Federal University of Minas Gerais

School of Engineering

Graduate Program in Electrical Engineering

Doctoral Thesis

Dominating Sets of the Gabriel Graph: an Approach for One-class and Online Learning Classifiers

Wagner José de Alvarenga Júnior

Belo Horizonte

2022

Wagner José de Alvarenga Júnior

Dominating Sets of the Gabriel Graph: an Approach for One-class and Online Learning Classifiers

Doctoral thesis submitted to the Graduate Program in Electrical Engineering at the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering

Supervisor: Prof. Dr. Antônio de Pádua Braga

Belo Horizonte

2022

A473d

Alvarenga Júnior, Wagner José de.
Dominating sets of the Gabriel Graph: na approach for one-class and online learning classifiers [recurso eletrônico] / Wagner José de Alvarenga Júnior. - 2022.
1 recurso online (97 f. : il., color.) : pdf.

Orientador: Antônio de Pádua Braga.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 90-97.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Aprendizado do computador – Teses. 3. Teoria dos grafos – Teses. I. Braga, Antônio de Pádua. II. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pela Bibliotecária Letícia Alves Vieira - CRB-6/2337
Biblioteca Prof. Mário Werneck - Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"DOMINATING SETS OF THE GABRIEL GRAPH: AN APPROACH FOR ONE-CLASS AND ONLINE LEARNING CLASSIFIERS"

WAGNER JOSÉ DE ALVARENGA JÚNIOR

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 14 de julho de 2022. Por:

Prof. Dr. Antônio de Pádua Braga
DELT (UFMG) - Orientador

Prof. Dr. Walmir Matos Caminhas
DELT (UFMG)

Prof. Dr. Janier Arias García
DELT (UFMG)

Prof. Dr. Aluízio Fausto Ribeiro Araújo
Centro de Informática (UFPE)

Prof. Dr. Raul Fonseca Neto
Departamento de Ciência da Computação (UFJF)

to Lívia, my father and mother, Daniela and Marcela,

and Antônio e Teresa.

Acknowledgments

First of all, I thank God for allowing that this work could be done.

I am grateful to my supervisor, Prof. Antônio Braga, who received me as a PhD student and have been advising in theoretical issues and encouraging with his own example.

I express my gratitude to my colleges from the LITC Lab., Prof. Alex Assis, Alexsander Costa, Prof. Cristiano Castro, Eduardo Gonçalves, Prof. Eduardo Ribeiro, Felipe Campos, Gabriel Lara, Gustavo Maia, Prof. Frederico Coelho, Lourenço Araujo, Prof. Luiz Torres, Murilo Menezes, Turibio Salis, Vítor Hanriot and Prof. Vítor Torres, for all the fruitful discussions that have given me more understanding on many subjects.

I will be in debt with Lívia, for a while. Her support and motivation have been present during all this work, despite my constant absence.

I thank the IHM company for providing the data used in the tests of this thesis, and the fellows Eduardo Magalhães and Pablo Drumond for the technical discussions.

Last but not least, I thank the PPGEE and its faculty for providing me means to develop my scientific knowledge, and the financial support provided by CAPES.

Abstract

Online Learning of non-stationary data streams is characterized by changes in the data generating function, which may impact the predictive performance of a model. Therefore, classifiers capable of adapting to such situations constitute a viable solution. Generally, such models rely on hyperparameters that need to be previously configured. A different task that also presents issues, concerning setting of hyperparameters, is the learning with one-class classifiers, in which the information from only one class is used to establish a decision boundary. The main proposal of this thesis is to use the structural information from a data set to define classifiers, in the two Learning Paradigms previously discussed. This goal is achieved by exploring the fact that an Independent Dominating Set, when induced from the Gabriel graph, tends by definition to result in a subset of dominating points, with representative characteristic of the original set. Thus, an Independent Dominating Set algorithm that neither requires setting hyperparameters nor the use of any optimization method to find a solution is proposed, as well as an online updating procedure for the Gabriel graph. These two methods are used to define the hypeparameters of models based on Radial functions: a KDE estimator for the online scenario and an RBF network as a one-class classifier. This graph dominance approach results in an appropriate and distributed number of Radial functions, in the input domain, and a stable radius that cover the training points and leads to a classifier with appropriate Capacity. An algorithm based on the independent dominating set of the Gabriel graph is also proposed to extract representative subsets from large data sets. This thesis also presents an online training method for a regularized SLFN network that continually maintains the learning process. The method uses an adaptive window to mitigate the impact of concept drifts. These methods were tested with synthetic data sets and with data from a real industrial process.

Keywords: Online learning, Gabriel graph, Dominating set, KDE, RBF, One-class classifier, SLFN.

Resumo

O Aprendizado Online de dados não estacionários é caracterizado por mudanças na função geradora dos dados, com possível impacto sobre o desempenho de um modelo preditor. Por isto, classificadores que apresentam Capacidade apropriada ao longo das predições são uma possível solução. Porém, tais modelos geralmente possuem hiperparâmetros que necessitam ser definidos previamente. Uma tarefa diferente desta, que também possui desafios relacionados a determinação da Capacidade de um modelo, é o aprendizado realizado com classificadores de classe única. Neste modelo, a superfície de decisão é induzida a partir dos dados referentes a uma única classe. A proposta principal desta tese está centrada no uso da informação da estrutura dos dados para se definir classificadores nos dois Paradigmas de Aprendizados anteriores. Para isto, utiliza-se de um Conjunto Dominante Independente do grafo de Gabriel, o qual tende por definição a ser um subconjunto de pontos dominantes distribuídos e com característica de representatividade do conjunto original. Desta forma, é proposto um algoritmo para a obtenção de um conjunto dominante independente, o qual não necessita de configurar parâmetros e também não utiliza métodos de otimização para achar a solução. É também proposto, uma abordagem para se atualizar, de forma online, o grafo de Gabriel com um novo ponto. Estes dois métodos são usados na definição hiperparamétrica de modelos com funções Radiais: um estimador KDE empregado no cenário de aprendizado online e uma rede RBF utilizada como classificado de classe única. Além destes, é proposto um algoritmo baseado no conjunto dominante independente e no grafo de Gabriel, cuja finalidade é extrair subconjuntos representativos de um conjunto original que possua muito pontos. Esta tese apresenta ainda um método de treinamento online para uma rede SLFN. O processo contínuo de treinamento utiliza uma janela adaptativa para atenuar o impacto causado por mudanças de conceito. Esses métodos foram testados com conjuntos de dados sintéticos e com dados de um processo industrial real.

Palavras-chave: Aprendizado online, Gabriel graph, Conjunto dominante, KDE, RBF, SLFN, Classificador de classe única.

List of Figures

List of Tables

List of abbreviations

AUC	Area Under the ROC Curve
IDS	Independent Dominating Set
KDE	Kernel Density Estimation
LOO-CV	Leave-one-out Cross Validation
OCSVM	One-class Support Vector Machine
RBF	Radial Basis Function
SLFN	Single Hidden Layer Feedforward Neural Network
SLFN-RP	Single Hidden Layer Feedforward Neural Network based on Random Projections
SRM	Structural Risk Minimization
SVDD	Support Vector Domain Description

Contents

Chapter 1

Introduction

Since a considerable amount of data sets are built in temporal sequences that often require online processing, an important consideration is to aim for algorithms whose parameters can be updated on a timely basis. [Gama et al., 2014]. Online Learning comprises a range of methods that aim at the prediction or decision-making tasks, using the information contained in data points that become sequentially available, when dealing with a specific problem [Hoi et al., 2021]. This approach has in its essence the construction of a model, performed simultaneously with your own use as a predictor. Thus, the model is continuously updated as a new point is presented.

Online Learning presents challenges that are inherent to its nature and how the data flow to be handled is presented. Nor the entire data that represents a particular problem is available *a priori*, neither is possible to consider it with finite size, in some cases. Another point is that the online learning task generally needs to process non-stationary data streams, which are characterized by changes of the data generating function [Gama et al., 2014; Takahashi and Braga, 2020; Widmer and Kubat, 1996]. Known as Concept Drift, this issue may impact the prediction performance of a classifier, since the new data distribution may not be promptly present in the representation knowledge of the model, immediately after the change [Kelly et al., 1999; Gama et al., 2014]. Thus, it is important to detect concept drifts and to adapt the learning model to the new scenario, or to use a robust classifier, that although performing an ordinary updating steps, is capable to adjust the model to the new scenario [Ditzler et al., 2015]. The great difficulty in carrying out the change detection is the diversity of ways and subtleties in which the concept drift can take place [Elwell and Polikar, 2011; Takahashi and Braga, 2020], and

the fact that drift detection methods require the setting of parameter. Failure to identify a drift or a false drift detection can negatively impact the prediction performance of the sequentially learning model, while the model adaptation is limited to the possible adjustments made between the time interval of presentation of two sequence points, as well as being based on the little information available when starting a new scenario.

Classification problems usually present enough data to estimate a classifier that is capable to separate reasonably the data points of each class. Although, in some cases the data set that describes a particular problem may be under-represented, with missing information from one of the classes [He and Garcia, 2009]. This condition can be the result of a sampling problem, a class that is eventually ill-defined or formed by rare points. An approach to address this problem is through the task of Novelty Detection, which concerns in the identification of unexpected events or patterns in a data set of interest [Modenesi and Braga, 2009; Pimentel et al., 2014]. A possible solution in this case, is the use of an one-class classifier induced from points belonging to only one of the existent classes [Japkowicz et al., 1995; Parra et al., 1996; Schölkopf et al., 2001; Tax and Duin, 2004], with the objective to establish a decision boundary able to separate these points from outliers. Thus, with such an approach the classification of a new sample is accomplished with a decision function that assigns it or not, to the class learned.

A fundamental problem in machine learning is the determination of an appropriate Capacity for a model [Vapnik, 1995], which dictates the level of complexity of the approximation function that is induced. In the online learning, the data generating function may change, at any time, whenever points are presented to the model. Thus, the adaptation of complexity for the approximation function maybe not be trivial for models that require the setting of hyperparameters, since the search for them during the online prediction may not be feasible, due to the required time to perform it. A point worth mentioning is, the larger the number of hyperparameter, the more difficult is the task of finding optimal values [Claesen and De Moor, 2015]. The definition of an one-class model with appropriate capacity, also presents its particularities. The inductive process using only points from one of the classes leaves some open questions, like how far from the extreme training points should the decision boundary be established, whether there should be training points outside of this last (how many?), and how smooth should be the decision boundary that encapsulate the training points used to induce the function. Usually, these issues are controlled by hyperparameters of an one-class classifier

Some methods make use of the information from the data structure to define the capacity of a classifier. One example is the use of the Gabriel graph [Gabriel and Sokal, 1969] to identify key points from a data set, that are used as support points in the solution of large margin classifies [Zhang and King, 2002]. This approach was applied for the definition of different models [Torres et al., 2014, 2015b,a, 2020], which do not require the configuration of hyperparameters. A variation of this approach, proposes the use of a Dominating Set [Haynes et al., 1998], which is induced from the Gabriel graph, in the definition of parameters of Radial functions from a RBF network [Queiroz et al., 2022]. These two approaches that use information from the data structure, were applied in the construction of binary classifies, as offline methods. The Gabriel graph was proposed as a method for geographic variation analysis, in which an adjacency criteria is established between two points, A and B, from a data set, if, and only if, the square of the Euclidean distance between A and B is less or equal the sum of the squares of the distances from A to C and from B to C, being C a different point. Such criteria results in a graph that preserves the local relation of the nodes.

A general definition of graph dominance states that: a subset of nodes is called dominating if every node in the graph is either in this subset, or is adjacent to a node in the dominating subset [Haynes et al., 1998]. However, several different types of dominating sets exist, with different characteristics of dominance, for example: *hop dominating set*, defined as a subset in which every node outside of it is at 2-step dominated (distance 2 apart) by some node in the subset [Natarajan and Ayyaswamy, 2015; Henning et al., 2020], *total dominating set*, defined for a graph with no isolated node [Cockayne et al., 1980; Allan et al., 1984; Henning, 2009], *connected dominating set*, in which the dominating set induces a connected graph [Sun et al., 2017; Wu et al., 2006], *independent dominating set*, in which no two nodes in the subset are adjacent and every node outside of it is adjacent to at least one node in the subset [Cockayne and Hedetniemi, 1976; Allan and Laskar, 1978; Goddard and Henning, 2013], *k-dominating set*, in which every node in the non-dominating set is adjacent to at least k nodes in the dominating set [Chellali et al., 2012], to name a few.

Generative models for classification problems rely on validity of the induction principle assumed and how representative is the sampled points used to induction [Duda et al., 2000]. Density estimation with parametric models assumes a fixed structure for the data and expect that the global model parameters be induced from it. When few representative points are available, a global assumption may results in a biased estimator [Vapnik, 1995]. However, non-parametric

estimators do not assume a predefined structure, providing a more realistic assumption. One example of non-parametric model is the Kernel Density Estimation (KDE) Estimator [Silverman, 1986], which induces a functional mapping through a superposition of kernel functions, centered in each point of a data set. Some of the first works with this approach appeared in the 1960s [Rosenblatt, 1956; Parzen, 1962]. In this type of model the kernel parameter (radius) has a central role in the bias and variance dilemma [Geman et al., 1992], defining the smoothness of the solution.

Radial Basis Functions (RBF) neural networks [Broomhead and Lowe, 1988] are function approximation models, that are induced by fit curve [Haykin, 1994]. In this conception, learning is equivalent to searching for a surface, in a high dimension space, that best fits the training data, according to some criterion. The RBF networks present as characteristic the use of Radial functions in the neurons of the hidden layer, resulting in a monotonically decreasing behavior, in relation to the central point of each function. This particularity produces a localized response at the neuron's output to a given input stimulus. The responses of the neurons from the hidden layer are linearly combined on the output of the network. In general, two approaches are used for training RBF networks. One estimates the parameters of hidden and output layers separately, while the the other estimates both together. In the first case, the parameters of the hidden layer are kept fixed after being estimated by some clustering method, while the parameters of the output layer are estimated by some linear method. In the second case, some nonlinear method needs to be used to obtain the joint solution of the parameters. The former solution presents well-known boundary conditions, while the latter may face issues related to a nonlinear method adopted and the intrinsic sensitivity of Radial functions. Even so, both approaches required the setting of hyperparameters.

Single hidden Layer Feedforward neural Networks (SLFNs) [Haykin, 1994] are universal approximation models, which are able to approximate any continuous function [Cybenko, 1989]. The usual methods applied in batch mode supervised learning of SLFNs are the backpropagation algorithm [Rumelhart et al., 1986] and its variations. In these gradient descent-based approaches all network parameters are estimated in a generally slow process that may face convergence issues, although efforts to speedup iterations in order to mitigate this problem have been made. Another point related to these training methods is the hyperparameter tuning, which is necessary for a good performance. A different training approach is the use of random projections in the hidden layer of the network [Schmidt et al., 1992], and the use of an analyt-

ical solution to calculate output weights. This procedure is based on the theory that non linear data becomes more probable to be linearly separable when projected in a high dimension space [Cover, 1965]. This approach became popular as Extreme Learning Machines (ELM) [Huang et al., 2004, 2006b].

1.1 Problem Definition

The problems of the Online Learning and the Learning when done with an one-class classifier, in offline mode, are briefly defined.

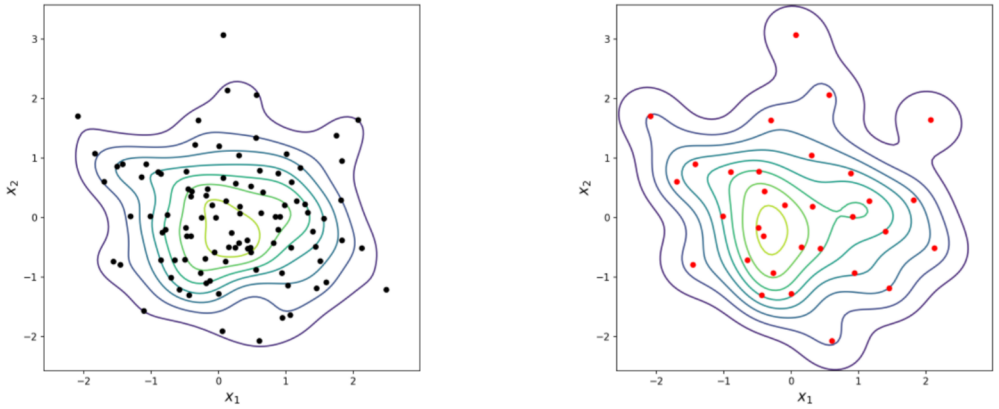
The problem of the Online Learning can be formulated with a data generating function, consisting of a sequence of points and respective labels, arranged in tuples and describing the conditions of a particular problem, for each instant of time. In this Paradigm, the objective is the induction of a approximation function, using some cost function. This task is carried out aiming at the classification of a new point, whenever this last becomes available, followed by the assess to its true label and the updating of the model. The problem of the Learning with an one-class classifier, in offline mode, can be stated for a set of points in the real domain, in which the objective is to induce a function capable to define a decision boundary, for these points, and to classifier new ones as belonging, or not, to the designated class.

1.2 Objectives

As described previously, the hyperparameters of online learning models and one-class classifiers determine the complexity of the approximation function, thus having a directly influence over the predictive performance (bias and variance dilemma [Geman et al., 1992]). The main objective of this thesis is to propose solutions for the Online Learning and for the Learning in offline mode, with an one-class classifier. A major concern of this research is to propose methods capable to have their parameters/hyperparameters defined from the information of the data structure, circumventing the obstacles to define them directly, during the Online Learning of non-stationary data streams or when classifying in offline mode, with one-class classifier, data sets containing information of only one class.

1.3 Contributions

The predictive models proposed in this thesis use Radial functions to map the data in a new domain where the classification is done. It is proposes a Graph Dominance approach, with dominating sets of the Gabriel graph, aiming to define the hyperparameters of the classifiers. This approach is illustrated in Figure 1.1, in which the graph on the right shows the density estimation with kernels centered only on the dominating points of a set, that has its density estimation plotted on the left graph. The non-dominating points are used to define the radius of the Radial functions.



(a) KDE of a set of points V.

(b) KDE of a dominating set $S \subset V$.

Figure 1.1: Graph Dominance approach in the covering of a set of points.

The contributions of this thesis are listed in the following:

- **Online updating of Gabriel graph:** The proposed approach for online update of Gabriel graph initially induces an graph from a set of points, saving information of midpoints and distances between these points in 3 matrices. An adjacency matrix is defined comparing the distance between any two points (adjacent or not) to the distance between their midpoint and the other points. A sliding window is used to save a sequence of the most recent data, that is represented with the nodes of the Gabriel graph. When a new point becomes available, the 3 matrices are updated, with computation of midpoints and distances, only related to the new point and the older point that is removed from the sliding window. This preserving information approach avoids a new graph calculation from scratch, thus reducing the computational cost involved in the graph construction. Figure 1.2 presents

an online updating sequence of a Gabriel graph, in 3 different instants, from left to right. Nodes and edges that are discarded from the Gabriel graph are showed in red, and those that are newly inserted appears in blue.

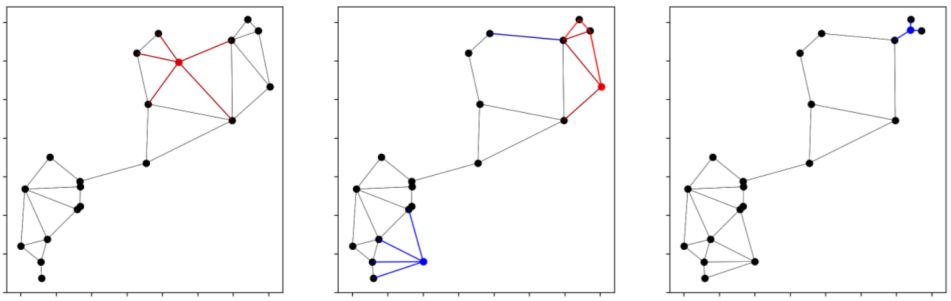


Figure 1.2: Online updating of a Gabriel graph (from the left to the right). Nodes and edges that are discarded, or inserted, are presented in red and blue, respectively.

- **An independent dominating set (IDS) algorithm:** The proposed algorithm induces an independent dominating set with a recurring process, using a metric to define a predefined order of dominance for the nodes of a graph (i.e. a Gabriel graph). The metric is based on the normalized degree combined with the normalized mean distance to adjacent of each node. These values are directly extracted from matrices of distance and adjacency, both computed when the Gabriel graph was induced. Thus, the method does not require parameters setting, once the solution is obtained from the data structure. Figure 1.3 shows an example of independent dominating set induced with the proposed algorithm. The graph on the left shows an initial set of points; on the center, is shown the Gabriel graph with the found dominating points (in red) and non-dominating points (in black); and on the right, only the dominating points.

- **Online KDE Estimator:**

  The approaches for online update of Gabriel graph and the independent dominating set algorithm are jointly used to define the parameters of an KDE Estimator, applied to the task of online classification of non-stationary data streams.

  The number of kernels and their centers are defined setting each kernel in each dominating point. Then, the kernel radius is determine as the mean value of the distances from each

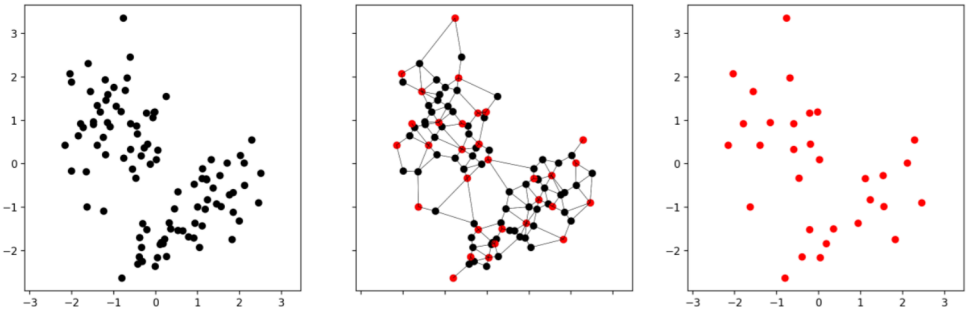


Figure 1.3: Example of an independent dominating set found with the proposed approach. On the left, an initial set of points; on the center the Gabriel graph with dominating points (in red) and non-dominating points (in black); and on the right, only dominating points.

non dominating point to its close dominating point (for each class). Thus, all parameters of the KDE estimator are directly defined from the data structure. A example of this approach is presented in Figure 1.4, showing the density estimation and the decision boundary found for the binary data set Two moons. The graph on the left, presents the points of each class in different colors, with dominating and non dominating points in nonempty and empty dots, respectively, while the graph on the right shows the density estimation with the dominating points, and the decision boundary (line in black).

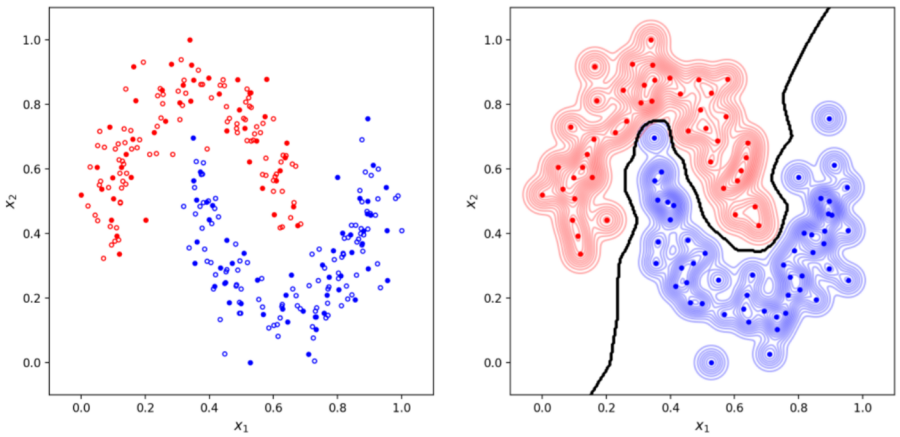


Figure 1.4: Graph on the left, presents the Two moons data set, with dominating and non dominating points, respectively, in nonempty and empty dots; graph on the right shows the density estimation with the dominating points. The decision boundary appears as a line in black.

- **One-class RBF Neural Network:** A training procedure for one-class RBF neural networks is propose, when the objective is to induce a function capable to define a decision boundary for a set of points that represents only one class. The dominating set induced from a Gabriel graph, is used to determine the parameters and number of Radial functions of the RBF network. Such procedure enables the definition of a single and stable radius for the Radial functions, centering each one of these, in each dominating point, while no parameters in the network output layer is required. Lastly, the decision boundary is defined choosing a number of training points to be outside of the established limit. Thus, all parameters of the RBF network are defined based on information from the structure of data. An example of the proposed solution is presented in Figure 1.5, for one of the classes of the Spiral data set. In the graph, dominating points are shown in red dots, non-dominating points are presented in black dots, and the decision boundary that encloses all points of the data set appears in grey.

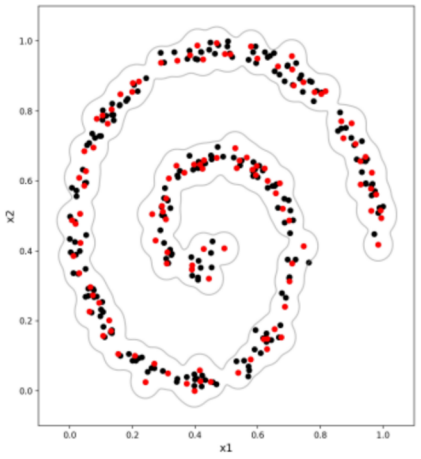


Figure 1.5: Decision boundary defined with the proposed one-class RBF network. The graph shows dominating points (in red), non-dominating points (in black) and the limit established (line in grey).

