelation.html>.

WIDMER, G.; KUBAT, M. Effective learning in dynamic environments by explicit context tracking. **European Conference on Machine Learning**, Springer-Verlag, p. 227–243, 1993. Available from Internet: <http://dl.acm.org/citation.cfm?id=645323.649587>.

WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. **Machine Learning**, v. 23, n. 1, p. 69–101, 1996. Available from Internet: <https://doi.org/10.1007/BF00116900>.

ZANG, W. et al. Comparative study between incremental and ensemble learning on data streams: Case study. **Journal Of Big Data**, Springer, v. 1, n. 1, p. 5, Jun 2014. Available from Internet: <https://doi.org/10.1186/2196-1115-1-5>.

ZLIOBAITE, I. Learning under concept drift: an overview. **Faculty of Mathematics and Informatics of Vilnius University, Lithuania, Tech**, 2010. Available from Internet: <http://arxiv.org/abs/1010.4784>.

ZLIOBAITE, I. et al. Evaluation methods and decision theory for classification of streaming data with temporal dependence. **Journal Machine Learning**, Kluwer Academic Publishers, v. 98, n. 3, p. 455–482, 2015. Available from Internet: <http://dx.doi.org/10.1007/s10994-014-5441-4>.

ZLIOBAITE, I.; KUNCHEVA, L. I. Determining the training window for small sample size classification with concept drift. **International Conference on Data Mining Workshops**, p. 447–452, 2009. Available from Internet: <https://doi.org/10.1109/ICDMW.2009.20>.

APPENDIX A — RESUMO EM PORTUGUÊS

A.1 Contexto

A pesquisa tradicional de mineração de dados está focada principalmente em ambientes estáveis, onde uma base de dados completa é apresentada ao algoritmo de aprendizagem e os conceitos alvos que devem ser aprendidos são fixos. No entanto, em algumas das aplicações mais recentes, os algoritmos de aprendizagem funcionam em ambientes dinâmicos, onde os dados são gerados continuamente. Algumas dessas novas aplicações incluem fluxos de consultas de previsão (COBOS et al., 2014), monitoramento de redes (GUPTA et al., 2016), redes de sensores (SUN; CAI; HUANG, 2010) e fluxos de redes sociais (BARDDAL et al., 2016). Nesses ambientes dinâmicos, os dados recebidos formam um fluxo contínuo de dados caracterizado por enormes quantidades de instâncias que chegam com alta velocidade, e que muitas vezes requerem uma resposta rápida e em tempo real. Comparado aos ambientes estáveis, o processamento de fluxos contínuos de dados implica novos requisitos para algoritmos, como restrições sobre o uso de memória, aprendizagem restrita e tempo de teste, e processamento das instâncias recebidas (BRZEZIŃSKI; STEFANOWSKI, 2011). Além disso, os problemas encontrados em um ambiente estável também estão presentes em um fluxo contínuo de dados, por exemplo, valores faltantes, superposição, ruído, atributos irrelevantes, classes desbalanceadas e outros. Além disso, devido à natureza não estacionária dos fluxos contínuos de dados, um problema adicional é que as características dos dados podem mudar ao longo do tempo, uma condição conhecida como *mudança de conceito* (*concept drift*). A mudança de conceito ocorre quando o conceito, sobre quais dados estão sendo coletados, se altera após algum período de estabilidade (GAMA, 2010). Os fluxos contínuos de dados que exibem mudanças de conceito são referidos como fluxos de dados em evolução ou não estacionários (GOMES et al., 2017).

Nesta dissertação, nos concentramos na classificação de fluxos contínuos de dados em ambientes não estacionários. A classificação de fluxos contínuos de dados é uma variação da tarefa de classificação supervisionada tradicional. Ambas as tarefas buscam prever um valor nominal de uma instância não rotulada representada por um vetor de características. Um breve resumo desses estudos anteriores relacionados à classificação de fluxos contínuos de dados segue.