- **An Extractor Algorithm of Representative Subsets:** The proposed algorithm was conceived with the objective to obtain representative subsets with a smaller number of points from an original data set. The method relies on the use of the proposed independent dominating set algorithm and the Gabriel graph, that are induced from subsets of the original data set. The method has different levels of points selection, in which nested subsets present a monotonic decrease numbers of points, however with capability to approximate

the original data set. An example of the proposed method is presented in Figure 1.6, which shows the original data set, on the left, and the subsets of three extraction levels that were used. The number of points in each set is presented above each graph.

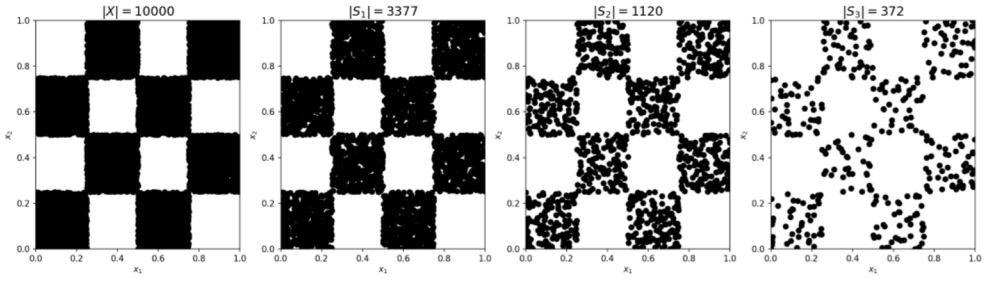


Figure 1.6: Representative subsets ($S_3 \subset S_2 \subset S_1 \subset X$) obtained with the proposed algorithm.

- **Online Regularized SLFN network with Random Projections:** An online training method for SLFN networks, that was named Single Hidden Layer Feedforward Neural Network based on Random Projections (SLFN-RP) [Alvarenga et al., 2021], is proposed to be used with some drift detection algorithm. From an initial set of points, the SLFN-RP randomly defines weights of the hidden layer. Tikhonov's regularization [Deng et al., 2009] is employed, using a direct approach of the leave-one-out cross validation (LOO-CV) to estimate a coefficient, with the initial set of points. Output weights are calculated using the least squares method. An adaptive window is used to enable the online learning and to adapt the network when a drift is detected. The method perform a quick network update, with matrix operations, and an inverse matrix calculation that it is only required after drifts.

1.4 Thesis Organization

This thesis is organized in 6 chapters. Next to chapter 1, chapter 2 presents a background with definitions that are the foundation of the proposed methods described in the following chapters. Chapter 3 presents the proposed methods: online updating of the Gabriel graph, Independent dominating set algorithm, and online KDE estimator, following by a section of experiments. Chapter 4 presents the following proposed methods: one-class RBF network and the algorithm that extracts representative subsets. A section of experiments presents comparative

tests and a test with a data set from a real industrial process. Chapter 5 describes the online regularized SLFN network with random projections, that is trained with the method named SLFN-RP, and presents a section of comparative tests. Chapter 6 presents some proposes for future work and states the final conclusions of this work.

Chapter 2

Background

2.1 Introduction

This Chapter presents a background with some important definitions that support the theory of the proposed methods in this thesis. The Chapter presents the following sections: Gabriel graph and Dominating sets, Statistical learning theory, Kernel density estimation, RBF and ELM neural networks, online learning and one-class classifier. These topics are the foundations of the Chapter 3, *Online learning with Gabriel graph, dominating sets and KDE estimator*, Chapter 4, *One-class RBF network and representative subsets*, and Chapter 5, *Online learning with a regularized SLFN network based on random projections*.

2.2 Gabriel Graph and Dominating Sets

A graph is a mathematical structure consisting of two sets: a finite number of nodes, representing points or elements of a particular problem, and a collection of edges connecting pairs of nodes, according to some relation [Haynes et al., 1998]. The Voronoi diagram [Sack and Urrutia, 1999], defined for a set of points (or sites), is formed by a set of regions that are each one limited by a finite number of half-planes, dividing the space where the set of points lie. The Delaunay triangulation [Su and Drysdale, 1997] has a one-one correspondence with the Voronoi diagram, being constructed with straight edges between each pair of points situated in two distinct Voronoi regions, that share a same half-plane.

The Gabriel graph is a subgraph of the Delaunay triangulation [Bhattacharya et al.,

1981], that although originally defined for problems in 2 dimensions, accepts generalization in its formulation [Matula and Sokal, 1980] to accommodate a definition with points in a dimensional space higher than 2, arbitrary distances, numerical proximity functions or other variations [Bose et al., 2010]. The definition for the Gabriel graph and for dominating sets are presented in the following sections.

2.2.1 Gabriel Graph

Definition 1: an Gabriel graph $G = (V, E)$ is defined with a set of points $X = \{\{\mathbf{x}_i\}_{i=1}^{N} | \mathbf{x} \in \mathbb{R}^n\}$, used as nodes, and a collection of edges E obeying the following definition

$$(\mathbf{x}_i, \mathbf{x}_j) \in E \leftrightarrow \|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq (\|\mathbf{x}_i - \mathbf{x}_k\|^2 + \|\mathbf{x}_j - \mathbf{x}_k\|^2) \quad \forall \mathbf{x}_k \in V, \mathbf{x}_i, \mathbf{x}_j \neq \mathbf{x}_k. \tag{2.1}$$

The nodes connections, defined with Equation 2.1, can be represented by an adjacent matrix $\mathbf{M}$, expressed by

$$m_{ij} = \begin{cases} 1, & if\ edge\ (\mathbf{x}_i, \mathbf{x}_j) \in E \\ 0, & otherwise \end{cases} \tag{2.2}$$

A fundamental characteristic of the Gabriel graph structure is the preservation of local relation of the nodes, preventing an edge $(\cdot, \cdot)$ between two nodes, v_i and v_j, with distance d, to exist, whenever a third node v_k is located inside of a hypersphere with diameter d, centered at the midpoint of the line segment which connects v_i and v_j.

2.2.2 Dominating Sets

The cardinality of a set V is represented by $|V|$. Two nodes are adjacent if they are connect by an edge. The set of nodes adjacent to a node v is called its open neighborhood $N(v)$. The close neighborhood of node v is defined as $\{v\} \cup N(v)$. The number of edges incident with a node v is called degree $deg(v)$. The maximum degree $\Delta(G)$ of a graph G is defined as $max\{deg(v) : v \in V\}$. The minimum degree $\delta(G)$ of a graph G is defined as $min\{deg(v) : v \in V\}$. The following Definitions (2-3-4-5) are borrowed from [Haynes et al., 1998]:

Definition 2: a dominating set D of a graph $G = (V, E)$ is defined as a subset of V, so that every node in the set D is adjacent to some node in $V - D$. Or equivalent, $\forall v \in D,\ N(v) \cap V - D \neq \emptyset$.

Definition 3: every connected graph $G = (V, E)$, with $|V| \geq 2$, has a dominating set D whose complement $V - D$ is also a dominating set.

Definition 4: an independent set I of a graph $G = (V, E)$ is a subset of V, in which no two nodes in I are adjacent. Or equivalent, $\forall (v, u) \in I,\ N(u) \cap v \neq \emptyset$.

Definition 5: an independent dominating set (IDS) $S \subseteq V$ of a graph $G = (V, E)$ is both an independent set and dominating set in V.

2.3 Statistical Learning Theory

Inductive models like neural networks induce a set of functions $f(\mathbf{x}, \mathbf{w})$, where $\mathbf{w} \in \Lambda$ represents the parameters, aiming to estimate an unknown relation between spaces $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}$ [Vapnik, 1995]. Such relation is fully described by an joint probability distribution $p(\mathbf{x}, y)$ where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \{-1, 1\}$ are sampled, respectively, from a fixed $p(\mathbf{x})$ and $p(y|\mathbf{x})$. Assessment of a inferred function $f(\mathbf{x}, \mathbf{w})$ is done with a loss function L, minimizing the Risk Functional $R(\mathbf{w})$ [Vapnik, 1992], defined as,

$$R(\mathbf{w}) = \int L(y, f(\mathbf{x}, \mathbf{w}))p(\mathbf{x}, y)dxdy, \tag{2.3}$$

where L is some loss function and $p(\mathbf{x}, y)$ is a the joint probability.

Usually $p(\mathbf{x}, y)$ is unknown, thus the Empirical Risk Functional $R_{emp}(\mathbf{w})$, defined in Equation 2.4, take the place of $R(\mathbf{w})$.

$$R_{emp}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(y, f(\mathbf{x}, \mathbf{w})), \tag{2.4}$$

in which N is the number of points in a set used in the learning process.

Although the minimization of the Empirical Risk Functional $R_{emp}(\mathbf{w})$ aims to provide the function $f(\mathbf{x}, \mathbf{w})$ with the smaller error, for a given training data, it is also desirable a solution with generalization ability. For this, one possible approach is the Structural Risk Minimization (SRM) inductive principle, which aims to minimize the risk functional and a function capacity term, which defines the capacity of the approximation function [Vapnik, 1992]

The Regularized Risk functional is a way to implement the SRM principle, enabling the definition of an appropriate trade-off of these two objectives, introducing a regularization parameter λ, that weights the influence of the $\|\mathbf{w}\|$ over the smoothness of the solution. The Equation 2.5 presents the loss function that is minimized when regularization is employed to estimate $f(\mathbf{x}, \mathbf{w})$.

$$L(y, f(\mathbf{x}, \mathbf{w})) = (f(\mathbf{x}, \mathbf{w}) - y) + \lambda\|\mathbf{w}\|. \tag{2.5}$$

2.4 Kernel Density Estimation

A kernel Estimator of kernel $K(\mathbf{x}_i, \mathbf{x}_j)$, induced from a set of point $X = \{\{x_i\}_{i=1}^m | x \in \mathbb{R}\}$, is defined by Equation 2.6 [Silverman, 1986].

$$\hat{p}_X(x) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{h}K\left(\frac{x - x_i}{h}\right) \tag{2.6}$$

where m is the total number of points in X, h is the kernel radius, and K is a kernel that needs to satisfy Equation 2.7.

$$\int_{-\infty}^{\infty} K(x)dx = 1 \tag{2.7}$$

In the multivariate case, with a set of points $X = \{\{\mathbf{x}_i\}_{i=1}^m | \mathbf{x} \in \mathbb{R}^n\}$, the kernel K can be done a multidimensional function, satisfying 2.7, or to use univariable kernel function, for each one of the n dimensions with its own radius h_j, for $j = (1, \cdots, n)$. Equations 2.8 and 2.9 present the former and later options, respectively. It is possible to assume a unique value of h in Equation 2.8, with a multidimensional kernel that satisfy Equation 2.7, as presented in Equation 2.10.

$$\hat{p}_X(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{h_1 \cdots h_n}K\left(\frac{x_1 - x_{i1}}{h_1}, \cdots, \frac{x_n - x_{in}}{h_n}\right) \tag{2.8}$$

$$\hat{p}_X(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m}\left[\prod_{j=1}^{n}\frac{1}{h_j}K\left(\frac{x_j - x_{ij}}{h_j}\right)\right] \tag{2.9}$$

$$\hat{p}_X(\mathbf{x}) = \frac{1}{mh^n}\sum_{i=1}^{m}K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{2.10}$$

Although the radius h of the kernel be the only parameter of the KDE Estimator in Equation 2.10, its value has strong influence over the bias and variance dilemma [Geman et al., 1992]. Thus, its definition is a important step towards a model with adequate Capacity [Vapnik, 1995].

2.5 RBF Neural Networks

The output of an RBF network [Haykin, 1994], for the particular case of only 1 neuron in the output layer, is given by Equation 4.1.

$$f(\mathbf{x}, \mathbf{C}, \mathbf{h}) = \sum_{i=1}^{p} w_i r_i(\mathbf{x}, \mathbf{c}, h) + w_0, \tag{2.11}$$

where p is the number of neurons in the hidden layer, $r_i(\mathbf{x}, \mathbf{c}, h)$ represents a Radial function of the hidden neuron i, centered in $\mathbf{c}$ and with radius of h, $\mathbf{C}$ and $\mathbf{h}$ are a matrix and a vector containing the centers e radius of each Radial function, respectively, and $\mathbf{w} = [w_0, w_1, ..., w_p]$ is a vector containing the weights from the output layer.

The Radial function perform a mapping based on their dispersion scale which is applied to the distance between the presented argument $\mathbf{x}$ and the function center $\mathbf{c}$. Among an infinity of possible Radial functions, one that stands out as an option in the construction of RBF networks is the Gaussian function, defined in Equation 2.12.

$$r(x, c, h) = e^{\frac{-(x-c)^2}{2h^2}}, \tag{2.12}$$

where $h > 0$ is the radius of the Gaussian function.

2.6 ELM Networks

The Extreme Learning Machines [Huang et al., 2004, 2006b] are neural networks that presents a fast training procedure, requiring only the number of hidden neurons to be previously set. The effectiveness of the random projections was presented in [Schmidt et al., 1992] where it is shown that output layer weights of SLFNs networks have more importance than their hidden layer parameters.

The original proposed SLFN trained with the ELM algorithm is given by,

$$f = \sum_{i=1}^{p} w_i h(\mathbf{x_i}) = \sum_{i=1}^{p} w_i h(\mathbf{z_i x_i} + z_{bi}), \tag{2.13}$$

where p is the number of hidden neurons, $\mathbf{z_i}$ and z_{bi} are the weight vector and bias, respectively, connecting inputs to the *i*th neuron, $h(\cdot)$ is a nonlinear activation function, and w_i is the weight connecting neuron i to output.

The ELM method refrains from estimating all the network parameters as it randomly defines the weights values for hidden neurons. This approach avoids the use of iterative gradient descent methods, such as the back-propagation, which requires the use of hyperparameters.

Equation 2.13 can be rewritten in matrix form as,

$$\mathbf{Hw} = \hat{\mathbf{Y}}, \tag{2.14}$$

where the matrix $\mathbf{H}$, containing the hidden layer output, is defined as,

$$\mathbf{H} = \begin{bmatrix} h(\mathbf{z}_1 \cdot \mathbf{x}_1 + z_{b1}) & \dots & h(\mathbf{z}_p \cdot \mathbf{x}_1 + z_{bp}) \\ \dots & \vdots & \dots \\ h(\mathbf{z}_1 \cdot \mathbf{x}_N + z_{b1}) & \dots & h(\mathbf{z}_p \cdot \mathbf{x}_N + z_{bp}) \end{bmatrix}_{N \times p} . \tag{2.15}$$

For the approximation of N observations by the network, the following relation is set

$$\mathbf{Hw} = \hat{\mathbf{Y}} = \mathbf{Y}. \tag{2.16}$$

Then the remaining parameters of the network (output weight matrix) are analytically calculated using the least squares method,

$$\mathbf{w} = \mathbf{H}^{+}\mathbf{Y}, \tag{2.17}$$

where $\mathbf{H}^{+} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ is the pseudoinverse of matrix $\mathbf{H}$. Equation 2.17 results in the training error minimization with minimum norm solution to the least squares problem defined by $\|\mathbf{Hw} - Y\|$ [Huang et al., 2004].

SLFN networks trained with ELM algorithm have the universal approximation property if some conditions regarding weight initialization and activation functions are met [Huang et al., 2006a]. The **ELM algorithm** can be summarized as follows: given a training set $\{(\mathbf{x}_i, y_i)_{i=1}^N | \mathbf{x} \in \mathbb{R}^n, y \in \{-1, 1\}\}$, activation function $h(\cdot)$, and p hidden neurons,
Step 1: Assign random values to $\mathbf{z}_i$ and z_{bi}, for $i = 1, ..., p$.
Step 2: Calculate the hidden layer output $\mathbf{H}$.
Step 3: Calculate the weights $\mathbf{w}$ for the output layer.

2.7 One-class classifier

In general, problems of fault detection, health diagnosis, banking and credit analysis, and others that are misrepresented by one of the classes, usually have points from the normal

class, labeled as negative, in enough number to provide an acceptable representation of this data condition. However, abnormal patterns, which constitute the positive class, are rare or absent, thus not sufficient to delineate a decision boundary, possible because the class is poorly sample and/or it is ill-defined. In this type of problem the boundary that separates positive from negative points may be induced with an one-class classifier, using only points from one of the classes. In this approach the learning is attained in the known class to established a separable boundary.

2.7.1 Problem Formulation of Learning with One-class Classifier

Let a set $X = \{\{\mathbf{x_i}\}_{i=1}^{N} | \mathbf{x} \in \mathbb{R}^n\}$, with N points, represents an unknown distribution function $\mathcal{X}$. The objective it to induce a function $f(\mathbf{x})$ from X, capable to establish a decision boundary for the points in X, such that all points in X are founded inside of this defined space, or some of the points are founded outside of the decision boundary, possibly as outliers. The function $f(\mathbf{x})$ must be capable to classify a new point $\mathbf{x}$ as belonging to the class stated, or not belonging.

2.8 Online Learning

The next sections introduce the following items: types of online Learning, the problem formulation of the Supervised Online Learning, concept drift, and some drift detection methods.

2.8.1 Types of Online Learning

Online learning can be classified into the 3 main following categories, with respect to the feedback information [Hoi et al., 2021], which aims to provide the true value of the output variable to the predictive model:

- **Online supervised learning:** where full feedback information is always revealed to a learner at the end of each learning iteration.
- **Online learning with limited feedback:** when feedback information is provided partially for some of the data points, still being able to arrive with delay.
- **Online unsupervised learning:** in which the learner receives only the data instances without any additional feedback (i.e., no true class label).

Regarding how concepts drifts are dealt, online learning algorithms can also be grouped into two classesDitzler et al. [2015]:

- **Actives:** that from the detection of a concept change (performed by the model itself or by some auxiliary method), use some mechanism to adjust the model to the new scenario.
- **Passives:** those that are robust to data changes, relying only in the continuous model updating, as new training data points become available.

A model of online learning from examples is showing the in Figure 2.1, in which the components of a Passive (blocks in blue) and an Active (blocks in green and blue) types are presented.

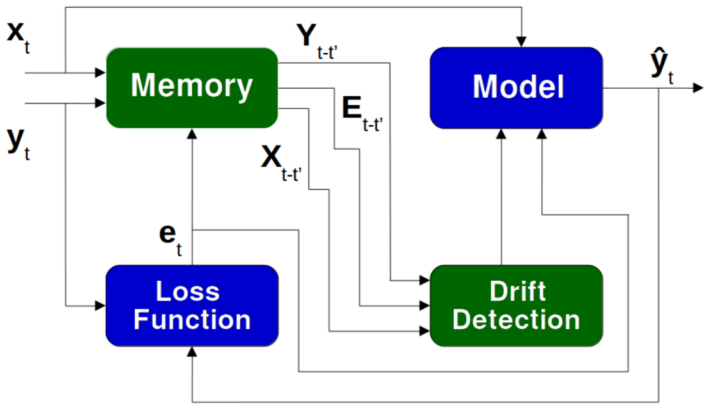


Figure 2.1: Types of online learning algorithms, Passives (blocks in blue) and Actives (blocks in blue and green), with respect to how concept drifts are treat.

2.8.2 Problem Formulation of the Supervised Online Learning

Given a data generating function $\Phi = \{\mathbf{x}_t, y_t\}_{t=1}^{N}$, where $\mathbf{x}_t \in \mathbb{R}^n$ represents the attributes, $y_t \in \mathbb{R}$ a response variable of a problem, and t an instant of time, the Supervised Online Learning paradigm can be formulated as the estimation of an approximation function $f(\mathbf{x}, \mathbf{w})$ which occurs whenever $\mathbf{x}_t$ is presented to the model, its prediction $f(\mathbf{x}_t, \mathbf{w}_{t-1})$ is calculated and the true label y_t is presented. Then a loss function is used to calculate the error $\epsilon_t = L(y_t, f(\mathbf{x}_t, \mathbf{w}_{t-1}))$. The previous model parameters $\mathbf{w}_{t-1}$ and the error ϵ_t are used to update the function f. These steps occur iteratively, as new points are provided. In order to initialize the learning process, an initial model $f(\mathbf{X}_0, \mathbf{w}_0)$ is generated from a small set of data $\mathbf{X}_0$.

2.8.3 Concept Drift

In a non-stationary data stream, changes in the data generating function $\Phi = \{\mathbf{x}_t, y_t\}_{t=1}^{N}$ may occur at any time, modifying the joint probability distribution $p(\mathbf{x}, y)$ that previously described the mapping relationship between the spaces $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}$. This event is known as Concept Drift and can be formally defined by [Gama et al., 2014],

$$p_{t=0}(\mathbf{x}, y) \neq p_{t=1}(\mathbf{x}, y). \tag{2.18}$$

Concept drift may occur due to one or more factors combined. The expression of the joint probability function as $p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y) = p(y|\mathbf{x})p(\mathbf{x})$ shows possible generating terms of a data change [Takahashi and Braga, 2020]. For a model that learns a data stream, changes are generally categorized as: *Real* concept drift, in which $p_{t_0}(\mathbf{y}|\mathbf{x}) \neq p_{t_1}(\mathbf{y}|\mathbf{x})$; and *Virtual* concept drift, described by $p_{t_0}(\mathbf{y}|\mathbf{x}) = p_{t_1}(\mathbf{y}|\mathbf{x})$, even though changes in data have occurred (i.e. in $p(\mathbf{x})$) [Elwell and Polikar, 2011]. Another cause of concept drifts are hidden contexts due to insufficient information in the predictive features.

Concept Drifts can be also discriminated with respect to the rate of change [Takahashi and Braga, 2020]. Concerning duration drifts can occur in a abrupt and gradual way, being incremental a particular case of the last one. Regarding return to previous state, recurrent, periodic and not periodic are possible types, while outliers do not account as drifts. Figure 2.2 show some examples of these types.

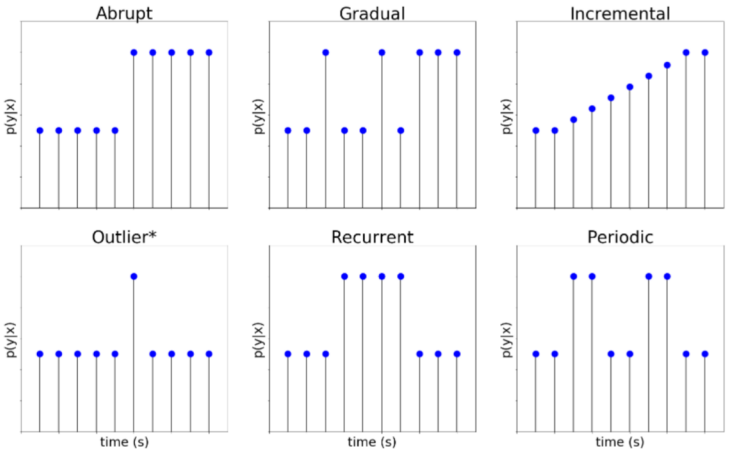


Figure 2.2: Different types of concept drifts. Outliers* do not account as drifts

2.8.4 Drift Detection Methods

In order to mitigate the negative impact of changes in the data when learning with non-stationary data streams, one approach is to use drift detection methods associated with the prediction model. Such action enable that the model model be adapted to the new scenario, once a concept drift is detected. Change detection techniques aim to identify changes in data that characterize a change in their generating function. For this, these methods directly analyze the data points or extract statistical characteristics from the latter and the prediction error. The use of sliding windows [Gama et al., 2004; Bifet and Gavalda, 2007] offers ways to extract this information, enabling the comparison of two different instants.

One of the first drift detection methods, named FLORA [Widmer and Kubat, 1996], processes data sets arranged in attribute-value pairs, using a sliding window. The proposal uses 3 distinct sets to separate the hypotheses into positive, negative and belonging to both, allowing the hypotheses to move between the sets, along the process. The extensions FLORA2, FLORA3 and FLORA4 were also proposed, respectively, to address the dynamic adjustment of the window size. to apply a concept reservoir to be used in the analysis of a new concept drift, and to treatment the noise presented in the input data.

A drift detection approach that uses Support Vector Machines to detect changes in a incremental batch of data is presented in [Klinkenberg and Joachims, 2000]. The method selects a size for a training window so that the estimated generalization error on new examples is minimized. The method uses an approach similar to the leave-one-out estimation to quantify the error.

A learning approach based on a window that grows when no change is apparent and shrinks when data has changed is presented in [Bifet and Gavalda, 2007]. The method named ADWIN, dynamically adjusting the length of a data window, with the most recently points, estimating statistics in two subwindows. Whenever different values are found, according to a predefined confidence level, the older subwindow is discard, assuming that a change has occurred.

Hellinger distance is a non-parametric metric used to quantify the similarity between probability distributions. Its values are symmetric and bounded in the $[0, \sqrt{2}]$ interval. The Hellinger distance drift detection method (HDDDM) [Ditzler and Polikar, 2011] is based on

histograms with predefined number of bins along each descriptive variable, with

$$\delta_t = \frac{1}{n}\sum_{k=1}^{n}\sqrt{\sum_{i=1}^{b}\left(\sqrt{\frac{P_{i,k}}{\sum_{j=1}^{b}P_{j,k}}}-\sqrt{\frac{Q_{i,k}}{\sum_{j=1}^{b}Q_{j,k}}}\right)^2}, \tag{2.19}$$

in which $b = \lfloor\sqrt{N}\rfloor$ is the number of bins, N is the number of points, n is the dimensionality, and P and Q are the mass distributions from current and previous instant, respectively.