A primeira abordagem é o aprendizado incremental (*incremental learning*). O aprendizado incremental é um tipo de aprendizagem onde o classificador atualiza seu modelo quando uma nova experiência significativa fica disponível. O principal desafio de aprender em vários ambientes é como preservar todos os conhecimentos adquiridos, para que o classificador consiga decidir qual conhecimento deve ser substituído ou retirado para melhorar seu desempenho (DRIES; RüCKERT, 2009). Isso é conhecido como o "dilema estabilidade-plasticidade", onde "estabilidade" significa manter conhecimento existente e "plasticidade" descreve a capacidade de aprender novos conhecimentos (ELWELL; POLIKAR, 2011).

Uma abordagem alternativa para o aprendizado incremental é o uso de algoritmos baseados em conjunto de classificadores, onde cada classificador é denominado *expert*. Os conjuntos de classificadores são populares na classificação de fluxos contínuos de dados, pois, além de alavancar *experts* fracos, eles podem ser usados para lidar com desafios que surgem de cenários de fluxos de dados, como a mudança de conceito (KOLTER; MALOOF, 2007; ELWELL; POLIKAR, 2011; BRZEZINSKI; STEFANOWSKI, 2014).

Ambas as abordagens podem lidar com fluxos contínuos de dados e mudanças de conceito, e cada uma delas possui seus pontos fortes. Em geral, os algoritmos de aprendizado incremental têm melhor eficiência, e os algoritmos baseados em conjuntos de classificadores adaptam-se melhor a mudanças de conceito (ZANG et al., 2014).

A.2 Motivação

Uma solução ideal para a classificação de fluxos contínuos de dados não estacionários seria combinar a modularidade de um conjunto de classificadores na presença de mudança de conceito, com a simplicidade de uso encontrada em um único classificador com aprendizado incremental. Com esta motivação em mente, propomos o uso da IGMN (Incremental Gaussian Mixture Network) (HEINEN; ENGEL; PINTO, 2011), que é uma rede neural baseada no aprendizado incremental de modelos probabilísticos. A IGMN atende a maioria dos requisitos que nos interessam, ou seja, (i) a topologia da rede neural é definida de forma automática e incremental, (ii) não requer acesso a dados anteriormente utilizados, (iii) mantém uma maioria substancial do conhecimento anteriormente adquirido ao aprender novas informações, (iv) o processo de aprendizagem pode prosseguir perpetuamente, (v) pode lidar com o dilema estabilidade-plasticidade e, finalmente, (vi) pode ser usada em tarefas de aprendizagem supervisionadas e não supervisionadas.

A.3 Visão Geral da Pesquisa

O IGMN possui uma natureza incremental e pode induzir e adaptar incrementalmente um modelo ao longo do tempo, de acordo com os dados recebidos. No entanto, não é ideal para lidar com mudanças de conceito, uma vez que não fornece explicitamente um mecanismo para esquecer conceitos desatualizados. Os conceitos desatualizados podem potencialmente sobrepor novos conceitos no espaço de atributos, o que, por sua vez, aumenta a incerteza dos dados. Se a distribuição de dados mudar, as informações dos dados antigos precisam ser adaptadas para se adequarem aos novos dados (ZLIOBAITE, 2010). Com isso em mente, as seguintes questões da IGMN serão abordadas:

1. IGMN não se adapta rápidamente à mudança de conceito.
2. A IGMN não implementa um mecanismo de "esquecimento" para reconhecer quando um componente perde significância no modelo.

Com base nesses fatos e em alguns resultados preliminares, o objetivo principal desta dissertação é apresentar uma nova abordagem para a classificação do fluxo de dados com base em uma adaptação do algoritmo IGMN, denominada IGMN-NSE (*Incremental Gaussian Mixture Network for Non-Stationary Environments*). A IGMN-NSE propõe uma melhoria de poder preditiva para tarefas de classificação e uma política para descartar conceitos desatualizados em contraste com o modelo IGMN padrão. Apresentamos e testamos este modelo em alguns conjuntos de dados sintéticos e reais conhecidos, comparando os resultados com os obtidos com modelos clássicos e a versão padrão do IGMN.