When a new point becomes available, Equation 2.19 is calculated, as well as the difference between current and past measures, $\epsilon_t = \delta_t - \delta_{t-1}$. Subsequently, ϵ_t is then compared to a threshold β_t to decide whether the drift happened or not. It is suggested in [Ditzler and Polikar, 2011] to calculate β with two different methods: using mean and standard deviation, and with *t*-statistic test. The first one is defined by

$$\beta_t = \hat{\epsilon} + \gamma\hat{\sigma}, \tag{2.20}$$

where $\hat{\epsilon}$ and $\hat{\sigma}$ are the mean and standard deviation of ϵ, and γ is a given parameter.

2.9 Conclusion

This chapter presented some basic knowledge and definitions for the Gabriel graph, Dominating sets, Kernel density estimation, RBF and ELM neural networks, online learning and one-class classifier. These information that are the foundation for the proposed methods, described in Chapter 3, *Online learning with Gabriel graph, dominating sets and KDE estimator*, Chapter 4, *One-class RBF network and representative subsets*, and Chapter 5, *Online learning with a regularized SLFN network based on random projections*.

Chapter 3

Online Learning with Gabriel Graph, Dominating Sets and KDE Estimator

3.1 Introduction

This Chapter describes 3 methods that are proposed in this thesis. The first is a proposal to online update the Gabriel graph, with a new point that becomes available, using a slide window and an approach that saves information from the graph in 3 matrices, in each update step. The second is an Independent dominating set algorithm, that use as input: a set of points, its respective matrix of distance, and an adjacent matrix representing the nodes connection of a graph. The process to induce the dominating set use a predefined order of dominance, that trade-off the normalized degree and the normalized mean distance of each node from the graph to provide a representative subset in the output, without need to setting parameters. The third proposal is an online KDE estimator, which has all of its parameters: number of kernels, centers and radius, defined with the jointly use of the online update the Gabriel graph and the IDS algorithm. These two last provide the data structure of the most recent points of a data stream, that is saved in a sliding window.

A section of related works introduces some previous approaches from the literature, following by the description of each method. Next, a section with experiments describes some comparative and time complexity tests that were conducted with the proposed methods.

3.2 Related works

The Gabriel graph was used in [Torres et al., 2014] to find the closer points of two separable classes. In this work, these points are called Geometric Vectors and are paired connected by an edge of the graph. In this large margin approach, the geometric vectors are used to find a separating hyperplane of minimum norm, calculated with the Moore-Penrose linear pseudoinverse. The proposed method showed results statistically equivalent to those obtained with SVMs. The approach in [Torres et al., 2014], was modified to be used with a density mixture of model in [Torres et al., 2015b]. Gaussian functions were used, with diagonal covariance matrices, containing diagonal elements $\sigma^2(l) = 2\|x_i(l) - mp(l)\|$, where l represents the variance for the l-th dimension, and mp is the midpoint between the two points x_i and x_j. The work in [Torres et al., 2020] propose a modification in [Torres et al., 2015b], with the variance defined as $\sigma^2 = R/3$, where $R = 0.5\|x_i(l) - mp(l)\|$. An important characteristic of these previous approaches is that neither any parameter needs to be set in advance and nor user interaction is required to balance bias and variance trade-off.

A method that optimizes the width of RBF kernels from an SVM is presented in [Menezes et al., 2019]. For a given training set of points, the approach uses density information from a likelihood space, and a dissimilarity function between the existing classes. An optimal width value is found in an uni-dimensional optimization problem, in which one should maximize the dissimilarity function.

The identification of geometric vectors with the Gabriel graph also appears in [Torres et al., 2015a], as a suitable approach for implementation in integrated circuits. In this work the final decision rule is a linear combination of linear classifiers, that is defined in each midpoint formed by a edge connecting two geometric vectors. The approach in [Torres et al., 2015a] was subsequently implemented in an FPGA platform [Arias-Garcia et al., 2020], using a different decision rule. In this work, the nearest neighbors method, with $k = 1$, and the quadratic Euclidean distance were used.

A method to determine the number and the parameters of Radial functions from an RBF network, using a Dominating Set of the Gabriel graph is found in [Queiroz et al., 2022]. The property of the Dominating Set ensures that all non dominating nodes in the graph are covered by the adjacent dominating nodes. The Radial functions are centered in the last, while the radii are defined as the average distance between the adjacent non dominating linked to

each dominating node. The proposed online classifier presented in this chapter, aims to extend this concepts, as an online KDE estimator, with a new approach to induce an independent dominating set from a Gabriel graph that is updated online, and a different approach to define the kernel radius.

An approach to define an Connected Dominating Set using genetic algorithm was presented in [Varsa and Sridharan, 2019]. In this type of dominance the dominating nodes are connected in the graph. The method considers multiple objectives with a fitness function to select the nodes, in a optimized process that requires hyperparameters setting. The approach is tested with simulated data of wireless sensor networks.

A KDE Estimator was presented in [Wanderley et al., 2013], aiming to solve problems with high dimension domain and with a small number of points available. The method uses a evolutionary strategy to search the space, selecting proper subsets of features based on a measure of fitness that considers the geometric mean metric to reward individuals with sensitivity and specificity balanced. After the subsets of features are identified, the data is modeled with a KDE Estimator and a Bayesian classifier is used to separates the classes.

The self-organizing maps (SOM) [Kohonen, 1982; Araujo and Rego, 2013] are a type of unsupervised neural networks that represent the input data, using nodes on a regular grid, according to a similarity relation between input space and the topological structure in low-dimension that is built. An interesting use of this unsupervised network is in the definition of center and radius of Radial functions. One example is the method that determine the parameters of the Radial functions in parallel with the output weights adaptation of an RBF neural network [Fritzke, 1994]. The method use a SOM network to create new Radial functions incrementally, based on error of the network, until a predetermined value of prediction performance is obtained. The radius is defined as the mean length of all edges that connect to the center of each function. A variation of this approach that follows a non-stationary data stream, removing nodes of an SOM network, from low density regions appears in [Fritzke, 1997].

A two-stage algorithm for real-time fault detection and classification was proposed in [Costa et al., 2015]. The detection stage is carried out with a density approach, using Cauchy function that is calculated recursively, while the classification stage is done with a evolving fuzzy-rule-based classifier. The method is able to perform detection and classification of different types, lengths and levels of faults, in a fully unsupervised and on-line manner, even though

an initial parameter needs to be set.

The proposed online KDE estimator that is presented in this chapter is compared with two models from the literature: the Hoeffding tree classifier, with ADWIN (HAT-ADWIN) [Bifet and Gavalda, 2009], and the ensemble classifier Dynamic Weighted Majority [Kolter and Maloof, 2007], with Naive Bayes as base learners. These methods are capable to learn with data streams, requiring the configuration of hyperparameters.

3.3 Online update of Gabriel Graph

Generally, in applications that require calculation of the Gabriel graph in online scenarios, where data points arrive sequentially, the general approach is to iteratively create each graph from scratch — whenever a new point is available. This solution is computational costly considering no information is preserved in the construction of the graph for two different iterations, while the difference between two sets that define the graphs is only one point. Incremental calculation of the graph, however, is non-trivial since a change in only one point may result in significant changes in the edge configuration of the graph. The proposed approach consists of calculating the Gabriel graph iteratively while preserving information between two consecutive iterations, i.e., it refrains from repetitively calculating the graph from scratch whenever a new point is available and an older point is disregarded.

In this context, consider a sliding window $W = [\mathbf{X}]$, with a sequence of m points $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_m]^T$, from a data generating function $\Phi = \{\mathbf{x}(t)\}_{t=1}^{N}$. The idea is to induce an initial Gabriel graph G from W, and to keep updating G as W slides, with the insertion of new points $\mathbf{x}$, one-by-one, while the oldest point in W is discarded.

3.3.1 Proposed Approach

Let an initial matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]^T$, with points $\mathbf{x} \in \mathbb{R}^n$, and a matrix $\mathbf{D}$, with dimensions $m \times m$, containing Euclidean distances $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2,\ \forall(i,j)$. A matrix $\mathcal{P} = [\mathbf{P}_1\ \mathbf{P}_2\ \cdots \mathbf{P}_m]^T$, with dimensions $(m \cdot m) \times n$, and sub-matrices $\mathbf{P}_k = [\mathbf{p}_1^k\ \mathbf{p}_2^k\ \cdots\ \mathbf{p}_m^k]$, with dimensions $m \times n$, containing midpoints $\mathbf{p}_j^k = \frac{(\mathbf{x}_k + \mathbf{x}_j)}{2}$, of line segments connecting points $\mathbf{x}_k$ and $\mathbf{x}_j$, $\forall j, k = (1, \cdots, m)$. Matrix $\mathcal{Q} = [\mathbf{Q}_1\ \mathbf{Q}_2\ \cdots\ \mathbf{Q}_m]^T$, with dimensions $(m \cdot m) \times m$, and symmetric sub-matrices $\mathbf{Q}_k$, with dimensions $m \times m$, containing distances $q_{ij}^k = \|\mathbf{x}_k - \mathbf{p}_j^i\|^2$,

from a point $\mathbf{x}_k$ to a midpoint $\mathbf{p}_j^i$, $\forall(i,j),\ k = (1, \cdots, m)$. Then, the adjacent matrix $\mathbf{M}$ of a Gabriel graph, induced from $\mathbf{X}$, is defined with Equation 3.1, using the distances in matrices $\mathbf{D}$ and $\mathcal{Q}$.

$$m_{ij} = \begin{cases} 1, & if\ \frac{d_{ij}}{2} \leq q_{ij}^k, \quad \forall i \neq j, \\ & \qquad\qquad \forall k \neq (i,j) \\ 0, & otherwise, \end{cases} \tag{3.1}$$

where m_{ij} and d_{ij} are elements of matrices $\mathbf{M}$ and $\mathbf{D}$, respectively, and q_{ij}^k is an element of sub-matrix $\mathbf{Q}_k \in \mathcal{Q}$. Constraints $\forall i \neq j$ and $\forall k \neq (i,j)$ prevents the occurrence of situations with midpoints formed with same point $\mathbf{p}_j^j$, and distance q_{ij}^i from a point $\mathbf{x}_i$ to a midpoint $\mathbf{p}_j^i$ or $\mathbf{p}_i^j$, respectively. Figure 3.1 illustrates the points, midpoints and distances used in the definition of the proposed approach.

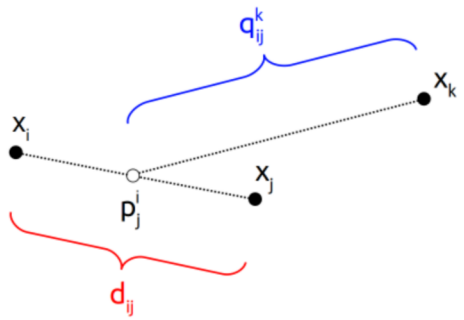


Figure 3.1: Scheme illustrating points, midpoints and distances used in the approach to update the Gabriel graph.

The update process of matrices $\mathbf{D}(t)$, $\mathcal{P}(t)$ and $\mathcal{Q}(t)$, with insertion of a new point $\mathbf{x}(t)$ in W, and discard of the oldest point, is presented next. These steps enable the definition of matrix $\mathbf{M}(t+1)$. Let $\mathbf{X}(t) = [\mathbf{x}(t), \mathbf{x}(t-1), \cdots, \mathbf{x}(t-m+1)]^T$. The update of matrix $\mathbf{D}(t)$ requires the computation of a vector $\mathbf{d}$ with $(m-1)$ distances from point $\mathbf{x}(t)$ to each point in $[\mathbf{x}(t-1), \mathbf{x}(t-2), \cdots, \mathbf{x}(t-m+1)]^T$. Vector $\mathbf{d}$ is inserted as first row and column in $\mathbf{D}(t)$,while its last row and column, containing distances from the oldest point is discarded. Equations 3.2 and 3.3 present matrices $\mathbf{D}(t)$ and $\mathbf{D}(t+1)$, respectively.

$$\mathbf{D}(t) = \begin{bmatrix} \mathbf{D}^* & \mathbf{d}_o \\ \mathbf{d}_o^T & 0 \end{bmatrix}, \tag{3.2}$$

$$\mathbf{D}(t+1) = \begin{bmatrix} 0 & \mathbf{d}^T \\ \mathbf{d} & \mathbf{D}^* \end{bmatrix}, \tag{3.3}$$

where $\mathbf{D}^*$ is a sub-matrix with dimensions $(m-1) \times (m-1)$, and $\mathbf{d}_o$ is a distance vector discarded.

The update of matrix $\mathcal{P}(t)$ demands the computation of matrix $\mathbf{P}_1 = [\mathbf{p}_1^1\ \mathbf{p}_2^1\ \cdots\ \mathbf{p}_m^1]$, with midpoints formed with point $\mathbf{x}(t)$ and each point in $\mathbf{X}(t)$. The sub-matrix $\mathbf{P}_m$ is discarded and $\mathbf{P}_1$ is inserted, while the last row of each sub-matrix $\mathbf{P}_k(t)$ is discarded, and each $\mathbf{p}_j^1$, in $\mathbf{P}_1$, is inserted in $\mathbf{P}_k(t+1)$, as first row ($j = k$). Equation 3.4 presents matrix $\mathcal{P}(t)$, representing its submatrices as $\mathbf{P}_k(t) = \mathbf{P}_k^{\ddagger}$.

$$\begin{aligned} \mathcal{P}(t) &= \begin{bmatrix} \mathbf{P}_1^{\ddagger} & \mathbf{P}_2^{\ddagger} & \cdots & \mathbf{P}_{m-1}^{\ddagger} & \mathbf{P}_m \end{bmatrix}^T, \\ \mathbf{P}_k^{\ddagger} &= \begin{bmatrix} \mathbf{P}_k^* & \mathbf{p}_o \end{bmatrix}^T, \qquad k = (1, 2, \cdots, m-1), \end{aligned} \tag{3.4}$$

where each $\mathbf{P}_k^*$ has dimensions $(m-1) \times n$, and $\mathbf{p}_o$ is a midpoint discarded. Equation 3.5 presents matrix $\mathcal{P}(t+1)$, representing its submatrices as $\mathbf{P}_k(t+1) = \mathbf{P}_k^{\dagger}$.

$$\begin{aligned} \mathcal{P}(t+1) &= \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2^{\dagger} & \mathbf{P}_3^{\dagger} & \cdots & \mathbf{P}_m^{\dagger} \end{bmatrix}^T, \\ \mathbf{P}_k^{\dagger} &= \begin{bmatrix} \mathbf{p}_j^1 & \mathbf{P}_{k-1}^* \end{bmatrix}^T, \qquad k = (2, 3, \cdots, m),\ j = k, \end{aligned} \tag{3.5}$$

The update of $\mathcal{Q}(t)$ requires the definition of a symmetric matrix $\mathbf{Q}_1$, with dimensions $m \times m$, containing distances q_{ij}^k, from point $\mathbf{x}(t)$ to each midpoint in $\mathcal{P}(t)$. Likewise, a matrix $\mathbf{B} = [\mathbf{b}_2\ \mathbf{b}_3\ \cdots\ \mathbf{b}_m]$, with dimensions $(m-1) \times (m-1)$, containing vectors $\mathbf{b}_k = [b_{ij}^k]$ with distances from each point $\mathbf{x}_k$, in $\mathbf{X}(t)$, with exception of $\mathbf{x}(t)$, to each midpoint in $\mathbf{P}_1$ is computed. Then, Equation 3.6 presents the update of matrix $\mathcal{Q}(t)$, with discard of sub-matrix $\mathbf{Q}_m$ and insertion of $\mathbf{Q}_1$. Last row and column of each sub-matrix $\mathbf{Q}_k(t)$ are discarded, while each vector $\mathbf{b}_k$ is inserted as first row and column of a sub-matrix $\mathbf{Q}_k(t+1)$. Equation 3.6 presents matrix $\mathcal{Q}(t)$, representing its submatrices as $\mathbf{Q}_k(t) = \mathbf{Q}_k^{\ddagger}$.

$$\begin{aligned} \mathcal{Q}(t) &= \begin{bmatrix} \mathbf{Q}_1^{\ddagger} & \mathbf{Q}_2^{\ddagger} & \cdots & \mathbf{Q}_{m-1}^{\ddagger} & \mathbf{Q}_m \end{bmatrix}^T, \\ \mathbf{Q}_k^{\ddagger} &= \begin{bmatrix} \mathbf{Q}_k^* & \mathbf{b}_o^T \\ \mathbf{b}_o & 0 \end{bmatrix}^T, \qquad k = (1, 2, \cdots, m-1), \end{aligned} \tag{3.6}$$

where each sub-matrix $\mathbf{Q}_k^*$ has dimensions $(m-1) \times (m-1)$, and $\mathbf{b}_o$ is a distance vector discarded. Equation 3.7 presents matrix $\mathcal{Q}(t+1)$, representing its submatrices as $\mathbf{Q}_k(t+1) =$

$\mathbf{Q}_k^\dagger$.

$$\mathcal{Q}(t+1) = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2^\dagger & \mathbf{Q}_3^\dagger & \cdots & \mathbf{Q}_m^\dagger \end{bmatrix}^T, \tag{3.7}$$

$$\mathbf{Q}_k^\dagger = \begin{bmatrix} 0 & \mathbf{b}_k^T \\ \mathbf{b}_k & \mathbf{Q}_{k-1}^* \end{bmatrix}^T, \qquad k = (2, 3, \cdots, m).$$

After the updating of matrices $\mathbf{D}$, $\mathcal{P}$ and $\mathcal{Q}$, matrix $\mathbf{M}$ is defined according to Equation 3.1. Algorithm 1 summarizes the process to update the adjacent matrix of a Gabriel graph, using a sliding window W with m points and unitary stride. The algorithm can be applied, without lost of generality, for cases with different numbers of points inserted and/or discarded, applying minor reformulation in the code.

input : $\Phi = \{\{\mathbf{x}_t\}_{t=1}^N | \mathbf{x} \in \mathbb{R}^n, N \to \infty\}, m$

output: $\mathbf{M}(t+1)$

$\mathbf{X} \leftarrow \Phi$; // initial set with m points

$\mathbf{D}(0) \leftarrow dist(\mathbf{X})$; // distance matrix

$\mathcal{P}(0) \leftarrow (\mathbf{X})$; // midpoint matrix

$\mathcal{Q}(0) \leftarrow dist(\mathbf{X}, \mathcal{P}(0))$

$\mathbf{M}(0) \leftarrow gabriel(\mathbf{D}(0), \mathcal{Q}(0))$; // adjacency matrix (Eq. 3.1)

$t = 1$

while $t \neq (N - m)$ **do**

  $\mathbf{x}(t) \leftarrow \Phi$; // access new point

  $W \leftarrow \mathbf{x}(t)$; // slide the window

  $\mathbf{D}(t+1) \leftarrow update(\mathbf{D}(t))$; // (Eq. 3.2)

  $\mathcal{P}(t+1) \leftarrow update(\mathcal{P}(t))$; // (Eq. 3.4)

  $\mathcal{Q}(t+1) \leftarrow update(\mathcal{Q}(t))$; // (Eq. 3.6)

  $\mathbf{M}(t+1) \leftarrow gabriel(\mathbf{D}(t+1), \mathcal{Q}(t+1))$; // (Eq. 3.1)

  $t = t + 1$

end

Algorithm 1: Online Gabriel graph.

3.4 Independent Dominating Set Algorithm

The presented algorithm induces an Independent Dominating Set (IDS) $S \subset V$, based on a criterion established from a distance matrix $\mathbf{D}$, and an adjacency matrix $\mathbf{M}$ of a Gabriel

graph $G = (V, E)$. The method uses a metric of dominance to define the dominating nodes.

Definition 5: the mean distance $d(v)$ of a node v is the mean distance from v to its adjacent in the open neighborhood $N(v)$, as defined in Equation 3.8.

$$d(v) = \frac{\sum_{q \in N(v)} d(v, q)}{|N(v)|}. \tag{3.8}$$

The maximum mean distance $\Theta(G)$ and minimum mean distance $\theta(G)$, of a graph G, are defined, respectively,

$$\Theta(G) = max\{d(v) : v \in V\}, \tag{3.9}$$

$$\theta(G) = min\{d(v) : v \in V\}. \tag{3.10}$$

The method relies on a metric $\phi(v)$, presented in Equation 3.13, to create a predefined order of dominance, for all nodes in V. To establish an order of dominance, the approach takes into account the normalized degree $deg_n(v)$ (Equation 3.12), combined with the normalized mean distance to the open neighborhood $d_n(v)$ (Equation 3.11):

$$d_n(v) = 1 - \frac{d(v) - \theta(G)}{\Theta(G) - \theta(G)}, \tag{3.11}$$

$$deg_n(v) = \frac{deg(v) - \delta(G)}{\Delta(G) - \delta(G)}, \tag{3.12}$$

$$\phi(v) = \frac{d_n(v) + deg_n(v)}{2}, \tag{3.13}$$

where $\theta(G)$ and $\Theta(G)$ are minimum and maximum mean distances, and $\delta(G)$ and $\Delta(G)$ are minimum and maximum degrees, respectively.

The values of $deg(v)$ and $d(v)$ are, respectively, extracted from matrices $\mathbf{M}$ and $\mathbf{D}$. Considering the vectors $\mathbf{u}_{deg} = [u_1, \cdots, u_N]$ and $\mathbf{o}_d = [o_1, \cdots, o_N]$, with values $u_i = deg(v_i)$, and $o_i = d(v_i)$, respectively, Equations 3.15 and 3.16 define the elements of $\mathbf{u}_{deg}$ and $\mathbf{o}_d$.

$$\mathbf{D}_M = \mathbf{D} \cdot \mathbf{M}, \tag{3.14}$$

$$u_i = \sum_{j=1}^{N} m_{ij}, \tag{3.15}$$

$$o_i = \frac{\sum_{j=1}^{N} d_{Mij}}{u_i}, \tag{3.16}$$

where m_{ij} and $d_{M\,ij}$ are elements of matrices $\mathbf{M}$ and $\mathbf{D}_M$, respectively, with the latter formed by the dot product in Equation 3.14.

The algorithm consists of a recurring process to select a node $v_i \in V$, with the higher value of $\phi(v_i)$, as a dominating node in S, according to Equation 3.17. Nodes in $N(v_i)$ are defined as non dominating. Then, set V is updated according to Equation 3.18.

$$\arg\max_{v_i \in V} \phi(v_i) = v^* \tag{3.17}$$

$$V = V - \{v_i, N(v_i)\}. \tag{3.18}$$

If two or more nodes exist with the same value of v^*, the values of $d_n(v)$ and $deg_n(v)$ are used as second and third selection rules. The process ends when all nodes in V are assessed.

The bi-objective optimization approach with metric $\phi(v)$ reject a node v_i as dominating when $deg_n(v_i) \approx deg_n(v_j)$, but $d_n(v_i) \ll d_n(v_j)$, or when $d_n(v_i) \approx d_n(v_j)$, but $deg_n(v_i) \ll deg_n(v_j)$, for any node $v_j \in N(v_i)$. The former refrains outliers as dominating, while the latter avoids a node with lower representation as dominating. Algorithm 2 summarizes the proposed algorithm, which has as output the independent dominating set S and its complement $V - S$, that is also a dominating set, although not being an independent set. Generally the subset S tends to be smaller than $V - S$. When there are no outliers in V, lines 7-8-9 may be included to ensure that any node v, with $N(v) = 1$, is considered dominating, increasing the representativeness of S.

input : $V, \mathbf{D}, \mathbf{M}$

output: S,$V - S$

$\mathbf{D}_M = \mathbf{D} \cdot \mathbf{M}$; // (Eq. 3.14)

$\mathbf{u_{deg}} \leftarrow (\mathbf{M})$; // (Eq. 3.15)

$\mathbf{o_d} \leftarrow (\mathbf{M}, \mathbf{D}_M)$; // (Eq. 3.16)

$d_n(V) = 1 - \frac{d(V)-\theta(G)}{\Theta(G)-\theta(G)}$; // (Eq. 3.11)

$deg_n(V) = \frac{deg(V)-\delta(G)}{\Delta(G)-\delta(G)}$; // (Eq. 3.12)

$\phi(V) = \frac{d_n(V)+deg_n(V)}{2}$; // (Eq. 3.13)

Choose $\{v\} \in V, \ni N(v) = 1$

$S \leftarrow v$

$V = V - \{v, N(v)\}$

while $V \neq \emptyset$ **do**

  Choose $\{v\} \in V, \ni \phi(v) = v^*$

  if $Q = \{v\}$ *is singleton* **then**

    $S \leftarrow v$

    $V = V - \{v, N(v)\}$

  else

    Choose $\{v\} \in Q, \ni d_n(v) = v^*$

    if $P = \{v\}$ *is singleton* **then**

      $S \leftarrow v$

      $V = V - \{v, N(v)\}$

    else

      Choose $\{v\} \in P, \ni deg_n(v) = v^*$

      $S \leftarrow v$

      $V = V - \{v, N(v)\}$

    end

  end

end

$output \leftarrow S, V - S$

Algorithm 2: Independent dominating set.

The predefined order of dominance established by metric $\phi(v)$ avoids the use of any optimization method to determine S. Thus, the solution is directly obtained from the data structure, with the values of $\phi(v)$ extracted from the distance and adjacency matrices $(\mathbf{D}, \mathbf{M})$.

It is important to note that the algorithm does not require any user defined parameters, and presents no stochasticity [1].

An example of the independent dominating set S and its complement $V - S$, induced with the proposed method are presented in Figure 3.2. From the left to the right it is presented: a set of points V in blue dots, containing clusters with different densities, the Gabriel graph induced from V with the dominating and non-dominating points, in red and black dots, respectively; the points in independent dominating set S, in red; and the points in its complement set $V - S$, in black.

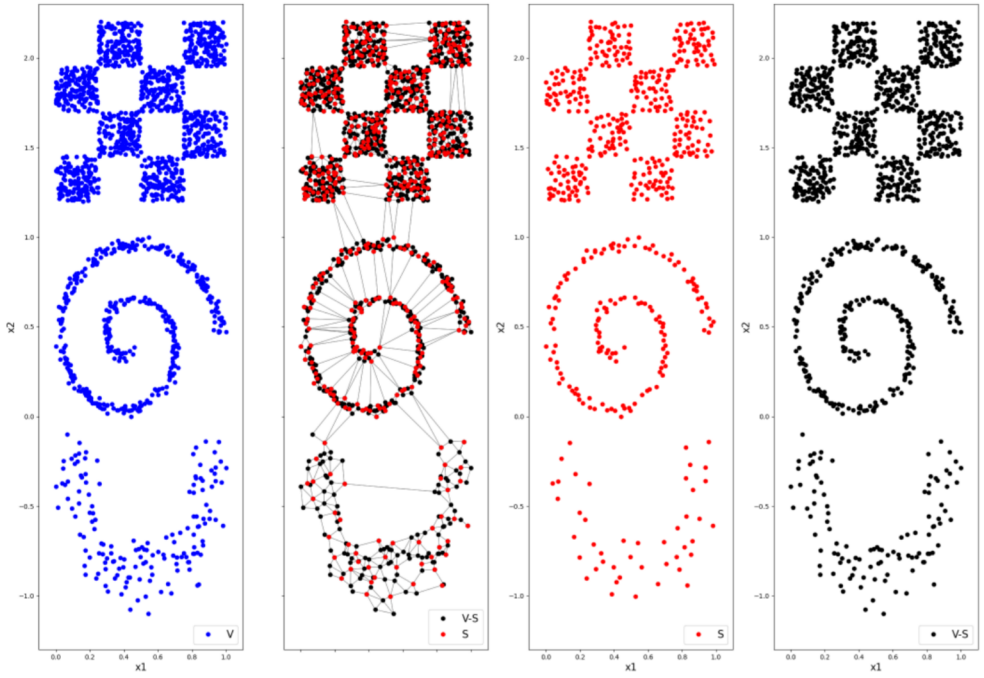


Figure 3.2: Example of a set of points V processed with the proposed Algorithm 2. From the left to the right, the graphs present: the set V, the Gabriel graph and the subsets S and $V - S$, the points in S, and the points in $V - S$.

[1] For the unusual case in which v^*, $d_n(v)$ and $deg_n(v)$ are not informative to decide between two or more nodes, their order in V is an alternative to ensure that a same solution for set V, is always obtained.

3.5 Online KDE Estimator

This section presents a proposal to use information extracted from dominating sets (Algorithm 2) of the Gabriel graph, to define the parameters of an online KDE Estimator. The approach uses the online update of the Gabriel graph (Algorithm 1), avoiding a new graph induction from scratch, every instant that one new point becomes available.

In the following, the covering of a set of points by an KDE Estimator with kernels centered on the dominating points is introduced, as well as the proposed approach to update the parameters of the KDE Estimator, on a online basis.

Let a set V with m points $\mathbf{x} \in \mathbb{R}^n$, and a robust dominating set $S \subset V$, capable of representing the probability distribution of Set V. Then, Equation 3.19 presents a general expression for a KDE estimator, using a RBF kernel K with radius h, and the points in V.

$$p_V(x) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{h} K\left(\frac{x - x_i}{h}\right) \tag{3.19}$$

Since $S \subset V$, Equation 3.19 can be decomposed into the two terms of Equation 3.20, the first one representing points in S and the second one points in the non dominating set $V - S$.

$$p_V(x) = \frac{1}{hm} \left[\underbrace{\sum_{i=1}^{g} K\left(\frac{x - x_i}{h}\right)}_{S} + \underbrace{\sum_{j=1}^{m-g} K\left(\frac{x - x_j}{h}\right)}_{V-S} \right] \tag{3.20}$$

where g is the cardinality of S.

Once the set S approximates the distribution of V, and by Definition 1, every node in the set $V - S$ is adjacent to some node in S, it is possible to induce a coverage space for points in $V - S$, using a KDE with kernels centered only over the points in S, as presented in Equation 3.21.

$$p_S(x) = \frac{1}{hg} \sum_{i=1}^{g} K\left(\frac{x - x_i}{h}\right) \tag{3.21}$$

The online approach of the method uses a sliding window $W = [\mathbf{X}, \mathbf{y}]$, with size m, containing a sequence of the most recent points $\mathbf{X}$, from a data generating function Φ, and their respective labels $\mathbf{y}$. Thus, the estimation for a new point $\mathbf{x}(t) \in \mathbb{R}^n$, is given by,

$$p(\mathbf{x}(t)) = \frac{1}{g(t-1)[h(t-1)]^n} \sum_{i=1}^{g(t-1)} K\left(\frac{\mathbf{x}_i - \mathbf{x}(t)}{h(t-1)}\right) \tag{3.22}$$

where $K(\cdot)$ represents the operation of a general kernel, $g(t-1)$ is the number of kernels, and $h(t-t)$ is the kernel radius at instant $t-1$.

Initially, a Gabriel graph $G(0) = (V(0), E(0))$, with adjacent matrix $\mathbf{M}(0)$, is induced from $\mathbf{X}$, by computing of matrices $\mathbf{D}(0)$, $\mathcal{P}(0)$, $\mathcal{Q}(0)$, and Equation 3.1. Then, an IDS $S(0)$ is induced using $V(0)$, $\mathbf{D}(0)$ and $\mathbf{M}(0)$, according to Algorithm 2. Each kernel $K(\cdot)$, in Equation 3.22, is centered in one dominating node in $S(0)$. Thus, the initial number of kernels $g(t)$ is defined according to Equation 3.23. Separation of $g(t)$ per class, is done using the labels $\mathbf{y}(t)$ in W.

$$g(t) = |S(t)|, \qquad g(t) = g_1(t) + g_2(t) + \cdots + g_C(t), \tag{3.23}$$

where C represents the total number of classes in W.

The kernel radius is defined using a distance relation, between points in the sets $S(t)$ and $V(t) - S(t)$. For each class, the mean of Euclidean distance is calculated from each non-dominating point in $V(t) - S(t)$ to its closer dominating point in $S(t)$, with same class. All these distances were previously calculated in $\mathbf{D}(t)$, and are discriminated using $S(t)$ and $\mathbf{y}(t)$. Let sets $S(t) = \{S_1(t), S_2(t), \cdots, S_C(t)\}$, and $V(t) - S(t) = \{V_1(t) - S_1(t), V_2(t) - S_2(t), \cdots, V_C(t) - S_C(t)\}$, with $S_k(t)$ and $V_k(t) - S_k(t)$ representing, respectively, subsets with all the dominating and non-dominating points of a class $k = (1, 2, \cdots, C)$. For each class k, a distance vector $\mathbf{a}_k^*(t)$, containing distances a_j^k, for $j = (1, \cdots, \rho)$, from each point in $(V_k - S_k)(t)$ to its shortest point in $S_k(t)$. Then, Equation 3.24 defines the kernel radius h_k, for each class k. Figure 3.3 presents an KDE example with the proposed method.

$$h_k(t) = \frac{\sum_{j=1}^{\rho} a_j^k}{\rho}, \qquad \rho = |V_k(t) - S_k(t)|, \tag{3.24}$$
$$j = (1, 2, \cdots, \rho), \quad k = (1, 2, \cdots, C).$$

After the density estimation with Equation 3.22, the proposed approach uses a Bayesian classifier [Duda et al., 2000] to separate the classes. For the binary case, with labels $y = \{-1, 1\}$, the decision rule is defined according to Equation 3.25.

$$\hat{y} = \begin{cases} 1, & \text{if, } p(\mathbf{x}|y=1) - zp(\mathbf{x}|y=-1) \geq 0, \\ -1, & \text{otherwise,} \end{cases} \tag{3.25}$$

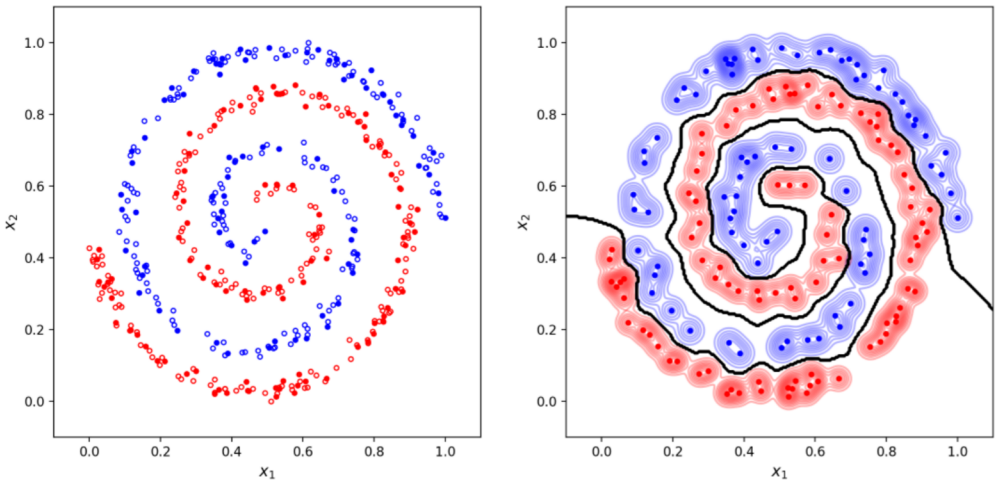


Figure 3.3: Graph on the left, presents the Spirals data set, with dominating and non dominating points, respectively, in nonempty and empty dots; graph on the right shows the KDE obtained with the proposed method and the decision boundary (line. in black).

where $z = \frac{p(y=-1)}{p(y=1)}$, is the ratio of *a priori* probabilities of the two classes, $p(y = -1)$ and $p(y = 1)$, that is inferred with the cardinality of the dominating sets $|S_1(t)|$ and $|S_2(t)|$, contained in W.

When a new tuple $[\mathbf{x}(t), y(t)]$ become available, the KDE Estimator is updated with the insertion of $[\mathbf{x}(t), y(t)]$ in W, while the older tuple is discarded. Then, matrices $\mathbf{D}(t)$, $\mathcal{P}(t)$, $\mathcal{Q}(t)$ and $\mathbf{M}(t)$ are updated according to Algorithm 1. Next, the independent dominating set $S(t)$ is defined, with Algorithm 2, enabling the updating of the kernel parameters. This process is summarized in the Algorithm 3.

The proposed method to estimate the parameters of an online KDE Estimator fits as a Passive single algorithm [Ditzler et al., 2015], as the model updating is done with each new point available, without any external support to treat concept drifts. The relation between the parameters: size of window m, number of kernels $g(t)$ and kernel width $h(t)$, explain how the proposed method aims to cover the points in W, in a self-acting approach. A large value of m will tend to result in a larger value of $g(t)$ (although in smaller number than the former), resulting in a small value of $h(t)$, and vice-versa, for points in W that represents a same distribution.

input : $\Phi = \{\{\mathbf{x}(t), y(t)\}_{t=1}^{N} | \mathbf{x} \in \mathbb{R}^n, y \in \{-1, 1\}, N \to \infty\}, m$

output: $\hat{y}(t)$

$W \leftarrow \Phi$; // initial set $W = [\mathbf{X}, \mathbf{y}]$, with m points

$\mathbf{D}(0) \leftarrow dist(\mathbf{X})$; // distance matrix

$\mathcal{P}(0) \leftarrow (\mathbf{X})$; // midpoint matrix

$\mathcal{Q}(0) \leftarrow dist(\mathbf{X}, \mathcal{P}(0))$; // distances from midpoints to points

$\mathbf{M}(0) \leftarrow gabriel(\mathbf{D}(0), \mathcal{Q}(0))$; // adjacency matrix (Eq. 3.1)

$S(0) \leftarrow Algorithm2(V(0), \mathbf{D}(0), \mathbf{M}(0))$; // dominating points

$g(0) = |S(0)|$; // (Eq. 3.23)

$\mathbf{a}_k^*(0) \leftarrow (\mathbf{D}(0), S(0), \mathbf{y}(0)), \quad k = (1, 2, \cdots, C)$

$h_k(0) = \frac{\sum_{j=1}^{\rho} a_j^k}{\rho}, \quad \rho = |V_k(0) - S_k(0)|$; // (Eq. 3.24)

$t = 1$

while $t \neq (N - m)$ **do**

  $\mathbf{x}(t) \leftarrow \Phi$; // access new point

  $p_k(\mathbf{x}(t)) = \frac{1}{g_k(t-1)[h_k(t-1)]^n} \sum_{i=1}^{g(t-1)} K\left(\frac{\mathbf{x}_i - \mathbf{x}(t)}{h_k(t-1)}\right), \quad k = \{1, 2\}$; // (Eq. 4.2)

  $z = \frac{|S_1(t)|}{|S_2(t)|}, \quad \hat{y}(t)$; // prediction (Eq. 3.25)

  $y(t) \leftarrow \Phi$; // access true label

  $W \leftarrow [\mathbf{x}(t), y(t)]$; // slide the window

  $\mathbf{D}(t+1) \leftarrow update(\mathbf{D}(t))$; // (Eq. 3.2)

  $\mathcal{P}(t+1) \leftarrow update(\mathcal{P}(t))$; // (Eq. 3.4)

  $\mathcal{Q}(t+1) \leftarrow update(\mathcal{Q}(t))$; // (Eq. 3.6)

  $\mathbf{M}(t+1) \leftarrow gabriel(\mathbf{D}(t+1), \mathcal{Q}(t+1))$; // (Eq. 3.1)

  $S(t+1) \leftarrow Algorithm2(V(t+1), \mathbf{D}(t+1), \mathbf{M}(t+1))$

  $g(t+1) = |S(t+1)|$; // (Eq. 3.23)

  $\mathbf{a}_k^*(t+1) \leftarrow (\mathbf{D}(t+1), S(t+1), \mathbf{y}(t+1))$

  $h_k(t+1) = \frac{\sum_{j=1}^{\rho} a_j^k}{\rho}, \quad \rho = |V_k(t+1) - S_k(t+1)|$; // (Eq. 3.24)

  $t = t + 1$

end

Algorithm 3: Online KDE Estimator.

3.6 Experiments

This section describes the tests that were conducted, aiming to evaluate the precision of the proposed KDE estimator, when performing the classification of non-stationary data streams, and to assess the time complexity of the proposed approach to online update of a Gabriel graph

and the inducement of an independent dominating set, using the Algorithm 2.

3.6.1 Comparative Tests

This section presents comparative tests with the proposed online KDE estimator, with nine synthetic data sets. Some of these data sets are often used in the literature as non-stationary classification problems. The proposed approach was compared with the Hoeffding tree model, with ADWIN, and with the ensemble classifier Dynamic Weighted Majority, when using Naive Bayes as base classifiers.

The data sets: *Spirals* [Ditzler and Polikar, 2011; Yu and Webb, 2019], consisting of two spirals of opposite classes; *Checkerboard* [Elwell and Polikar, 2011; Ditzler and Polikar, 2011, 2012; Yu and Webb, 2019], which is a generalization of the nonlinear XOR problem; *Gaussians* [Ditzler and Polikar, 2011], consisting of two slightly overlapped Gaussian distributions; and *Two moons*; were modified with the introduction of two types of concept drifts: abrupt and incremental. The data sets with abrupt concept drifts consist of 10.000 points and 3 drifts that are the result of angular rotation of the data, in the points 2.500, 5.000, and 7.500 of the data stream. The data sets *Spirals*, *Checkerboard* and *Gaussians* were rotated with 90°, while *Two moons* was modified, applying a rotation of 115°, in each concept drift. The incremental case consist of data sets with 5.000 points and a concept drift that occurs continuously with the total angular rotation of 180°, 315°and 90°, in the data sets *Spirals*, *Two moons* and *Checkerboard*, respectively. The incremental concept drift was introduce in the *Gaussians* data set as moving one of the Gaussians centers towards the other center, following by a movement in the opposite direction. These 8 data sets have 2 dimensions and 2 classes. The abrupt case is presented in Figure 3.4, while incremental data sets are illustrated in Figure 3.5. A ninth data set, *Eight gaussians*, was also tested, consisting of 20.000 points, with 4 dimensions, 4 classes and eight moving Gaussians (two per each class), with a same variance in each dimension. The *Eight gaussians* data set presents abrupt and incremental drifts, and it is shown in Figure 3.6, with its 4 variables plotted pairwise in 6 graphs. All variables of the 9 data sets were normalized in the interval $[0, 1]$.

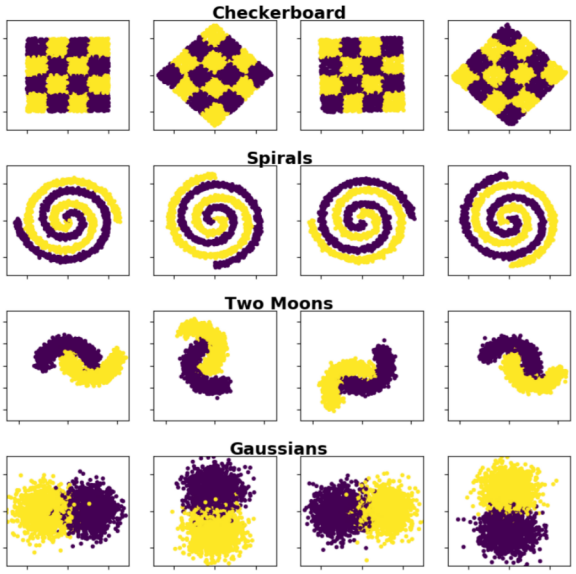


Figure 3.4: Data sets with 3 abrupt drifts: *Checkerboard*, *Spirals*, *Two moons*, and *Gaussians* (from top to bottom); presenting 4 different instants (from left to right).

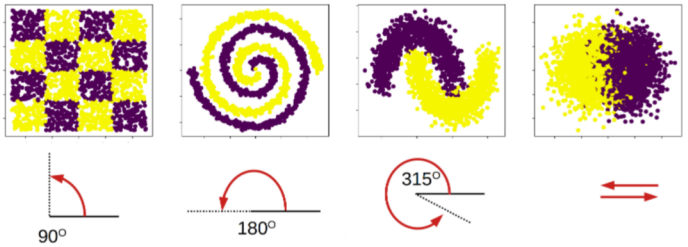


Figure 3.5: Data sets with incremental drifts: *Checkerboard*, *Spirals*, *Two moons*, and *Gaussians* (from left to right).

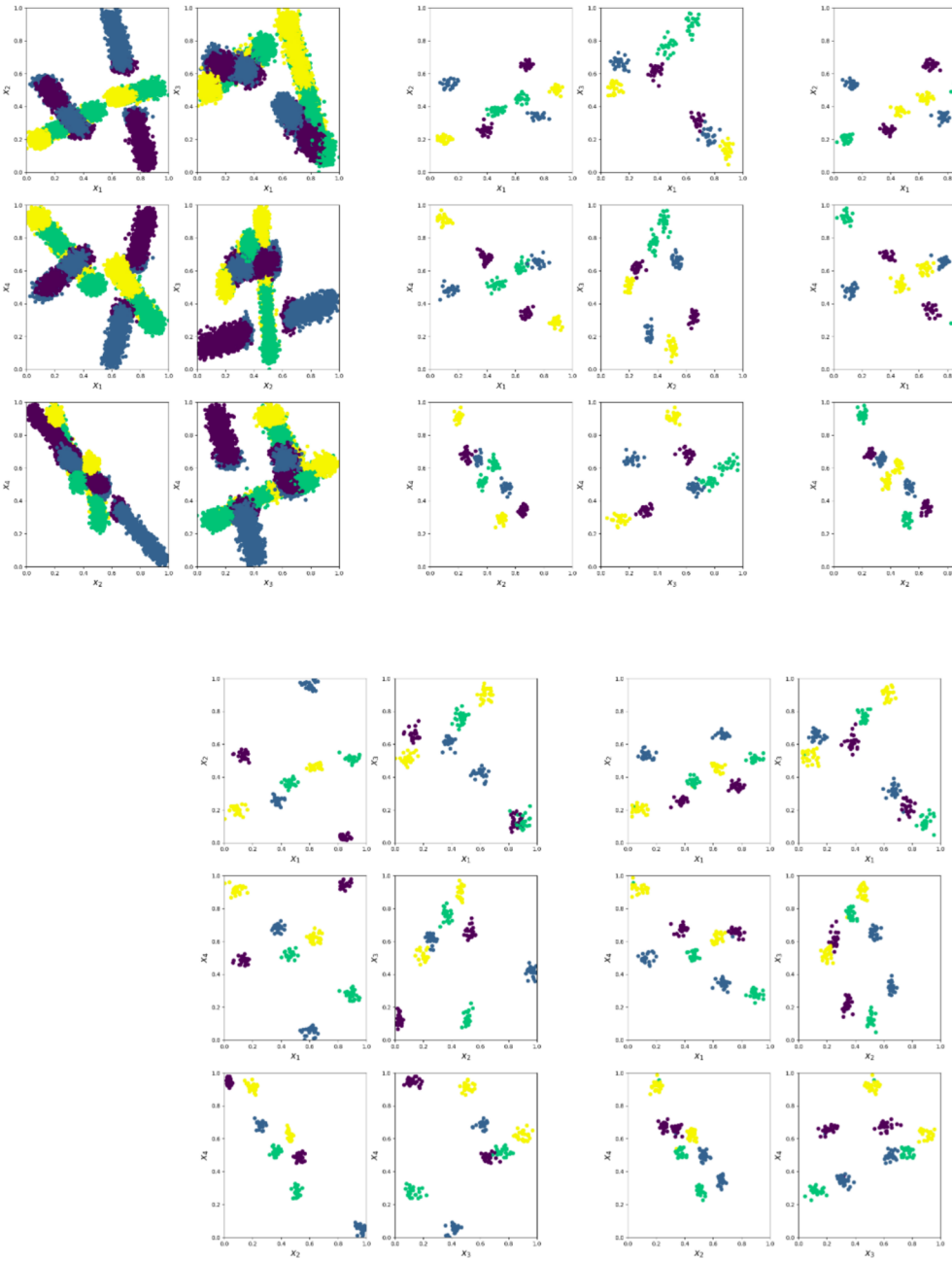


Figure 3.6: *Eight gaussians* data set, with its 4 classes and 4 variables plotted pairwise in a set of 6 graphs. Clockwise from the top left, the first set presents all points of the data set, then each set shows a different instant of this last.

The proposed method was tested with 4 different sizes of the sliding window $m = (200, 300, 400, 500)$, using the Gaussian kernel, $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2h^2}}$. The complement of the independent dominating set, founded with the Algorithm 2, was used as dominating set. For the DWM and Hoeffding tree models, a previous grid search test was conducted, using 1.000 points of each data set. The hyperparameters tested for the Hoeffding tree model were: *split*

criterion, (Gini, Information Gain and Hellinger); *grace period*, (25, 50, 75, 100, 125, 150, 175, 200, 225, 250); *max depth*, (4, 8, 10, 12, 14, 16, 20, 22, 24, 26); *split confidence*, ($1e^{-7}$, $1e^{-6}$,$1e^{-5}$, $1e^{-4}$,$1e^{-3}$, $1e^{-2}$,$1e^{-1}$); and *tie threshold*, ($5e^{-4}$, $1e^{-4}$,$5e^{-3}$, $1e^{-3}$,$5e^{-2}$, $1e^{-2}$,$5e^{-1}$). The hyperparameters tested for the DWM model were: *number of estimators*, $(5, 10, 15, 20, 25)$; *period between removal*, $(30, 40, 50, 70)$; *factor for decreasing weights*, $(0.3, 0.4, 0.5, 0.6, 0.7)$; and *threshold for deleting*, $(0.005, 0.01, 0.05, 0.1, 0.5)$. All the methods learned the data sets point-by-point.

The test results with the moving average error for each classifier is presented in Figure 3.7. The moving average error for the 4 data sets with abrupt concept drifts and the *Eight gaussians* revealed that the proposed method presented the faster recovery after each concept drift. Probably, the mechanism of the proposed approach, that determines the position of a new kernel when a new dominating point appears, enables the method to quickly cover the space where the data is changing. The results with the data sets containing incremental concept drifts also evidences this characteristic, with the proposed approach presenting low error. The test results are summarized in Table 3.1, showing the metric Area under the curve ROC (AUC) as a overall performance. The time required in each test is presented in Table 3.2, with the respective time of the grid test. The results show a higher prediction accuracy for the proposed approach, than the DWM and Hoeffding tree models. The assessment of requite of time to perform the classification shows higher values for the proposed approach.