A.4 Conclusões

Nesta dissertação, estudamos o uso da IGMN para a classificação de fluxo de dados em cenários onde mudanças de conceito estão presentes. Este estudo resultou na implementação de um novo modelo, chamado *Incremental Incremental Gaussian Mixture Network for Non-Stationary Environment* (IGMN-NSE). Este foi o principal objetivo do estudo atual, sendo a principal contribuição um classificador adaptativo que aprende incrementalmente e se adapta a ambientes em mudança usando uma estratégia de "esquecimento" para remover componentes desatualizados.

Além das características herdadas da IGMN, a IGMN-NSE também apresenta melhorias para tarefas de classificação, com a implementação de um critério de detecção de novidades para criar novos componentes, e uma estratégia de adaptação do tamanho inicial de um componente criado. Também apresentamos o uso de um método incorporado para seleção de atributos na IGMN. Com este método, nosso algoritmo identifica os atributos mais representativos para uma base de dados.

Reconhecemos os pontos fracos da IGMN em ambientes não estacionários e implementamos um mecanismo para reconhecer quando os componentes perderam significância no modelo. Esse processo foi transformado em um mecanismo de esquecimento para a remoção dos componentes desatualizados do modelo.

No entanto, o IGMN-NSE apresentou problemas quando os experimentos foram realizados com uma disponibilidade tardia dos rótulos corretos. Neste cenário nosso modelo perdeu desempenho em termos de acurácia. Outro problema foi encontrado quando as bases de dados apresentaram muitos atributos nominais, uma vez que o IGMN-NSE considera cada atributo como numérico, os atributos nominais representam um desafio no processo de aprendizagem. Nossa solução prática foi converter esses atributos nominais em atributos binários, mas essa alteração aumenta o número de atributos (um atributo nominal com 3 valores diferentes é convertido em três atributos binários) e a IGMN-NSE perde desempenho quando comparado com outros algoritmos que não precisam desse ajuste.

Finalmente, apesar dos problemas apresentados, os experimentos e resultados demonstram que o algoritmo desenvolvido pode ser considerado um bom classificador para fluxos contínuos de dados que apresentam mudanças de conceito, com base nos experimentos e nos resultados apresentados. O modelo funcionou bem nas bases de dados tanto estáveis quanto com as que apresentaram mudanças de conceito (reais e sintéticas), superando, em alguns casos, outros classificadores semelhantes.

A.5 Trabalhos futuros

Para verificar benefícios oferecidos pela IGMN-NSE, novos experimentos com ambientes mais complexos seriam necessários, por exemplo: classificação de fluxos de texto, evolução de características, etc. Infelizmente, a janela de tempo disponibilizada para a conclusão desta pesquisa não foi suficientemente grande, e isso será deixado para trabalhos futuros. Estudos futuros podem envolver:

- *IGMN como seletor de atributos.* Propomos um método embutido simples para seleção de atributos usando a IGMN, o que deu melhores resultados em experimentos preliminares quando este subconjunto foi usado no IGMN. Em um trabalho futuro, propomos avaliar se o subconjunto selecionado para IGMN represente um subconjunto com alta qualidade, e se esses subconjuntos podem ser usados em outros classificadores.

- *IGMN-NSE como membro de um conjunto de classificadores.* Nesta dissertação, apresentamos a IGMN-NSE como um único classificador adaptável, ou seja, ele usa a aprendizagem online e implementa um mecanismo de esquecimento. No entanto, este classificador pode ser usado como um membro de um método baseado em um conjunto de classificadores (*ensemble*), e trabalhar em conjunto com outros classificadores, ou ter um conjunto de classificadores com apenas classificadores IGMN-NSE com diferentes configurações.

- *Análise de séries temporais.* Na IGMN-NSE, a principal característica para reconhecer mudanças nos dados, é a remoção de componentes desatualizados. Com o uso da análise de séries temporais, podemos identificar como e por que um componente perdeu significância no modelo, uma vez que cada componente mantém informações representativas dos dados que representa. Por exemplo, em sistemas de recomendação, quando certas sugestões não são mais recomendadas, essa informação pode ser representada por um componente e, a partir disso, podemos identificar quais características desses dados produziram essa perda de significância específica no modelo.