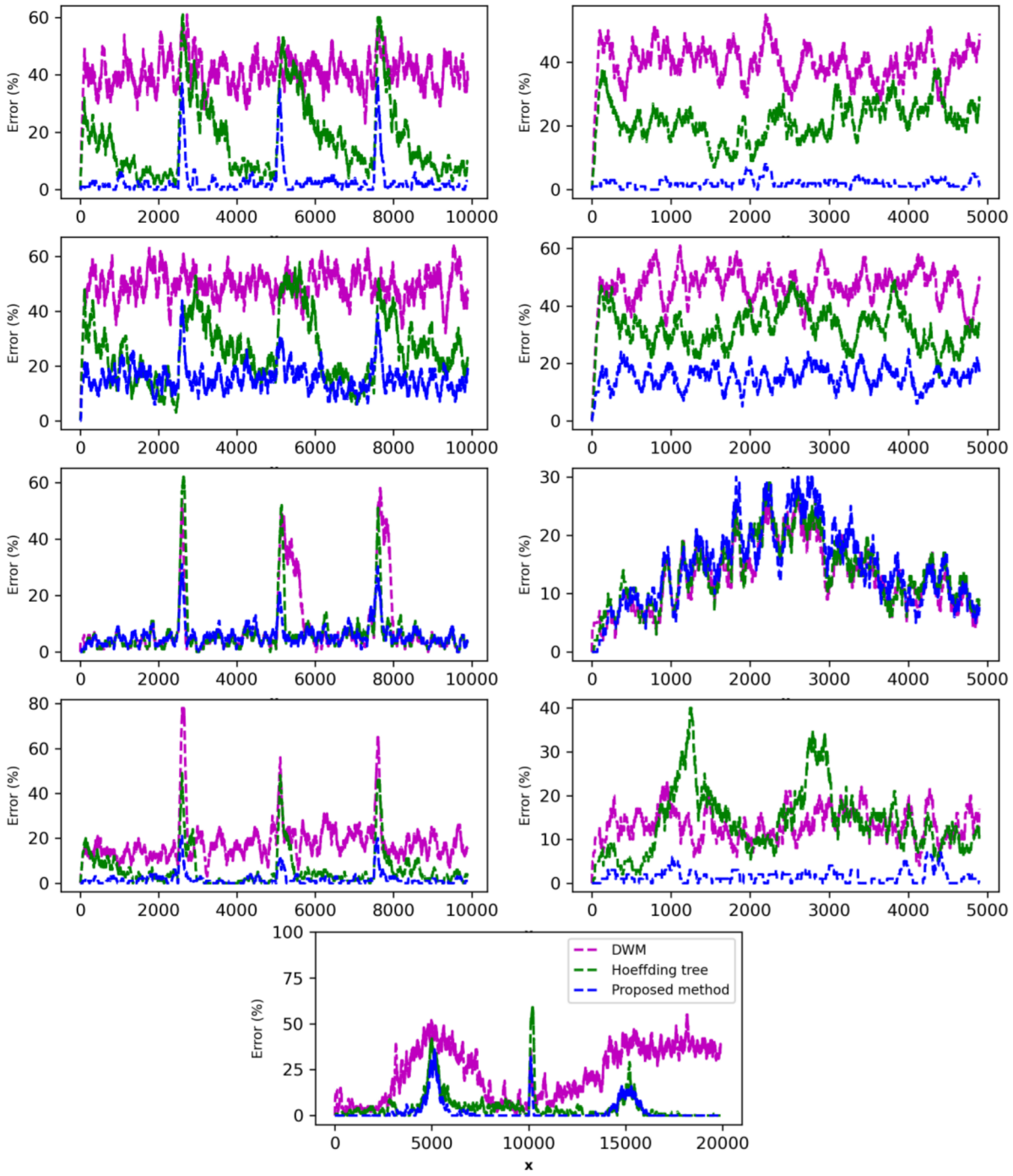


Figure 3.7: Moving average error (%) for the classifiers tested. Graphs on the left show data sets with abrupt drifts. Graphs on the right present the incremental case. From the top to bottom, the data sets are: *Spirals*, *Checkerboard*, *Gaussians* and *Two moons*. The graph on the bottom center shows the results for the *Eight gaussians*.

Table 3.1: AUC for the methods tested with the 9 data sets.

Data sets	n	N	Type of drift	Proposed approach				Hoeffding tree	DWM
				$\mathbf{m} = 200$	$\mathbf{m} = 300$	$\mathbf{m} = 400$	$\mathbf{m} = 500$		
Spirals	2	10.000	Abrupt	0.9703	0.9789	0.9790	0.9771	0.8423 ± 8e-3	0.5888
		5.000	Incre.	0.9798	0.9924	0.9976	0.9986	0.7978 ± 2e-2	0.5986
Checkerboard	2	10.000	Abrupt	0.8431	0.8598	0.8759	0.8810	0.7688 ± 2e-2	0.5015
		5.000	Incre.	0.8452	0.8640	0.8610	0.8656	0.7040 ± 2e-2	0.5242
Gaussians	2	10.000	Abrupt	0.9415	0.9415	0.9384	0.9336	0.9367 ± 2e-3	0.9174
		5.000	Incre.	0.8548	0.8636	0.8676	0.8612	0.8626 ± 3e-3	0.8725
Two moons	2	10.000	Abrupt	0.9818	0.9802	0.9783	0.9780	0.9427 ± 3e-3	0.8162
		5.000	Incre.	0.9836	0.9822	0.9802	0.9756	0.8559 ± 1e-2	0.8654
Eight gaussians	4	20.000	Abrupt & Incre.	0.9992	0.9992	0.9988	0.9988	0.9907 ± 5e-4	0.8421

Table 3.2: Time(s) required in the classification and grid test.

Data sets	Type of drift	Proposed approach	Grid Hoeffding tree	Hoeffding tree	Grid DWM	DWM
Spirals	Abrupt	933.00	1814.16	3.07 ± 6e-2	275.09	8.49
	Incre.	430.56	1,847.12	1.70 ± 5e-2	274.12	2.29
Checkerboard	Abrupt	814.58	1,654.41	3.27 ± 1e-1	226.84	2.59
	Incre.	522.72	1,631.45	1.81 ± 5e-2	228.33	1.28
Gaussians	Abrupt	765.35	1,979.61	1.77 ± 4e-2	91.19	3.45
	Incre.	315.53	2,055.04	1.01 ± 1e-2	91.2	1.61
Two moons	Abrupt	941.58	2,106.45	2.48 ± 2e-1	169.98	4.35
	Incre.	531.90	2,355.15	1.08 ± 4e-2	189.89	2.08
Eight gaussians	Abrupt & Incre.	2,009.67	2,649.72	6.12 ± 2e-1	121.88	13.43

3.6.2 Time Complexity Tests

A test to assess the time complexity to induce a Gabriel graph using the proposed approach, with (Equation 3.1), and an IDS, using Algorithm 2, was accomplished. The updating step of a Gabriel graph with one new point, including the updating of matrices $\mathbf{D}$, $\mathcal{P}$ and $\mathcal{Q}$, was also considering. The Gabriel graph induced with Equation 2.1 and implemented with a nested loop (one inside another) was used as a reference baseline. This last has chosen because of its simple implementation. These two inductive approaches of a Gabriel graph were tested with a same set of points, resulting in a same graph. The test consisted in measure the time of execution of each method. Synthetic data sets were generated containing Gaussian distributions with centers located randomly and a same variance in each of the dimensions. Different values for dimension n, (2, 3, 4, 5, 6, 7, 8, 9,10), and for number of points m, (100, 200, 300, 400, 500, 600, 700) were tested. The number of Gaussian distributions was set equal to the value of n, on each test.

The test with the online updating approach of the Gabriel graph, was performed using each Gabriel graph that was induced with the two former approaches. The test consisted in measure the time of execution of the updating of matrices $\mathbf{D}$, $\mathcal{P}$ and $\mathcal{Q}$, and Equation 3.1, when a new point is added to the graph and some point is removed. The test results are presented in Table 3.3, showing the mean time (s) of execution for the reference approach, the proposed approach to induce the Gabriel graph and the proposed approach to update the Gabriel graph with a new point. Each value in Table 3.3 is the mean value of 10 repetitions that were considered. The standard deviations, not presented in the Table, were smaller than: $1e^{-0}$ and $4e^{-1}$ for the reference and proposed approaches to induce the Gabriel graph, and $9e^{-2}$ for the proposed approach of updating the graph. For a better perception of tendency, the values of the Table 3.3 are plotted in the Figure 3.8.

Table 3.3: Time complexity for the reference and the proposed approach to induce a Gabriel graph, for different values of number of points (m) and dimension (n). The results for the online update the Gabriel graph is also presented.

	m	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
Reference	**100**	0.2855	0.2907	0.2903	0.2990	0.2959	0.3000	0.3035	0.3005	0.3054
	200	1.2671	1.3016	1.3211	1.3450	1.3676	1.3942	1.3957	1.4683	1.4426
	300	3.4442	3.2981	3.3529	3.6779	3.9041	4.1140	4.0643	3.9150	4.0895
	400	6.3739	6.8962	6.9162	7.0186	7.4357	7.3499	7.0493	7.1894	7.3090
	500	10.5798	11.3367	11.3254	12.3146	12.9124	13.6049	12.6336	13.7289	14.1522
	600	18.0769	16.3187	15.7423	16.4119	16.8805	17.3914	17.1808	17.8978	18.4782
	700	21.2069	22.1323	22.8044	23.8941	24.7580	25.6137	25.2177	26.4660	27.3498
Proposed approach	**100**	0.0271	0.0265	0.0265	0.0292	0.0293	0.0320	0.0303	0.0331	0.0333
	200	0.1726	0.1832	0.1902	0.2117	0.2190	0.2386	0.2403	0.2652	0.2907
	300	0.7230	0.7580	0.7968	0.9829	1.1413	1.3017	1.2170	1.2584	1.3085
	400	2.0377	2.2889	2.3704	2.4734	2.7658	2.8869	2.6443	2.8215	3.0304
	500	3.8411	4.3138	4.4102	4.9734	5.5162	5.6954	5.3693	6.1475	6.3976
	600	6.9563	6.8091	7.0932	7.5452	8.0003	8.4590	8.5280	9.1112	9.5812
	700	9.8503	10.6235	11.2787	11.9832	12.6989	13.4486	13.5808	14.4132	15.1570
Online updating	**100**	0.0035	0.0034	0.0034	0.0036	0.0036	0.0035	0.0035	0.0035	0.0037
	200	0.0232	0.0237	0.0237	0.0239	0.0242	0.0245	0.0244	0.0252	0.0252
	300	0.1107	0.1123	0.1091	0.1209	0.1253	0.1332	0.1286	0.1247	0.1251
	400	0.2787	0.3001	0.2783	0.2715	0.2914	0.2906	0.2703	0.2734	0.2750
	500	0.5643	0.5365	0.5279	0.5801	0.5893	0.5926	0.5518	0.6062	0.5692
	600	1.0363	0.9076	0.8968	0.9000	0.9002	0.9033	0.9027	0.9062	0.9076
	700	1.6172	1.6248	1.6298	1.6368	1.6359	1.6394	1.6417	1.6393	1.6442

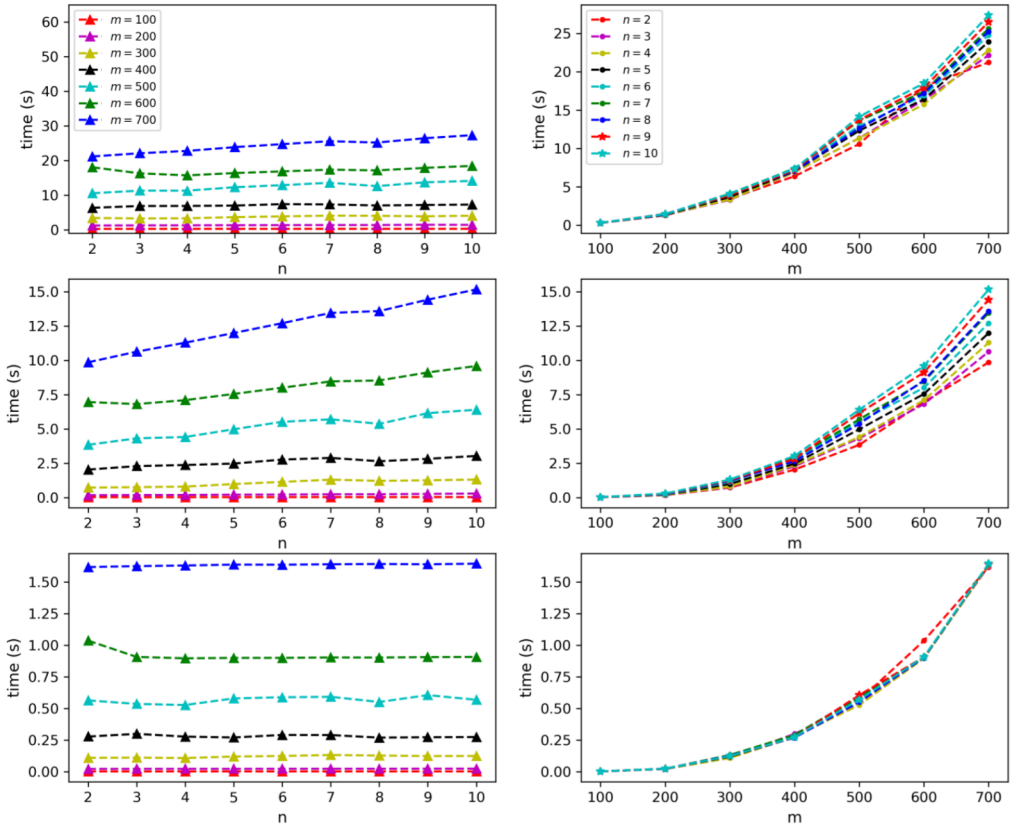


Figure 3.8: Time complexity for the reference and proposed approach to induce a Gabriel graph, and the proposed approach to update the Gabriel graph, from the top to the bottom, respectively. Graphs on the left present time versus dimension (n), for different values of number of points (m). Graphs on the right show time versus m, for different values of n.

The results of the test comparing the reference and the proposed approach to induce a Gabriel graph, as well as, of the proposed approach to update the graph with one new point, show a similar tendency for the relations of time complexity per dimension and time complexity per number of points considered to induce the graph. The increasing of points considered to induce the graph (m) resulted in more time to induce the Gabriel graph, while the increase in the dimension (n) tends to consume a tiny amount of extra time, for the values of n tested. When comparing the two inductive approaches each other, all values of the proposed approach were smaller than the reference. The approach to update the Gabriel graph was in average 11x faster

than the proposed approach to induce an initial graph.

Subsequently, a second test was conducted aiming the evaluation of the time complexity of the Algorithm 2. In this test, independent dominating sets were induced with the Algorithm 2, from each Gabriel graph that was induced in the previously test, with the proposed approach. The mean time of execution for the Algorithm 2 are presented in Table 3.4. The values are the mean value of 10 repetitions, in which the standard deviations were smaller than $2e^{-3}$ for the time evaluated. The mean values of time from the Table 3.4 are plotted in the Figure 3.9.

Table 3.4: Time complexity for the Algorithm 2 and the respective value of $\Delta(G)$ and $|S|$, for different values of m and n. The Gabriel graphs used in the tests were the ones induced on the tests of Table 3.3.

	m	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
tine (s)	**100**	0.0013	0.0012	0.0011	0.0011	0.0010	0.0010	0.0009	0.0010	0.0010
	200	0.0023	0.0019	0.0018	0.0017	0.0016	0.0015	0.0014	0.0014	0.0014
	300	0.0036	0.0035	0.0026	0.0027	0.0027	0.0026	0.0027	0.0022	0.0021
	400	0.0055	0.0051	0.0038	0.0033	0.0036	0.0032	0.0027	0.0025	0.0025
	500	0.0075	0.0054	0.0049	0.0046	0.0044	0.0042	0.0038	0.0040	0.0038
	600	0.0101	0.0062	0.0055	0.0049	0.0045	0.0042	0.0039	0.0037	0.0036
	700	0.0095	0.0076	0.0064	0.0059	0.0054	0.0050	0.0048	0.0045	0.0042

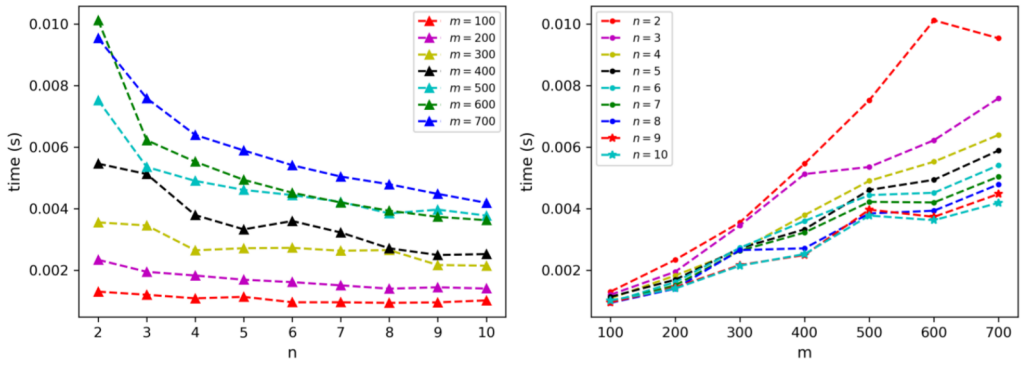


Figure 3.9: Time complexity for the Algorithm 2. Graph on the left present time versus dimension (n), for different values of number of points (m). Graph on the right show time versus m, for different values n.

The results of time complexity for the Algorithm 2, presented in Table 3.4 and Figure 3.9, show that the mean time required in the computation enlarges with the increasing of the number of points (m), while an inverse relation is observed for the number of dimensions (n),

with a decrease of the mean time when n increases. The former relation ($m \times time$) is possible explained by an greater amount of computation operations required, while the latter ($n \times time$) possible required a lesser amount, once the number of nodes connection possible increased with n and the Algorithm 2 will tend to require less computation steps to process the data.

3.7 Conclusion

This chapter presented contributions to the online updating of the Gabriel graph, a method to define an independent dominating set and an approach to define the parameters of an KDE Estimator, which is used to classify non-stationary data streams. The online updating proposal of the Gabriel graph adopts a strategy that preserves information, saving it in three distinct matrices, and compares the distance between any two points, with distances from other points to the midpoint formed with the two former points. This approach avoids a new calculation from scratch. Moreover, the comparative tests with the proposed method to induce an initial Gabriel graph and the reference approach, showed a reduction of time needed for the computation of the former, while the proposed updating approach of the Gabriel graph required a time interval considerable smaller than the graph induction from scratch.

It was also described an algorithm to induce an independent dominating set, using as input: a set of points and its respective distance matrix, along with an adjacent matrix representing the nodes connection of a correspondent graph. Based on these information, which represents the data structure, a metric of dominance is derived to ordering the recurring process in which the nodes are defined as dominating. Thus, the method does not require the setting of hyperparameters. This algorithm was applied jointly with the updating proposal of the Gabriel graph, in a new method to define the parameters of an KDE estimator, with the objective to classify non-stationary data streams. The approach does not requires the setting of hyperparameters and the use of optimization methods. Thus, no grid search is necessary, in the beginning and/or when the data changes significantly. In the Online Learning Paradigm, where the data generating function may change at any moment, such features are important characteristics. Furthermore, the comparative tests revealed that the KDE estimator was capable to adjust to the occurrence of abrupt and incremental drifts, probably due to the readily parameters definition with the dominating set.

Chapter 4

One-class RBF Neural Network and Representative Subsets

4.1 Introduction

In general, problems of fault detection, health diagnosis, banking and credit analysis, and others that are misrepresented by one of the classes, that usually is labeled as positive, generally have points from the negative class in enough number to provide an acceptable representation of its distribution. However, the positive class may be rare or absent, thus not sufficient to establish a boundary for the class. In this type of problem the boundary that separates the positive from the negative class may be induced with an one-class model, using only points from one of the classes. In this approach the learning is attained in the known class to established the separable boundary.

From the perspective of Neural Networks, the approach of using a dominating set to define an appropriate number of kernels, positioning their centers and defining a kernel radius, as described in Chapter 3, can be applied to the training of RBF networks, as proposed in [Queiroz et al., 2022], with a binary classifier. This Chapter presents a training procedure for one-class RBF neural networks, when the objective is to induce a function capable to define a decision boundary for a set of points that represents one class. The proposed approach relies on the information provided from the data structure, using a dominating set of the Gabriel graph, that are induced from the set of points representing the class to be learned, aiming to define all parameters of the RBF network. The number of hidden nodes of the network and the center of

each Radial function are defined according to the number of dominating points founded in the dominating set, and setting the center of each Radial function on one of the dominating points. The radius adopted in all Radial function is determine as the mean value of the distances from each non-dominating point to its close dominating point. Assuming that the radius found is capable to reject disturbances between near Radial functions, the output layer of the network consist of only one node, which computes a linear combination of the hidden layer outputs with an unweighted sum. Thus, no parameters is used in the output layer of the network, while all parameters from the hidden layer is defined according to the data structured.

This Chapter also presents an Algorithm to extract representative subsets from an set of multidimensional points in the real domain. The proposed method may be configured with a different number of outputs, which results in distinct levels of extraction, producing a nested subsets of selected points. The method uses the independent dominating set (Algorithm 2), presented in Chapter 3, and the Gabriel graph to extract the most representative points from subsets of the original data set, which is divided in a equally number of points and processed independently from each other.

The next section presents some related work to the proposed method. In the following, the proposed training procedure for an one-class RBF network is described. Next, a section presents the extractor algorithm of representative subsets, followed by a section of experiments with comparative tests and a test with a data set from a real industrial process.

4.2 Related works

An approach of one-class model that relies on the density estimation with one Gaussian function in a new space domain for the input data is presented in [Parra et al., 1996]. The method uses a neural network with one hidden layer to obtain the input data transformation, aiming preservation of area and volume. Assuming statistical independence, the data in the new domain can be constrained by a Gaussian distribution, with minimization of the sum of variances.

An investigative study between SVM and the Gabriel graph appears in [Zhang and King, 2002], highlighting the relationship of both methods, when problems demanding the construction of a separating boundary for the discrimination of two different classes are solved. An

one-class classifier based on SVM (OCSVM) was proposed in [Schölkopf et al., 2001]. This kernel approach maps the data into a feature space and defines a hyperplane, aiming at the separation of the data points from the origin with maximum margin. The method relies in the support vectors to construct the hyperplane, solving a quadratic program problem, in which a hyperparameter ν [0,1) is used aiming to control the number of outliers, thus enabling a solution with a soft margin. The method does not assumed any particular type of kernel, with possible requires the setting of its own hyperparameter. A similar one-class model that also uses the support vectors to establishes a boundary for a data set, is the method known as support vector domain description (SVDD) [Tax and Duin, 2004]. Different from the OCSVM model, this approach creates spherically shaped decision boundary minimizing a decision function constrained radially. The problem formulation considers possibles outliers with the setting of a hyperparameter, that trade-offs between the volume of the boundary space and the errors, using a hyperparameter.

4.3 One-class RBF Neural Network

The use of Radial functions enables that a single hidden layer network deals with disjoint regions in the decision space that pertaining the class learned [Broomhead and Lowe, 1988]. For other side, the use of a dominating set in the parameters definition of an RBF network allows a well distribute number of Radial functions, in the domain of the reference points, while a radius defined from the distance relation between non-dominating points and the close dominating point provides a mapping function capable to cover all points and to established a decision boundary. This section describes the proposed approach to define the parameters of an one-class RBF neural network, using the data structure information, obtained with a dominating set of the Gabriel graph.

Consider the following structure for the one-class RBF network, as showed in Figure 4.1, and its output defined in Equation 4.1, for a input $\mathbf{x}$.

$$f_{RBF}(\mathbf{x}) = \sum_{i=1}^{p} r_i(\mathbf{x}, \mathbf{c}, h) \tag{4.1}$$

where p is the number of hidden neurons, $r_i(\mathbf{x}, \mathbf{c}, h)$ represents a Radial function of node i, with center $\mathbf{c}$ and radius h.

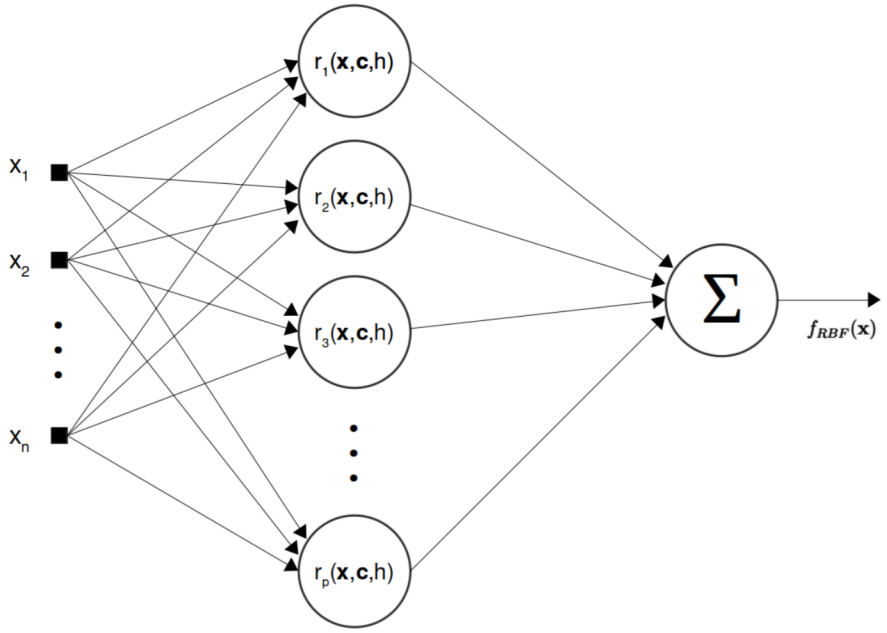


Figure 4.1: Structure of the proposed one-class RBF network.

Consider a set V with m points $\mathbf{x} \in \mathbb{R}^n$, and a robust dominating set $D \subset V$, capable of representing the probability distribution of Set V. Let the number of hidden neurons p be equal to $|D|$, centering each Radial function in one of the dominating points in D. Then, the output of each hidden neuron i, for an input $\mathbf{x}$, in given by the Radial function, i.e. the Gaussian function defined in Equations 4.2. The radius h is defined as the mean Euclidean distance from each non-dominating point in $V - D$, to its closer dominating point in D, according to Equation 4.3.

$$r_i(\mathbf{x}, \mathbf{c}, h) = e^{\frac{-(\|\mathbf{x}-\mathbf{c}\|)^2}{2h^2}}, \tag{4.2}$$

$$h = \frac{\sum_{j=1}^{\rho} a_j}{\rho}, \qquad \rho = |V - D|, \quad j = (1, 2, \cdots, \rho), \tag{4.3}$$

where a_j is the distance from each non-dominating point $\mathbf{x}_j$ in $V - D$ to its close point in D.

Once the parameters of the network is defined, the limit θ_{RBF} is set, evaluating all non-dominating points with Equation 4.1. Then, the decision boundary for the points, is directly defined by a limit value of $\theta_{RBF} = f_{RBF}(\mathbf{x}_j)$, for one point $\mathbf{x}_j$ of the non-dominating points in $V - D$. For a chosen value of θ_{RBF}, it is guarantee a border inside which a predefined number of points in $V - S$ are located. Afterwards, a decision function to label a new point $\mathbf{x}$ as located outside, $f(\mathbf{x}) = 1$, or inside, $f(\mathbf{x}) = -1$, the decision boundary is established with a decision

function $f(\mathbf{x})$, presented in Equation 4.4

$$f(\mathbf{x}) = \begin{cases} 1, & \text{if } f_{RBF}(\mathbf{x}) \leq \theta_{RBF}, \\ -1, & \text{otherwise.} \end{cases} \tag{4.4}$$

Once that the general definitions for the one-class RBF network were introduced, the particular case using the Algorithm 2 and the Gabriel graph is now explained. The network is induced in 3 steps:

1. Consider a set of points $X = \{\{\mathbf{x}\}_{i=1}^{N} | \mathbf{x} \in \mathbb{R}^n\}$, representing one class from a particular problem. First, a matrix $\mathbf{D}$, with dimensions $N \times N$, containing Euclidean distances $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$, $\forall(i,j)$, is computed and an adjacent matrix $\mathbf{M}$, of the Gabriel graph $G = (V, E)$, is calculated, according to Equation 2.1.

2. A dominating set is induced from the Gabriel graph $G = (V, E)$ established, using the Algorithm 2. Both the dominating set S or its complement $V - S$, founded with the Algorithm 2, can the used as a dominating set D. The last tends to contain a larger number of points than the former, nevertheless both with representative characteristic.

3. The parameters of the RBF network (Radial function, centers $\mathbf{c}_i$ and dispersion factor h) are defined with the graph dominance approach, centering each Radial function on one dominating point and using Equation 4.3 to define the radius. Afterwards the limit θ_{RBF} is defined, choosing a value from the ordered vector $\mathbf{p}$, which contains the values of $f_{RBF}(\mathbf{x})$, for each $\mathbf{x}$ in $V - D$.

One important characteristic of the radius that is applied over all Radial functions, is the stable value that tends to be defined with the graph dominance approach. This definition for h avoids possible large values, when i.e. defined by local rules, that possibly could impact the shape of the decision boundary. This feature is illustrated with an example using the two moons data set, in which one of the classes was changed, in such a way that the mean distance between points from a same class were different, between classes. This data set is named here as *Two moons resized*. Figure 4.2 presents the points of the data set (in black), the dominating and non dominating points of the Gabriel graph (in red and black, respectively), and the dominating points (in red), in the top graphs from the left to the right; and in the bottom graphs from the left to the right, the radii (circular disks in grey) positioned over each dominating point, the output of the RBF network as a estimator (lines in red), and the decision boundary (line in grey), . This

example shows non evidence of disturbance over the decision boundary that could be the result of a inadequate value of h.

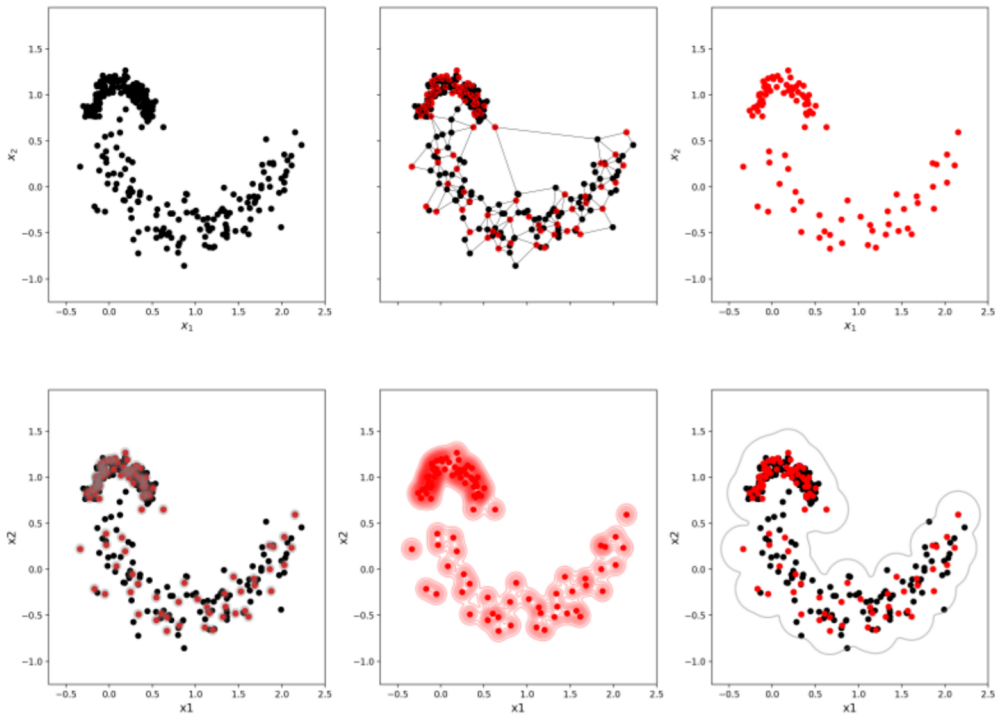


Figure 4.2: Example of the stable value of h, when defined with the graph dominance approach.

Another characteristic of the proposed method, that it is worth to mention, is how the number of points presented in the data set, from which it is intended to induce the decision boundary, impacts this last. For two data sets that although representing a same distribution, contain a different number of points, a tighter boundary space towards the training points will tend to be produced for the one with the larger number of points. Since its value of h will tend to be smaller. An example that illustrates this tendency is presented in Figure 4.3, showing a same distribution with two data sets containing 2.000 and 5.000 points (graphs on the left and on the center, respectively). The proposed approach was applied, producing the respective decision boundaries, for each of them. It is also shown the two decision boundaries overlapped in the graph on the right. It is noted that the decision boundary induced from with the set of 5.000 points (green) has a tighter shape.

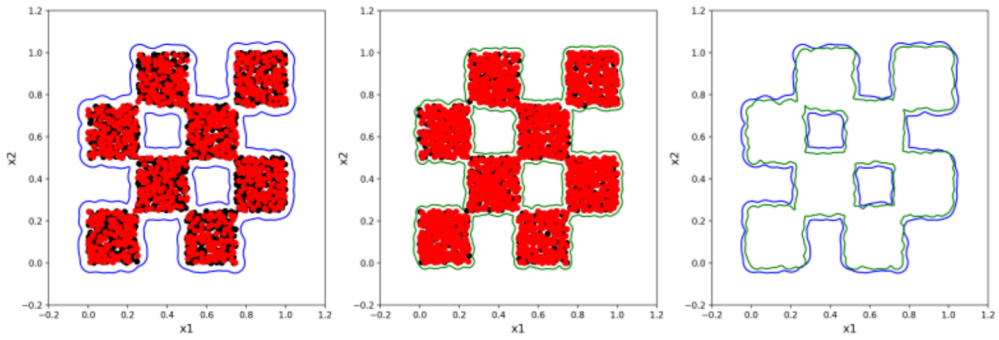


Figure 4.3: Influence of the number of points over the decision boundary. The graphs on the left and center present the dominating and non-dominating points in red and black dots, respectively, and the decision boundary induced with data sets containing 2.000 and 5.000 points (lines in blue and green),respectively. The graph on the right shows both decision boundaries overlapped.

4.4 Extractor Algorithm of Representative Subsets

Some problems in which the one-class model approach is apply often present data sets with a large number of points. This brings some difficulty when inducing models from these big data sets: (i) data may contain a overabundance of points with the same value, or almost the same value, (ii) the amount of time required in the training of a one-class model with this type of data set is a issue, specially for deep models, which naturally consume a long time to estimate a vast number of parameters, (iii) data may change somehow, due to concept drifts, thus possibly requiring a re-training of the model that was previous induced. This section presents an algorithm that extracts representative sets from a data set, containing smaller number of points. The method has different levels of extraction, in which subsets of elements are defined. The graph dominance approach is used to identify representative points, evaluating subsets of the original set independently. These sampled sets exhibit a monotonic decrease numbers of points, nevertheless with representative property that approximate the original data set.

Consider a set $X = \{\{\mathbf{x_i}\}_{i=1}^{N} | \mathbf{x} \in \mathbb{R}^n\}$, with N points from a data set. A robust dominating set $S \subset X$, capable of representing the probability distribution of Set X . In the first extraction level, the set X is divided in equally l_1 parts, as matrices $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_{l_1}$, with dimensions $(N/l_1) \times n$, each one. From each matrix $\mathbf{X}_i$, an IDS S_{1i} is induced, using Algorithm

2. As a result, this first extraction level produces the set $S_1 = \{S_{11}, S_{12}, \cdots, S_{1l_1}\}$, with each S_{1i} possibly having different number of points. The objective is to keep inducing new IDS's from S_1, aiming to reduce the number of representation points. Thus, in a second extraction level, the set S_1 is equally divided in l_2 subsets $S_1 = \{S'_{11}, S'_{12}, \cdots, S'_{1l_2}\}$. From each S'_{1j} an IDS S_{2j} is induced, resulting in the set $S_2 = \{S_{21}, S_{22}, \cdots, S_{2l_2}\}$. Thus, this process generated nested subsets $S_k \subset S_{k-1} \subset \cdots \subset S_2 \subset S_1 \subset X$, where k' is the number of extraction levels, that is predefined.

The Algorithm 4 presents the proposed procedure to extract a representative set from data set X. Two extraction levels are considered in this case. The Algorithm is implemented as a reoccurring process, calling Algorithm 2 and saving the dominating points of each extraction level. For a particular order of points in a data set, no stochastic is presented in the method. Examples of the proposed algorithm, using 2 different data sets and 3 extraction levels are presented in the Figure 4.4

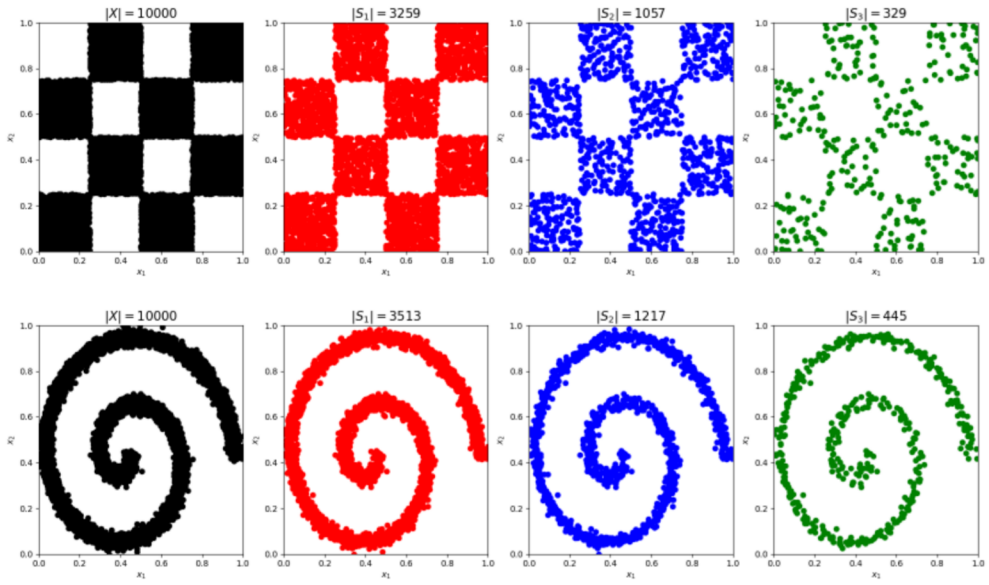


Figure 4.4: Examples of sets extracted with the proposed algorithm, for 2 different data sets and 3 extraction levels. Number of points in each set is presented above each graph.

input : $X = \{\{\mathbf{x}_i\}_{i=1}^{N} | \mathbf{x} \in \mathbb{R}^n\}$,$l_1$,$l_2$

output: S_2

$S_1, S_2, S_{aux} \leftarrow \emptyset$

$i = 1$

while $i < N$ **do**

  $\mathbf{X}_i \leftarrow \Phi_N$; // Access new points

  $\mathbf{D}_i \leftarrow dist(\mathbf{X}_i)$; // distance matrix

  $\mathbf{M}_i \leftarrow gabriel(\mathbf{X}_i, \mathbf{D}_i)$; // adjacency matrix

  $S_{1i} \leftarrow algorithm2(\mathbf{X}_i, \mathbf{D}_i, \mathbf{M}_i)$

  $S_1 \leftarrow S_{1i}$; // 1st extraction level

  $S_{aux} \leftarrow S_{1i}$

  if $|S_{aux}| >= l_2$ **then**

    if $|S_{aux}| > l_2$ **then**

      $\mathbf{Z}_j \leftarrow S_{old}$; // $S_{aux} = \{S_{old}, S_{new}\}$, $|S_{old}| = l_2$

      $S_{aux} = S_{new}$

    else

      $\mathbf{Z}_j \leftarrow S_{aux}$

      $S_{aux} \leftarrow \emptyset$

    end

    $\mathbf{D}_j \leftarrow dist(\mathbf{Z}_j)$

    $\mathbf{M}_j \leftarrow gabriel(\mathbf{Z}_j, \mathbf{D}_j)$

    $S_2 \leftarrow algorithm2(\mathbf{Z}_j, \mathbf{D}_j, \mathbf{M}_j)$; // 2nd extraction level

  end

  $i = i + l_1$

end

Algorithm 4: Extractor Algorithm of Representative Subsets.

4.5 Experiments

This section presents experiments that were conducted with the proposed approach to define the parameters of an one-class RBF neural network and with the Algorithm 4, aiming to evaluate their performance, when applied to the one-class problem. This section also presents the result of a test carried out with a real data set from an industrial process, in which the objective was to identify when the process operation is outside of the normal condition, using both proposed methods jointly.

4.5.1 Comparative Tests

This section presents comparative tests with the proposed one-class RBF network when performing the classification of data sets, using only the information of one of the classes to induce the model. The Algorithm 4 is also evaluated as a sampling procedure, providing representative points from the available class, that are used for the definition of the network parameters. It is also assessed the use of the complement of an independent dominating set, induced with Algorithm 4, when providing the information to set the network parameters. The proposed methods are compared with the one-class SVM and with an RBF network that uses both classes to induce the model, as a reference of binary classifier.

The tests were divided in two groups. In the first, the approach that defines the parameters of the RBF network is applied direct with the entire class available from the data sets. In this group, the following data sets were tested: *Spirals*, *Two moons*, *Two moons resized*, *Checkerboard*, *Gaussians*, *Iris* and *Banknote*. The second group consist of tests in which the Algorithm 4 was used previously to extract a representative set, from which the RBF network was induced. For this group, the following data sets were used: *HTRU2*, *Magic Gamma* and *Dry bean*. The description of each data set is presented in Table 4.1. Some of the data set are not binary and for this reason, had part of the classes binding to form the negative class, used to induce the network. The *Iris* data set had its classes *versicolor* and *virginica* set as negative class. *Spirals*, *Two moons*, *Checkerboard* and *Gaussians* are the same data sets used in Chapter 3, although without drifts, in which one class was set as positive and the other as negative. The *Two moons resized* had both classes adopted as negative and a introduced grid of points, covering the variables ranges, as the positive class. The *Dry bean* data set had the class *Bombay* defined as positive and the others as negative. The variables from all data sets were normalized in the [0,1] interval. The tests were conducted using cross-validation with 10 folds.

The comparison between the use of a independent dominating set $|S|$ and its complement $|V - S|$, aiming to define the parameters of an one-class RBF network was done with the data sets from the first group. The definition of a value for the radius h, with Equation 4.3, was adopted in both cases. The results of the cross-validation test are presented in Table 4.2, which shows the number of points in each subset extracted with the Algorithm 2, the metrics of Accuracy and Area under the ROC curve for the RBF network, and the time (s) required to induce the Gabriel Graph, find the dominating set with Algorithm 2 and to define and test the

Table 4.1: Data set characteristics.

Data sets	n	Positive/Negative	Size	Approach tested
Spirals	2	(1.000 / 1.000)	2.000	RBF
Two moons	2	(1.000 / 1.000)	2.000	RBF
Two moons Resized	2	(100 / 300)	400	RBF
Checkerboard	2	(1.000 / 1.000)	2.000	RBF
Gaussians	2	(1.000 / 1.000)	2.000	RBF
Iris	4	(50 / 100)	150	RBF
Banknote	5	(610 / 762)	1.372	RBF
HTRU2	9	(1.639 / 16.259)	17.898	Algorithm 4 + RBF
Magic Gamma	11	(6.688 / 12.332)	19.020	Algorithm 4 + RBF
Dry bean	17	(522 / 13.089)	13.611	Algorithm 4 + RBF

RBF network.

One difference between the two cases tested, that can be clearly seem from the results of Table 4.2, is the difference already known on the number of dominating points between sets S and $V - S$ (also producing different number of Radial functions) and the result in the value of h. An inverse relation is observed for number of dominating points and the value of h. This is a intrinsic characteristic of the proposed method, which defines the value of h based on the structure of the data, aiming at covering the set of points appropriately. Another characteristic that can be observed is that the decision boundary tends to be more tight, into the direction of the training points, when the value of h is smaller. This can be observed in the *Gaussians* and *Checkerboard* data sets, which have classes not separated by any margin, and present a larger value of AUC for the case using the subset $V - S$. Overall, the both subsets ($|S|$ and $|V - S|$) provided similar predictive performance, when defining the parameters of the RBF network.

The comparative tests of the proposed one-class RBF network and the Algorithm 4, with the one class SVM and the binary RBF network, were done using the data sets from the two groups, with cross-validation (10 folds). The tests using the Algorithm 4 were conducted with the following value for the parameter l_1 (100, 200, 300), which defines the size of the subsets in which the data set is divided and independently processed. The parameters of the RBF network was defined using the independent dominating set $|S|$. The one class SVM and the binary RBF network were tested using a grid search jointly with the cross-validation. The OCSVM was

defined with a Radial kernel with the values of radius ($1e^{-7}$, $1e^{-6}$, $\cdots$, $1e^{6}$, $1e^{7}$) and with the values of ν ($1e^{-7}$, $1e^{-6}$, $\cdots$, $1e^{-1}$, $2e^{-1}$, $\cdots$, $5e^{-1}$). The binary RBF network was trained using the K-means method to define the center and the dispersion factor of the Radial functions, and the least squared method to define the weights of the output layer. The number of centers considered in the grid were the following rounded percentage of the training data size (5%, 10%, 15%, 20%, 25%, 30%). The radius of each Radial functions was defined as the mean distance of the neighbors points. Once the *Two moons resized* data set does have an appropriate positive class, the binary RBF network was not tested with it.

The results of the comparative tests are presented in Table 4.3, in which the metrics of Accuracy and AUC, and the respective time (s) of execution for each method is shown. For the data sets from the first group, the predictive performance of the one-class RBF network was above of those reached with the OCSVM, and close to the values of Accuracy and AUC, achieved by the binary RBF network. For the tests with the data sets from the second group, the proposed approach showed a better performance than the OCSVM model, with the *Magic Gamma* data set, for all values of l_1 tested, and for the *Dry bean* data set, using $l_1 = 300$. The performance of the proposed approach with the *HTRU2* data set were below of the one achieved by the OCSVM model, for all values of l_1 tested. When the proposed approach (using $l_1 = 300$) and the OCSVM model are Compared with the binary RBF network, their predictive performance were close to this last for the *Dry bean* data set, and present a large difference on the other data set. Regarding the time (s) of execution, the proposed approach required more time than the OCSVM, with the tests using the data sets from the first group. For data sets from the second group, the proposed approach presented smaller values than the OCSVM. Probably the conjugated use of Algorithm 4 with the one-class RBF network was the reason of time reduced. These two observation is also valid when comparing the proposed approach with the binary RBF network. The OCSVM and the binary RBF network presented similar values, for both group of data sets tested. Lastly, a pertinent observation is the fact that the proposed one-class RBF network does not requires configuration of hyperparameters, while it is necessary in the OCSVM model and in the approach used with the binary RBF network. Another point is that although the proposed RBF network was tested using cross-validation, for a fair comparison with the other methods, its definition does not require its use.

Table 4.2: Cross-validation results when using the sets S and $V - S$, in the parameters definition of an one-class RBF network. Table shows number of points in each set, the mean radius, the accuracy and AUC metrics, and time(s).

Data sets	**Set**	$\|V\|$	$\|S\|$	h	$r(\mathbf{x}, \mathbf{c}, h)$ **centered over** S		
					Acc.	**AUC**	**time**
Iris	S	90	21	0.1235	97.60 ± 5e-2	0.985 ± 3e-2	0.25 ± 3e-3
Spirals		900	294	0.0099	99.37 ± 6e-3	0.993 ± 6e-3	44.94 ± 2e-0
Two moons		900	279	0.0123	95.53 ± 1e-2	0.935 ± 2e-2	46.29 ± 2e-0
Checkerboard		900	281	0.0183	87.00 ± 1e-2	0.806 ± 2e-2	44.93 ± 6e-1
Gaussians		900	273	0.0132	83.53 ± 3e-2	0.755 ± 4e-2	44.86 ± 6e-1
Banknote		684	161	0.0426	97.09 ± 1e-2	0.953 ± 2e-2	26.59 ± 3e-1
Two moons Res.		270	91	0.0282	91.14 ± 3e-2	0.723 ± 7e-2	2.60 ± 8e-3
Data sets	**Set**	$\|V\|$	$\|V - S\|$	h	$r(\mathbf{x}, \mathbf{c}, h)$ **centered over** $V - S$		
					Acc.	**AUC**	**time**
Iris	$V - S$	90	69	0.0893	95.20 ± 6e-2	0.970 ± 4e-2	0.27 ± 8e-3
Spirals		900	606	0.0076	99.40 ± 9e-3	0.995 ± 7e-3	44.89 ± 4e-1
Two moons		900	621	0.0100	96.90 ± 1e-2	0.958 ± 2e-2	45.64 ± 9e-1
Checkerboard		900	619	0.0139	90.23 ± 2e-2	0.855 ± 3e-2	46.10 ± 1e-0
Gaussians		900	627	0.0102	87.77 ± 2e-2	0.818 ± 4e-2	45.07 ± 7e-1
Banknote		684	525	0.0321	78.68 ± 6e-2	0.629 ± 1e-1	27.33 ± 5e-1
Two moons Res.		270	179	0.0231	90.00 ± 2e-2	0.675 ± 7e-2	2.79 ± 1e-1

Table 4.3: Comparative test with the proposed one-class RBF network and Algorithm 4, with the models OCSVM and binary RBF network.

Data sets	**Proposed Approach(es)**					**OCSVM**			**RBF (2 classes)**		
	l_1	$\|S\|$	Acc.	AUC	time	Acc.	AUC	time	Acc.	AUC	time
Spirals	-	294	99.37 ± 6e-1	0.9925 ± 6e-3	43.24 ± 4e-1	75.23 ± 3e-2	0.6300 ± 5e-2	1.20 ± 4e-3	100.00 ± 0e-0	1.0000 ± 0e-0	2.26 ± 8e-3
Two moons	-	279	95.53 ± 1e-0	0.9345 ± 1e-2	43.09 ± 1e-1	86.00 ± 1e-2	0.7930 ± 2e-2	1.30 ± 7e-3	98.93 ± 3e-3	0.9885 ± 4e-3	2.34 ± 1e-2
Two moons res.	-	91	91.14 ± 3e-2	0.723 ± 7e-2	2.60 ± 8e-3	86.71 ± 4e-2	0.6350 ± 5e-2	0.16 ± 7e-4	-	-	-
Checkerboard	-	281	87.00 ± 1e-0	0.8055 ± 2e-2	43.13 ± 2e-1	67.30 ± 2e-2	0.5115 ± 3e-2	1.31 ± 7e-3	92.17 ± 3e-2	0.9250 ± 2e-2	2.44 ± 1e-2
Gaussians	-	273	83.53 ± 3e-2	0.7545 ± 4e-2	44.24 ± 5e-1	80.03 ± 5e-2	0.7040 ± 8e-2	1.34 ± 4e-3	96.03 ± 7e-3	0.9580 ± 1e-2	2.45 ± 6e-2
Iris	-	21	97.60 ± 5e-0	0.9850 ± 3e-2	0.24 ± 2e-3	92.00 ± 9e-2	0.9500 ± 5e-2	0.06 ± 6e-4	100.00 ± 0e-0	1.0000 ± 0e-0	0.13 ± 1e-2
Banknote	-	161	97.09 ± 1e-0	0.9531 ± 2e-2	25.51 ± 1e-1	78.16 ± 2e-2	0.6249 ± 4e-2	0.74 ± 4e-3	99.44 ± 9e-3	0.9961 ± 6e-3	1.47 ± 4e-3
HTRU2	100	3,353	96.03 ± 2e-3	0.5936 ± 2e-2	8.78 ± 1e-1	96.34 ± 2e-3	0.6230 ± 2e-2	616.25 ± 1e+1	98.42 ± 2e-3	0.9042 ± 1e-2	351.51 ± 5e-0
	200	3,004	95.83 ± 2e-3	0.5699 ± 2e-2	21.20 ± 3e-1						
	300	2,729	95.70 ± 1e-3	0.5545 ± 1e-2	50.54 ± 3e-1						
Magic gamma	100	2,051	79.84 ± 6e-3	0.5277 ± 1e-2	6.35 ± 1e-1	78.90 ± 2e-3	0.5064 ± 4e-3	390.37 ± 5e-0	88.01 ± 7e-3	0.8179 ± 7e-3	341.35 ± 17e-0
	200	1,765	82.43 ± 8e-3	0.5899 ± 2e-2	17.50 ± 3e-1						
	300	1,603	81.09 ± 6e-3	0.5576 ± 1e-2	41.05 ± 7e-1						
Dry bean	100	2,714	99.50 ± 3e-3	0.8795 ± 9e-2	8.56 ± 2e-1	99.87 ± 1e-3	0.9872 ± 1e-2	450.10 ± 8e-0	99.99 ± 2e-4	1.0000 ± 1e-4	199.14 ± 5e-0
	200	2,409	99.64 ± 3e-3	0.9140 ± 7e-2	25.23 ± 4e-1						
	300	2,245	99.93 ± 6e-4	0.9893 ± 1e-2	55.97 ± 9e-1						

4.5.2 Test With a Data Set From a Real Industrial Process

The proposed one-class RBF network and the proposed method to extract representative subsets (Algorithm 4) were tested with a data set collected from a real industrial process. The data set consists of 6 variables, in the real domain, from a flotation unit of an iron ore mining company. Timestamp tags for 18 faults that occurred with a pump of the process is also available. A total of 2,284,459 points of normal operation and 1,060,679 points of pre-fault occurrences were sampled for the test. For each fault a sampling interval of 7 days (60,480 points), ending in the fault occurrence, was considered as a pre-fault period. Although a true label for each point of the data set was not available, two distinct classes were created, according to samplings procedures: *Normal* and *Pre-fault*. The former with a majority number of points centering around a mean expected value of each variable, and the latter containing a mixture of points near the mean expected mean and instants in which some of the variables show some deviation from the mean value. The variables from the data set were normalized in the [0,1] interval. The objective of the test was to identify when the process operation is outside of the normal conditions, with the proposed one-class RBF network, when trained only with the *Normal* class.

A cross-validation test with 5 folds was conducted, training the model with 4/5 of the *Normal* class and testing with all points from the *Pre-fault* class and with the remaining 1/5 of the *Normal* class. In each fold, the Algorithm 4 was used to extract a representative subset from each fold of the *Normal* class V, using 4 extraction levels. The number of dominating points selected for each subset is presented in Table 4.4. These subsets were obtained from each fold V, containing 4/5 of the points from the *Normal* class. In the following, each subset S_4 was used to defined the parameters of the RBF network, with Equations 4.2 and 4.3, and centering Radial functions in each point in the subset S_4. Aiming at allowing some points from the *Normal* class to be outside the decision boundary, a value of 0.5% was adopted as a rejection level, enabling the definition of the limit θ_{RBF}. It was adopted the label Positive for the points outside the decision boundary and Negative for those founded inside it.

The results of the cross-validation test are presented in Table 4.5, with the mean percentage of points, classified as Positive, for the folds of training and test of the *Normal* class, in separated, and for the *Pre-fault* class. The results shows that the proposed approach classified 13.63 $\%$ of point from the *Pre-fault* as Positive, while the number of Positives in both folds of

Table 4.4: Number of points in each subset of the cross-validation test, founded with the Algorithm 4.

Representative sets	fold 1	fold 2	fold 3	fold 4	fold 5
$\|V\|$	1,827,039	1,826,895	1,826,980	1,826,957	1,827,028
$\|S_1\|$	351,096	354,860	352,828	355,303	355,957
$\|S_2\|$	68,287	70,317	68,852	69,463	68,909
$\|S_3\|$	13,773	14,360	13,872	13,872	13,632
$\|S_4\|$	2,641	2,785	2,700	2,723	2,652

training and test of the *Normal* class were equal (for the number of decimal adopted). In other words, the network was able to identify points outside of the expected regular behavior, while no false positive was founded in the test fold of the *Normal* class.

Table 4.5: Cross-validation results.

Data sets	Points classified as Positive [%]
training	0.54
test (normal)	0.54
test (pre-fault)	13.63

4.6 Conclusion

The presented chapter described a new approach to define the parameters of an one-class RBF network, that is formulate with the objective to establish a decision boundary, when induced from a set of points representing only one class. This is an important characteristic to have in a classifier, when no information (or almost none) about one of the classes is available. The parameters of the proposed network are defined based on the structural information of the data, that is provided by a dominating set of the Gabriel graph, in a procedure that does not require the setting of hyperparameters. The proposed approach defines a well distributed number of Radial functions, in the input domain, and defines a stable radius that is determined aiming at covering all the data points. The comparative tests showed that this one-class RBF network was able to outperformed the OCSVM model, with the data sets tested, from the first group, and was even capable to near reach the predictive performance of the binary RBF network, that used both classes to induce the model.

The second proposed method, Algorithm 4, extracted a nested representative subsets from a data set, using the independent dominating set (Algorithm 2) and the Gabirel graph. The method have different levels of extraction. In the first level, the original set is divided in subsets and processed independently. From the second to the last extraction level, the subsets are still processed independently, but with some level of interdependence, as some points for one subset were previously in distinct subsets, in previously levels. The final subsets in the output tends to sustaining the representation of the original set, preserving representative points.

The jointly use of both methods to solve the problem of identify when the process operation is outside of the normal conditions, demonstrated the quality of the Algorithm 4 to select a relative smaller number of representative points, from a big data set, and a precision RBF network capable to reject false positives.

Chapter 5

Online Learning With a Regularized SLFN Network Based on Random Projections

5.1 Introduction

This chapter describes the training procedure, named SLFN-RP, that was conceived for online learning of Single hidden Layer Feedforward neural Networks. The method relies on a sliding window that is capable to shrink its size, after the occurrence of a concept drift, allowing a fast adaptation of the network to the new scenario.

A section of related work presents some approaches related to the proposed method. In the following, the SLFN-RP algorithm is presented, detailing each step involved in the method. Next, an experiment section describes the comparative tests that were conducted and presents the results that were obtained.

5.2 Related Work

Online Sequential ELM (OS-ELM) [Liang et al., 2006] is a variation of ELM algorithm that sequentially updates the model using data in either a point-by-point or batch mode. The algorithm consists of two stages: firstly, an initial model is trained using the ELM method with a single batch of points; then, the incremental learning is initialized. The update step for the

case with just one point is given by Equation 5.1.

$$\mathbf{P}_t = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1}\mathbf{h}_t\mathbf{h}_t^T\mathbf{P}_{t-1}}{1 + \mathbf{h}_t^T\mathbf{P}_{t-1}\mathbf{h}_t}, \tag{5.1}$$

in which $\mathbf{P}_t = (\mathbf{H}_t^T\mathbf{H}_t)^{-1}$, and $\mathbf{h}_t$ is the hidden layer output for $\mathbf{x}_t$. The output weights are updated according to Equation 5.2

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{P}_t\mathbf{h}_t(\mathbf{y}_t - \mathbf{h}_t^T\mathbf{w}_{t-1}), \tag{5.2}$$

where $\mathbf{y}_t$ is the label related to $\mathbf{x}_t$ and $\mathbf{w}_{t-1}$ is the output weights vector for instant $t-1$.

Despite the short time of training and good generalization performance achieved by ELM algorithm, over-fitting tends to occur due to lack of constraints in the output weights norms during the network training. A common approach to treat this issue is to employ regularization [Deng et al., 2009], in order to control the magnitude of network weights. An off-line ELM alternative, that is parameter free [Silvestre et al., 2015] has a regularization strategy based on affinity matrices using *a priori* data structure information. An off-line approach to apply regularization over ELM networks, using synthetic points around the geometric vectors, is found in [Assis et al., 2021]. The method that was proved to have the same effect as the Tikhonov regularization, improves the generalization performance.

The Online Sequential Regularized ELM (OS-RELM) [Shao and Er, 2016] is an ELM incremental extension that is able to use as inputs a batch or a single point to update the network, employing regularization. With a different approach than OS-ELM, which updates the weight vector $\mathbf{w}_t$ as a function of the previous value $\mathbf{w}_{t-1}$, OS-RELM is based on an incremental implementation of inverse matrix calculation. Equation 5.3 is used to define the initial model,

$$\mathbf{w}_0 = \mathbf{H}_0^T\mathbf{A}_0^{-1}\mathbf{y}_0, \tag{5.3}$$

where $\mathbf{A}_0 = (\mathbf{H}_0\mathbf{H}_0^T + \lambda\mathbf{I})$ and λ is the regularization coefficient.

The OS-RELM algorithm is accomplished in two stages with distinct equations to update $\mathbf{A}_t^{-1}$. The first one comprises the situation in which the number of points N learned so far is smaller than the number of hidden neurons p, whereas in the second stage $N \geq p$. Equation

5.4 is used to update the inverse matrix $\mathbf{A}_t^{-1}$ when $N < p$,

$$\mathbf{A}_t^{-1} = \begin{bmatrix} \mathbf{F} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}, \tag{5.4}$$

$$\mathbf{F} = \mathbf{A}_{t-1}^{-1} + \mathbf{A}_{t-1}^{-1}\mathbf{H}_{t-1}\delta\mathbf{H}_t^T\mathbf{S}_t^{-1}\delta\mathbf{H}_t\mathbf{H}_{t-1}^T\mathbf{A}_{t-1}^{-1},$$

$$\mathbf{B} = -\mathbf{A}_{t-1}^{-1}\mathbf{H}_{t-1}\delta\mathbf{H}_t^T\mathbf{S}_t^{-1},$$

$$\mathbf{C} = -\mathbf{S}_t^{-1}\delta\mathbf{H}_t\mathbf{H}_{t-1}^T\mathbf{A}_{t-1}^{-1},$$

$$\mathbf{D} = \mathbf{S}_t^{-1},$$

$$\mathbf{S}_t = \delta\mathbf{H}_t\delta\mathbf{H}_t^T + \lambda\mathbf{I} - \delta\mathbf{H}_t\mathbf{H}_{t-1}^T\mathbf{A}_{t-1}^{-1}\mathbf{H}_{t-1}\delta\mathbf{H}_t^T,$$

where $\delta\mathbf{H}_t$ is the hidden layer output for the most recent point (or batch), and $\mathbf{H}_t$ is defined with Equation 5.5.

$$\mathbf{H}_t = \begin{bmatrix} \mathbf{H}_{t-1} \\ \delta\mathbf{H}_t \end{bmatrix}. \tag{5.5}$$

The transition from $N < p$ to $N \geq p$ requires first the computation of the inverse matrix defined as $\mathbf{A}_t^{-1} = (\mathbf{H}_t\mathbf{H}_t^T + \lambda\mathbf{I})^{-1}$. Then, when $N \geq p$, matrix $\mathbf{A}_t^{-1}$ is updated with Equation 5.6.

$$\mathbf{A}_t^{-1} = \mathbf{A}_{t-1}^{-1} - \mathbf{A}_{t-1}^{-1}\delta\mathbf{H}_t^T(\mathbf{I} + \delta\mathbf{H}_t\mathbf{A}_{t-1}^{-1}\delta\mathbf{H}_t^T)\delta\mathbf{H}_t^T\mathbf{A}_{t-1}^{-1}. \tag{5.6}$$

After the updated of matrix $\mathbf{A_t}^{-1}$, with Equation 5.6, the output layer weights are calculated with Equation 5.3. The OS-RELM regularization coefficient is estimated in a regular basis with leave-one-out cross-validation, followed by a recalculation of $\mathbf{A_t}^{-1}$. An important characteristic of OS-RELM is the necessity to retain all data already processed in memory.

5.3 Proposed Approach

The training procedure proposed is conceived for online learning of an SLFN network applied to online learning of non-stationary data streams. Considering an SLFN with linear output, its output equation is given with Equation 5.7.

$$f(\mathbf{x}, \mathbf{Z}, \mathbf{w}_t) = \sum_{i=1}^{p} w_{ti}h_i(\mathbf{x}, \mathbf{z}_i) + w_{t0}, \tag{5.7}$$

where p is the number of hidden neurons, $\mathbf{Z}$ is a matrix with weights $\mathbf{z}_i$ from each neuron in the hidden layer, $h_i(\cdot)$ is the ith activation function and $\mathbf{w}_t = [w_{t0}, w_{t1}, ..., w_{tp}]$ is the output weight vector for an instant t.

The procedure to induce parameters is different for each SLFN-RP layer. When the first N points $(\mathbf{x}_i, y_i)_{i=t-N}^{t}$ are presented, weights from hidden layer ($\mathbf{Z}$) are randomly defined and remain constant. Some degree of uncertainty is imprinted to the model by the stochasticity of random projections, however, separability is aimed by the high dimensional mapping [Cover, 1965]. For additive neurons the hidden layer output is defined with Equation 5.8

$$\mathbf{H} = \begin{bmatrix} h(\mathbf{z'}_1 \cdot \mathbf{x}_1 + z_{b1}) & \dots & h(\mathbf{z'}_p \cdot \mathbf{x}_1 + z_{bp}) \\ \dots & \vdots & \dots \\ h(\mathbf{z'}_1 \cdot \mathbf{x}_N + z_{b1}) & \dots & h(\mathbf{z'}_p \cdot \mathbf{x}_N + z_{bp}) \end{bmatrix}_{N \times p}, \tag{5.8}$$

where $\mathbf{z}_i = [\mathbf{z'}_i \quad z_{bi}]$ is the weight vector and bias from the *i*th hidden neuron.

SLFN-RP employs Tikhonov's regularization [Deng et al., 2009] to control the magnitude of the network weights. Given an initial set of points $(\mathbf{x}_i, y_i)_{i=t-N}^{t}$ and a randomly defined matrix $\mathbf{Z}$, the hidden layer output $\mathbf{H}_0$ is calculated with Equation 5.8. Then, Equation 5.9 is used to obtain the inverse matrix $\mathbf{A}_t^{-1}$ for $t = 0$.

$$\mathbf{A}_0^{-1} = (\mathbf{H}_0^T \mathbf{H}_0 + \lambda \mathbf{I})^{-1}, \tag{5.9}$$

where λ is the regularization coefficient.

Using the least squares method the output layer weights vector $\mathbf{w_t}$ is defined according to Equation 5.10.

$$\mathbf{w}_0 = \mathbf{H}_0 \mathbf{A}_0^{-1} \mathbf{y}_0. \tag{5.10}$$

The regularization coefficient is also estimated with the initial set of points, using the *leave-one-out* cross validation (LOO-CV). For linear prediction models, it is possible to calculate the LOO-CV error directly. Equation 5.11 defines the LOO-CV squared error.

$$\mathbf{e}_{loo}^2 = \begin{bmatrix} e_1^2 \\ e_2^2 \\ \vdots \\ e_N^2 \end{bmatrix} = \frac{1}{N} \mathbf{y}_0^T \mathbf{P} (diag(\mathbf{P}))^{-2} \mathbf{P} \mathbf{y}_0, \tag{5.11}$$

where $e_i^2 = y_i - f_i(\mathbf{x}_i)^2$, is the squared error of the *i*th point, $\mathbf{H}_0$ is the hidden layer output for

the N initial points, and $\mathbf{P}$ is defined according to Equation 5.12.

$$\begin{aligned}
\mathbf{y}_0 - \hat{\mathbf{y}}_0 &= \mathbf{y}_0 - \mathbf{H}_0\mathbf{w}_0, \\
&= \mathbf{y}_0 - \mathbf{H}_0\mathbf{A}_0^{-1}\mathbf{H}_0^T\mathbf{y}_0, \\
&= (\mathbf{I} - \mathbf{H}_0\mathbf{A}_0^{-1}\mathbf{H}_0^T)\mathbf{y}_0, \\
&= \mathbf{P}\mathbf{y}_0, \\
\mathbf{P} &= \mathbf{I} - \mathbf{H}_0\mathbf{A}_0^{-1}\mathbf{H}_0^T,
\end{aligned} \tag{5.12}$$

where $\hat{\mathbf{y}}_0$ is the network output for the N points. Given a vector of regularization coefficients $\boldsymbol{\lambda}$, the optimum lambda value λ^* is chosen based on the minimization of the LOO-CV error. Then, the initial network is obtained with λ^*.

An adaptive moving window $\mathbf{W}_{mov}$ is used to feed SLFN-RP while learning from non-stationary data streams. The $\mathbf{W}_{mov}$ is a FIFO type window (first-in-first-out), which retains the most recent tuples $(\mathbf{x}, y)$ in its memory. The newest tuple $(\mathbf{x}_t, y_t)$ is saved in $\mathbf{W}_{mov_t}$ according to Equation 5.13.

$$\begin{aligned}
\mathbf{W}_{mov_t} &= \begin{bmatrix} \mathbf{W}' \\ [\mathbf{x}_t \quad y_t] \end{bmatrix} \\
\mathbf{W}_{mov_{t-1}} &= \begin{bmatrix} \mathbf{W}'' \\ \mathbf{W}' \end{bmatrix}
\end{aligned} \tag{5.13}$$

where $\mathbf{W}_{mov_{t-1}}$ is the window from instant $t-1$, sub-matrix $\mathbf{W}''$ is the representation of deleted tuples from $\mathbf{W}_{mov_{t-1}}$, and $\mathbf{W}'$ stores the remaining tuples.

Adaptation of window $\mathbf{W}_{mov_t}$ is proposed in order to cope with concept drifts, in such way that the number of tuples in $\mathbf{W}_{mov_t}$ varies according to each instant. During drift occurrences, the older points in $\mathbf{W}_{mov}$ are regarded as less informative of the new scenario [Bifet and Gavalda, 2007]. Thus $\mathbf{W}_{mov}$ has a maximum W_{max} and a minimum W_{min} size. The size of $\mathbf{W}_{mov}$ increases during stable conditions, up to W_{max}, retaining a larger number of patterns to be learned still. While in drift, $\mathbf{W}_{mov}$ size shrinks to W_{min} removing older and less informative points. Then, $\mathbf{W}_{mov}$ increases as new points are available, until W_{max} is reached. This adaptive characteristic tends to smooth the drift impact on the classifier, with adaptation to the new scenario.

After the estimation of $\mathbf{w}_0$, and every time a new point is presented, the inverse matrix $\mathbf{A}_t^{-1}$, the adaptive window W_{mov} and the output layer weights $\mathbf{w}_t$ are modified. The drift

detection is performed separately by a suitable algorithm.

In the absence of drifts, the older tuple $(\mathbf{x}_o, y_o)$ from $\mathbf{W}_{mov}$ is deleted and the newest tuple $(\mathbf{x}_t, y_t)$ is saved using Equation 5.13, with $\mathbf{W}'' = [\mathbf{x}_o, y_o]$. Then the inverse matrix $\mathbf{A}_t^{-1}$ is updated in two steps, using the Sherman-Morrison formula [Golub and Van Loan, 2013].

The first step consists in add the hidden layer output $\mathbf{h}_t$, for the new point $\mathbf{x}_t$, in matrix $\mathbf{A}_{t-1}^{-1}$ to form a intermediate matrix $\mathbf{A'}_t^{-1}$, with Equation 5.14.

$$\mathbf{A'}_t^{-1} = \mathbf{A}_{t-1}^{-1} - \frac{\mathbf{A}_{t-1}^{-1}\mathbf{h}_t\mathbf{h}_t^T\mathbf{A}_{t-1}^{-1}}{1 + \mathbf{h}_t^T\mathbf{A}_{t-1}^{-1}\mathbf{h}_t}. \tag{5.14}$$

The second step consist in remove the hidden layer output $\mathbf{h}_o$, associated with $(\mathbf{x}_o, y_o)$, from $\mathbf{A'}_t^{-1}$, according to Equation 5.15.

$$\mathbf{A}_t^{-1} = \mathbf{A'}_t^{-1} + \frac{\mathbf{A'}_t^{-1}\mathbf{h}_o\mathbf{h}_o^T\mathbf{A'}_t^{-1}}{1 - \mathbf{h}_o^T\mathbf{A'}_t^{-1}\mathbf{h}_o}. \tag{5.15}$$

When no drift has occurred and the size of W_{mov} is smaller than W_{max}, only the most recent tuple $(\mathbf{x_t}, y_t)$ is saved in W_{mov}, and no point is discarded ($\mathbf{W}''$ is empty in this case). The inverse matrix is then updated only with Equation 5.14 and the output weights vector is calculated with Equation 5.10.

Whenever a concept drift is detected, the most recent tuple $(\mathbf{x_t}, y_t)$ is saved in W_{mov}, as it would occur in the absence of drift. However, W_{mov} adapts itself, keeping only the most recent W_{min} points, or a smaller number if $W_{mov_{t-1}}$ has a number of tuples smaller than W_{min}. Equations 5.16 and 5.17 define the size of sub-matrices $\mathbf{W}''$ and $\mathbf{W}'$.

$$size(\mathbf{W}'') = \begin{cases} (W_{size} - W_{min} + 1), & \text{if} \quad W_{size} \geq W_{min}. \\ 0, & \text{otherwise.} \end{cases} \tag{5.16}$$

$$size(\mathbf{W}') = W_{size} - size(\mathbf{W}''), \tag{5.17}$$

where $size(\cdot)$ is the number of tuples and W_{size} represents the size of $W_{mov_{t-1}}$. Given $size(\mathbf{W}'')$, $size(\mathbf{W}')$ and $(\mathbf{x_t}, y_t)$, the new adapted window is then obtained with Equation 5.13.

The Learning steps of the training method SFLN-RP are summarized in Algorithm 5.

input : $\{\{\mathbf{x}_t, y_t\}_{t=1}^{N} | \mathbf{x} \in \mathbb{R}^n, y \in \{-1, 1\}, N \to \infty\}$, p, $\boldsymbol{\lambda}$, W_{max}, W_{min}

output: $\mathbf{Z}$, $\mathbf{w}_t$, λ

$\mathbf{W}_{mov_0} \leftarrow (\mathbf{X}_0, \mathbf{y}_0)$; // Access first points

$\mathbf{Z}$; // Define randomly

$\mathbf{H}_0 \leftarrow (\mathbf{Z}, \mathbf{W}_{mov_0})$; // (Eq. 5.8)

$\lambda^* \leftarrow loo(\boldsymbol{\lambda})$; // Leave-one-out (Eq. 5.11)

$\mathbf{A}_0^{-1} = (\mathbf{H}_0^T \mathbf{H}_0 + \lambda \mathbf{I})^{-1}$; // (Eq. 5.9)

$\mathbf{w}_0 = \mathbf{H}_0 \mathbf{A}_0^{-1} \mathbf{y}_0$; // (Eq. 5.10)

$t = 1$

while $t \neq N$ **do**

  $\mathbf{W}_{\mathbf{mov_t}} \leftarrow \{\mathbf{x}_t, y_t\}$; // Access new sample

  DriftFlag $\leftarrow$ *(True/False)* ; // Drift occurrence

  if *DriftFlag* = *False* **then**

    if $size(\mathbf{W}_{mov_t}) \leq W_{max}$ **then**

      Update $\mathbf{A}_t^{-1}$; // (Eq. 5.14)

      Update $\mathbf{W}_{\mathbf{mov}}$; // add $(\mathbf{x}_t, y_t)$ (Eq. 5.13)

    else

      Update $\mathbf{A}_t^{-1}$; // (Eq. 5.14 and 5.15)

      Update $\mathbf{W}_{mov}$; // add $(\mathbf{x}_t, y_t)$, delete $(\mathbf{x}_o, y_o)$

    end

  else

    $size(\mathbf{W}'), size(\mathbf{W}'') \leftarrow (\mathbf{W}_{mov_{t-1}}, W_{min})$; // (Eq. 5.17 and 5.16)

    Adapt $\mathbf{W}_{\mathbf{mov}}$; // add $(\mathbf{x}_t, y_t)$, turn into W_{min} size

    $\mathbf{A}_t^{-1} = (\mathbf{H}_t^T \mathbf{H}_t + \lambda \mathbf{I})^{-1}$; // (Eq. 5.9)

  end

  $\mathbf{w}_t = \mathbf{H}_t \mathbf{A}_t^{-1} \mathbf{y}_t$; // (Eq. 5.10)

  $t = t + 1$

end

Algorithm 5: SLFN-RP.

5.4 Experiments

In this section, a performance comparison of the methods SLFN-RP with OS-ELM and OS-RELM. Four synthetic datasets that are used in the assessment. These datasets are: *Spirals*

[Ditzler and Polikar, 2011; Yu and Webb, 2019] and *Gaussians* [Ditzler and Polikar, 2011], already described in section 3.6.1; a small version of the *Checkerboard* [Elwell and Polikar, 2011; Ditzler and Polikar, 2011, 2012; Yu and Webb, 2019] , presented in Figure 5.1; and the *Hyperplane* [Ditzler and Polikar, 2011; Yu and Webb, 2019] , that is generated with a hyperplane in d-dimensional space (here d=10), defined as a set of points x in the interval $[0, 1]$ satisfying $\sum_{i=1}^{d} w_i x_i = w_0$. Points that satisfy the condition $\sum_{i=1}^{d} w_i x_i \geq w_0$ are labeled as positive class and those that satisfy $\sum_{i=1}^{d} w_i x_i < w_0$ are labeled as negative. Concept drifts (abrupt) were introduced in this last data set, splitting it into ten batches of equal size with $1,000$ points each. These batches were individually modified: three random attributes were selected, in each batch, and a constant of (-1 or 1) was added to them. Prior to testing all data sets were normalized to the [0,1] interval. Main characteristics of the datasets are presented in Table 5.1.

Table 5.1: Datasets characteristics.

Datasets	Classes	Attributes	Drifts	Size
Spirals	2	2	3	10,000
Hyperplane	2	10	9	10,000
Checkerboard	2	2	3	10,000
Gaussians	2	2	3	10,000

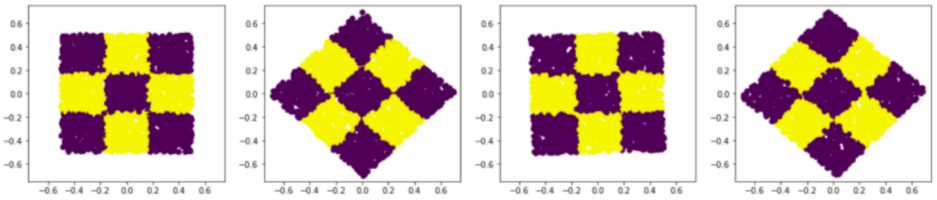


Figure 5.1: Small version of the *Checkerboard* data set, presenting 4 different instants along the sequence of the data points.

The SLFN network tested with the proposed method has $p = 300$ hidden neurons and a single output. The activation functions in the hidden and output neurons are, respectively, hyperbolic tangent and identity. The Hellinger distance was chosen as the drift detection method to be used with the SLFN network in the experiments of this work. Some reasons for that are: (*i*) Hellinger distance is a non parametric metric, thus it is not required to know *a priori* the data distribution; (*ii*) differently from many detection methods that infer a drift based on the

classification error, detection with Hellinger distance considers the variables that are changing; and (*iii*) this method is capable to detect drifts when processing a data set point-by-point. The candidates for the regularization coefficient used in LOO-CV search are $\boldsymbol{\lambda}$=(10, 5, 1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 1e-10). The method HDDDM [Ditzler and Polikar, 2011] that applies Hellinger distance as a metric was chosen to identify concept drifts, using a window size W_{max} from the SLFN-RP model.

The OS-ELM and OS-RELM were tested with the same type of activation function and identical number (300) of hidden neurons than SLFN-RP. The randomly sampled weight vectors of hidden layer neurons were the same for SLFN-RP, OS-ELM and OS-RELM. The adaptive LOO procedure of OS-RELM used corresponding $\boldsymbol{\lambda}$ candidate values, totalizing an equal number of drifts detected by the Hellinger method in each dataset. Tests were repeated 30 times.

A grid search was implemented to assess hyperparameters W_{max} and W_{min}, as shown in Table 5.2. Table 5.3 presents the selected hyperparameter values for each dataset.

Table 5.2: Grid Search hyperparameters values.

W_{max}	$(100, 200, 300, 400)$
W_{min}	$(20, 40, 60, 80)$

Table 5.3: SLFN-RP hyperparameters found with Grid Search.

Datasets	W_{max}	W_{min}
Spirals	400	60
Hyperplane	200	20
Checkerboard	300	60
Gaussians	100	60

The prediction results for the methods SLFN-RP, OS-ELM and OS-RELM methods are shown in Figure 5.2, with the accuracy (moving average) during the predictions. Each graph shows the exact time that a drift occurs, appearing as a dashed line. The result with the *Hyperplane* dataset shows a faster recover after each drift for the SLFN-RP method, when compared to OS-ELM and OS-RELM methods. In the tests performed with the *Spirals* and

Gaussians data sets the networks trained with OS-ELM and OS-RELM seem to baffle labels after 2nd and 3rd drifts, whereas the SLFN-RP method presented a fast reestablishment of performance. For the *Checkerboard*, the SLFN-RP method shows also a fast recovery after drifts, while the OS-ELM and OS-RELM methods presented a considerable drop of accuracy after the first drift. Fast recovery of the SLFN-RP method, during drifts, is explained by the adaptive window, which provides to the network patterns from the new scenario, while points related to the previous scenario tend to be dropped. Thus, the learning quickly focus on the most recent points.

Table 5.4 consolidates the results presented in Figure 5.2 showing the final accuracy for SLFN-RP, OS-ELM and OS-RELM in each dataset. The SLFN-RP method has higher accuracy in all four data sets tested against OS-ELM and OS-RELM algorithms.

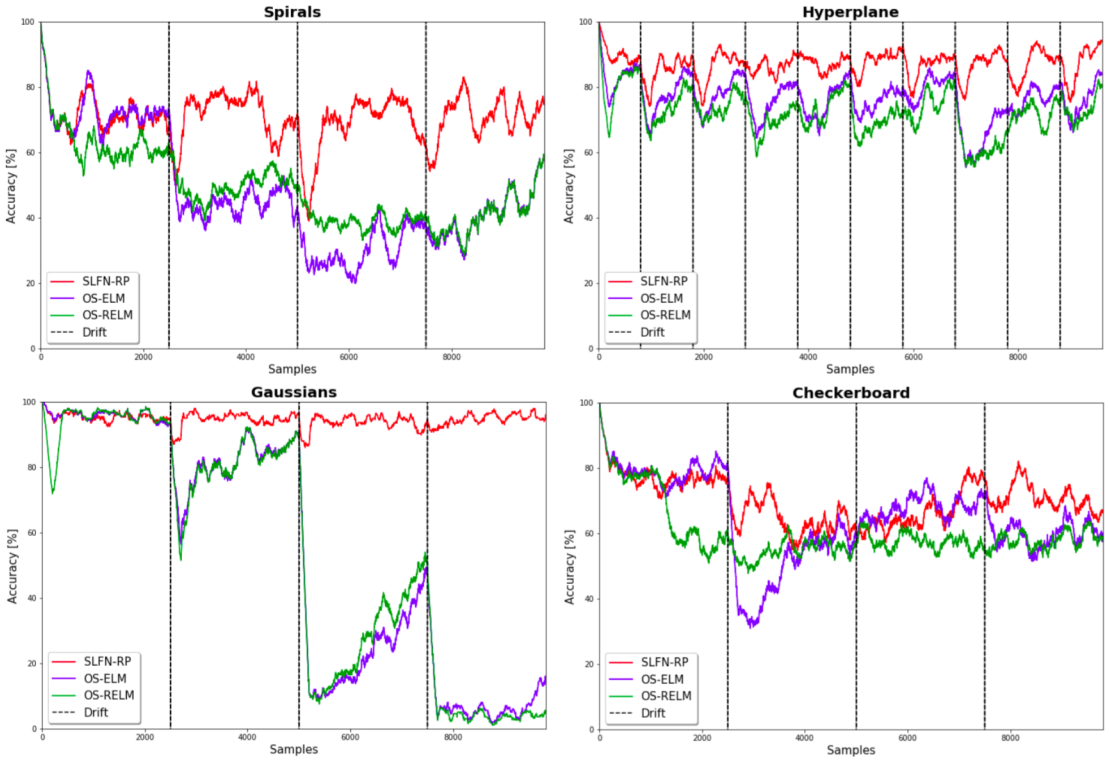


Figure 5.2: Accuracy (moving average) of the methods SLFN-RP, OS-ELM and OS-RELM with the data sets: *Hyperplane*, *Spirals*, *Checkerboard* and *Gaussians*.

A second set of experiments investigates the computational complexity for each method to learn the *Hyperplane* and *Spirals* data sets with 200,000 points each. The time(s) required for

Table 5.4: Accuracy (%) of the methods SLFN-RP, OS-ELM and OS-RELM.

Datasets	**SLFN-RP**	**OS-ELM**	**OS-RELM**
Spirals	70.4 ± 5e-4	47.1 ± 3e-4	48.5 ± 1e-2
Hyperplane	87.1 ± 1e-3	76.6 ± 4e-3	72.4 ± 4e-3
Checkerboard	69.5 ± 1e-2	64.2 ± 3e-3	59.6 ± 3e-2
Gaussians	94.6 ± 2e-4	52.4 ± 3e-4	52.1 ± 1e-3

SLFN-RP, OS-ELM and OS-RELM methods to learn the two data sets is presented in Figure 5.3. The method OS-RELM presented larger time than the SLFN-RP and OS-ELM methods, probably due to the need to retain in memory the data already predicted. The method SLFN-RP presented similar results when comparing to the OS-ELM. This can be explained with the use of an adaptive window with relative small size, the quickly training calculation with matrix operations, and the inverse matrix that is only required to be computed when a drift occurs. It is possible to observe that the dimension of the data sets impact in the processing, with the time slightly greater for the *Hyperplane*.

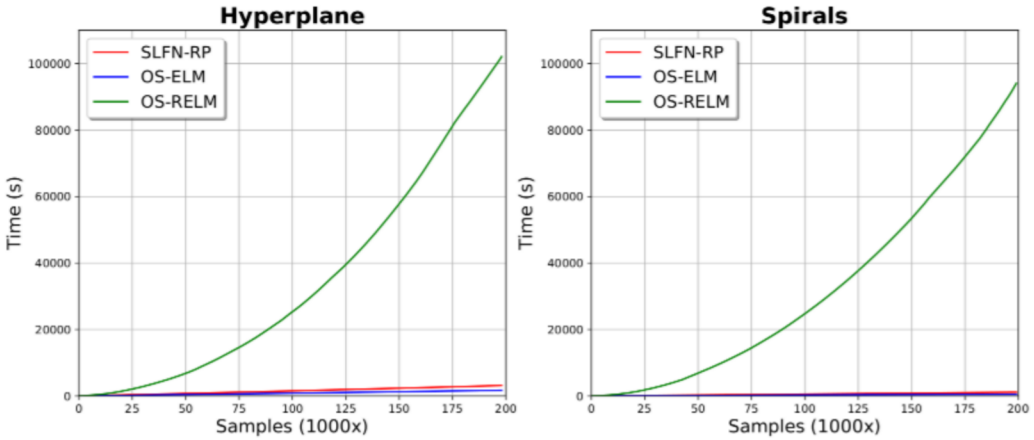


Figure 5.3: Time(s) required for SLFN-RP, OS-ELM and OS-RELM methods to process the *Spirals* and *Hyperplane* datasets.

5.5 Conclusion

This chapter presented a new method to training single hidden layer feedforward neural network for the online learning of non-stationary data streams. The method uses an adaptive window to cope with drifts, allowing the learning to focus on the most recent points. An impor-

tant aspect of this approach is the quick network update, with matrix operations, and an inverse matrix calculation that it is only required after drifts. This training algorithm is of simple implementation and has computational complexity comparable with the OS-ELM method.

Chapter 6

Conclusion

This chapter presents the final conclusions of this thesis. In the following, some proposals based on the methods presented in this thesis are described as possible future works.

6.1 Conclusions

This thesis presented the classification problems of online learning, with non-stationary data streams, and the off-line learning with one-class classifiers. It was discussed characteristics and issues related to each of these cases, specially the ones related to the induction of a model with appropriate capacity. The online learning problem requires a classifier capable to update on a timely basis, adapting to possible concept drifts. Whereas the induction of an one-class classifier should provide an appropriate decision boundary, capable to separate the training points from outliers. The approach adopted in this work to address the requisite of complexity of the inductive function, in both cases, was the use of information from the data structure to automatically define the hyperparameters from models that apply Radial functions, more specifically, an KDE estimator for the online scenario and an RBF network as an one-class classifier. The independent dominating set, and its complement, when induced from the Gabriel graph, which preserves local relations of the nodes connection, tends by definition to result in a distributed subset of dominating points, with representative characteristic of the original set. The centering of a Radial function in each dominating point paves the way for covering of the original set of points, adopting a single radius that is defined based on the distances relation between dominating and non-dominating points.

This configuration approach for Radial functions was applied with the KDE estimator, using the online updating proposal of the Gabriel graph. Such method was able to adjust to different types of concept drifts, with no need to configure hyperparameters. The approach was also applied with the one-class RBF network, demonstrating that the single radius applied in all the Radial functions has a stable characteristic, that tends to result in a decision boundary without disturbance and been positioned relatively close to the extreme points of the limit established. The tests revealed that the subsets extracted with the Algorithm 4 provided representative points to be used in the induction of the one-class RBF network, for both comparative tests and the test with the data set from a real industrial process.

Lastly, the training method for single hidden layer feedforward neural network, for the online learning of non-stationary data streams, demonstrated that the continue updating of the inverse matrix with the adding and discarding of one point, was able to focus the learning on the most recent data scenario, while the window adaptation after a concept drift, provided means for a relative fast recovery. The tests also showed that the steps involved in the training has computational complexity comparable with the OS-ELM method.

6.2 Future Work

The next items highlight some possible topics that could be explored as a future work:

- *Selection of points (sliding window)* - A concern inherent to learning models is the problem known as Catastrophic Forgetting [McCloskey and Cohen, 1989; Ratcliff, 1990; French, 1999]. Firstly observed in connectionist models, this disturbance arises when a knowledge is forgetting due to a new pattern that needs to be assimilated, resulting in performance deterioration. From this perspective, the online SLFN-RP network could have its prediction accuracy increased with a point selection strategy, i.e. as found in active learning algorithm [Horta and de Pádua Braga, 2014]. Indeed, preliminary tests with the SLFN-RP approach, using a second sliding window to save the points founded in the separable boarder of a 2 class problem, that were selected with the method in [Horta and de Pádua Braga, 2014], has shown good results.

- *Dominating Sets* - The present work explored the use of an independent dominating set as a mechanism to extract structural information from data sets, supporting the definition

of complexity for two types of models that were proposed (online KDE Estimator and one-class RBF network). Other types of dominance could be also researched, for use with these particular types of classifiers, as well as for models in different paradigms, once other types of dominance could fit better for a different type of classifier.

- *Gabriel graph approximation and Representative subsets* Regarding the proposal that extracts representative subsets, a different approach that could be investigated is the use of the Algorithm 4 with the Gabriel graph, to create a approximation of the result obtained, when both methods are used with an entire data set. The investigation would be to essay an approximation with a grid that separates the points of the data set, in subsets according to an equally divided space of the data domain, and posterior evaluation of each subset separately, taking the final extraction as the union of the separated solutions. Such approach is aligned with the fundamental characteristic of the Gabriel graph structure, which preserves local relation of the nodes connection. This action would resulting in a lower amount of computation required.

- *Different types of graph* - The present work proposed the used of an independent dominating set that is induced from the Gabriel graph. An possible investigation would be the inducement of the dominating set from other type of graph, i.e. the Delaunay triangulation. A different type of graph exhibits a distinct nodes connections, which results in a different dominating set, when induced. This particularity may lead to a classifier with a decision surface with distinct characteristics than the obtained in this thesis.

- *Classifier of a Deep Neural Network* - Neural Networks with deep architectures, like Convolutional Neural Networks, usually presents a sequence of filters that extract relevant features from the inputs to be classified by a shallow classifier in the output of the network. Some approaches for training this type of networks proposes to separate the parameters estimation of the filters from the classifier in the output [da Silva Ribeiro, 2021]. One possible research topic would be the use of the Algorithm 4 to extract a representative subset of the output of the filters, already trained, and the use of them with an KDE estimator as the classifier in the output of the network. Possibly, some dimensional reduction method would complement the solution.

- *Imbalance data sets* - The classification task is subject to process data sets with different number of points in each class presented. Such issue with the classes probabilities is

known as Class Imbalance [He and Garcia, 2009], and tends to result in a model with a bias towards the majority class, when classification is done with a regular classifier. One possible research topic would be the use of Algorithm 4 to under-sample the majority class and to induce a classifier from this subset. Another topic would be assessment of the proposed KDE Estimator as a classifier, once initial tests that were conducted, showed that the relation between its parameters (number of dominating points / Radial functions and radius) seems to be positive when treating imbalance data sets. Regarding the Class Imbalance in the online learning context, the different probabilities of the classes occurrence along the sequence of a data stream may be even harder, when manifested in conjunction with the concept drift, due to the difficulty of treating both incidents simultaneously [Ditzler and Polikar, 2013]. Once the proposed online KDE Estimator has shown some robustness against the class imbalance and was able to deal with the concepts drifts, in the tests of section 3.6.1, the research of this classifier in the non-stationary scenarios with imbalance data streams would to be other possible topic.

- *Refinements of the algorithms* - The Algorithm 1, which implement the steps from the proposed approach of online updating of the Gabriel graph, was implemented without the intention to be simulated in a GPU. Indeed, only 1 tread of a computer with 2 cores (2 treads in each core) was used in the tests from sections 3.6 and 4.5.2. Thus, a natural increment of this implementation would be a new code conceived for GPU, aiming a faster processing. Other possible refinement for the Algorithm 1 is its implementation with a separator grid that would enable the graph construction to be done separately for each resulting subset of points. Then, the final graph would be assembled using the graphs previous induced.

- *Incremental Learning (Application of the proposed methods to real problems)* - As done in the test using a data set from a industrial process (section 4.5.2), the one-class RBF network and the Algorithm 4 could be applied to other real problems, that is represented by big data sets. One interest feature of their use in conjunction, that constitute a research topic, is the possibility to treat conditions in which the positive class may drift occasionally, resulting in new activated regions in the space domain. The use of a predictive model in this case possible would need an adaptation or a retraining to cope with the new scenario. In this case, the Algorithm 4 could be used to select a representative subset from a set of points, representing the new condition, while the proposed one-class RBF network

could assimilate this knowledge, incorporating new Radial functions, in a incremental approach. Thus the RBF network could sustain its knowledge, while plasticity to learn the new patterns would be provided.

- *Online one-class classifier* - Although the dominance approach was used here to define, in offline mode, the one-class RBF network, and the online KDE estimator for classification of data sets with two or more classes, an interesting research topic would be to extend its use in Online Learning with one-class classifiers. Particularly for models which uses RBFs in its definition and require continuous update of the parameters.

6.3 Publications

The following work were published during the research of this thesis:

- Alvarenga, W. J., Campos, F. V., Hanriot, V. M., Gonçalves, E. B., Costa, A. C., Araujo, L. R., Magalhães, E., and Braga, A. P. (2021). Online learning of neural networks using random projections and sliding window: A case study of a real industrial process. *Engineering Applications of Artificial Intelligence*, 100:104181.

- Queiroz, M., Coelho, F., Torres, L. C. B., Campos, F. V., Lara, G., Alvarenga, W., and Braga, A. P. (2022). RBF neural networks design with graph based structural information from dominating sets. *(under review)*.

- Ushikoshi, T. A., Freitas, E. J. R., Menezes, M., Alvarenga, W., Torres, L. C. B., and Braga, A. P. (2022). Hebbian Learning with Kernel-based Embedding of Input Data. *Neural Computing and Applications*, (to be published)

Bibliography

Allan, R. B. and Laskar, R. (1978). On domination and independent domination numbers of a graph. *Discrete mathematics*, 23(2):73–76.

Allan, R. B., Laskar, R., and Hedetniemi, S. (1984). A note on total domination. *Discrete Mathematics*, 49(1):7–13.

Alvarenga, W. J., Campos, F. V., Hanriot, V. M., Gonçalves, E. B., Costa, A. C., Araujo, L. R., Magalhães, E., and Braga, A. P. (2021). Online learning of neural networks using random projections and sliding window: A case study of a real industrial process. *Engineering Applications of Artificial Intelligence*, 100:104181.

Araujo, A. F. and Rego, R. L. (2013). Self-organizing maps with a time-varying structure. *ACM Computing Surveys (CSUR)*, 46(1):1–38.

Arias-Garcia, J., Mafra, A., Gade, L., Coelho, F., Castro, C., Torres, L., and Braga, A. (2020). Enhancing performance of gabriel graph-based classifiers by a hardware co-processor for embedded system applications. *IEEE Transactions on Industrial Informatics*, 17(2):1186–1196.

Assis, A. D., Torres, L. C., Araújo, L. R., Hanriot, V. M., and Braga, A. P. (2021). Neural networks regularization with graph-based local resampling. *IEEE Access*, 9:50727–50737.

Bhattacharya, B. K., Poulsen, R. S., and Toussaint, G. T. (1981). Application of proximity graphs to editing nearest neighbor decision rule. In *International Symposium on Information Theory, Santa Monica*.

Bifet, A. and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM.

Bifet, A. and Gavalda, R. (2009). Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, pages 249–260. Springer.

Bose, P., Collette, S., Hurtado Díaz, F. A., Korman, M., Langerman, S., Sacristán Adinolfi, V., and Saumell Mendiola, M. (2010). Some properties of higher order delaunay and gabriel graphs. In *22nd Canadian Conference on Computational Geometry*, pages 13–16.

Broomhead, D. S. and Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom).

Chellali, M., Favaron, O., Hansberg, A., and Volkmann, L. (2012). k-domination and k-independence in graphs: A survey. *Graphs and Combinatorics*, 28(1):1–55.

Claesen, M. and De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*.

Cockayne, E. J., Dawes, R., and Hedetniemi, S. T. (1980). Total domination in graphs. *Networks*, 10(3):211–219.

Cockayne, E. J. and Hedetniemi, S. T. (1976). Disjoint independent dominating sets in graphs. *Discrete Mathematics*, 15(3):213–222.

Costa, B. S. J., Angelov, P. P., and Guedes, L. A. (2015). Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier. *Neurocomputing*, 150:289–303.

Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14 Issue 3(3):326–334.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

da Silva Ribeiro, E. (2021). *LDMAT: função de custo baseada em matrizes de distâncias para o treinamento de redes neurais convolucionais*. PhD thesis, Universidade Federal de Minas Gerais.

Deng, W., Zheng, Q., and Chen, L. (2009). Regularized extreme learning machine. In *2009 IEEE symposium on computational intelligence and data mining*, pages 389–395. IEEE.

Ditzler, G. and Polikar, R. (2011). Hellinger distance based drift detection for nonstationary environments. In *2011 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE)*, pages 41–48. IEEE.

Ditzler, G. and Polikar, R. (2012). Incremental learning of concept drift from streaming imbalanced data. *IEEE transactions on knowledge and data engineering*, 25(10):2283–2301.

Ditzler, G. and Polikar, R. (2013). Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2283–2301.

Ditzler, G., Roveri, M., Alippi, C., and Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25.

Duda, R. O., Hart, P. E., et al. (2000). *Pattern classification*. John Wiley & Sons.

Elwell, R. and Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Fritzke, B. (1994). Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural networks*, 7(9):1441–1460.

Fritzke, B. (1997). A self-organizing network that can follow non-stationary distributions. In *International conference on artificial neural networks*, pages 613–618. Springer.

Gabriel, K. R. and Sokal, R. R. (1969). A new statistical approach to geographic variation analysis. *Systematic zoology*, 18(3):259–278.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58.

Goddard, W. and Henning, M. A. (2013). Independent domination in graphs: A survey and recent results. *Discrete Mathematics*, 313(7):839–854.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. Johns Hopkins University Press.

Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

Haynes, T. W., Hedetniemi, S., and Slater, P. (1998). *Fundamentals of domination in graphs*. CRC press.

He, H. and Garcia, E. A. (2009). Learning from imbalance data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

Henning, M. A. (2009). A survey of selected recent results on total domination in graphs. *Discrete Mathematics*, 309(1):32–63.

Henning, M. A., Pal, S., and Pradhan, D. (2020). Algorithm and hardness results on hop domination in graphs. *Information Processing Letters*, 153:105872.

Hoi, S. C., Sahoo, D., Lu, J., and Zhao, P. (2021). Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289.

Horta, E. G. and de Pádua Braga, A. (2014). An extreme learning approach to active learning. In *ESANN*. Citeseer.

Huang, G.-B., Chen, L., Siew, C. K., et al. (2006a). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Networks*, 17(4):879–892.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, volume 2, pages 985–990. IEEE.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006b). Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501.

Japkowicz, N., Myers, C., Gluck, M., et al. (1995). A novelty detection approach to classification. In *IJCAI*, volume 1, pages 518–523. Citeseer.

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

Klinkenberg, R. and Joachims, T. (2000). Detecting concept drift with support vector machines. In *ICML*, pages 487–494.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69.

Kolter, J. Z. and Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790.

Liang, N.-Y., Huang, G.-B., Saratchandran, P., and Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, 17(6):1411–1423.

Matula, D. W. and Sokal, R. R. (1980). Properties of gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographical analysis*, 12(3):205–222.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Menezes, M. V., Torres, L. C., and Braga, A. P. (2019). Width optimization of rbf kernels for binary classification of support vector machines: A density estimation-based approach. *Pattern Recognition Letters*, 128:1–7.

Modenesi, A. P. and Braga, A. P. (2009). Analysis of time series novelty detection strategies for synthetic and real data. *Neural processing letters*, 30(1):1–17.

Natarajan, C. and Ayyaswamy, S. (2015). Hop domination in graphs-ii. *An. Stt. Univ. Ovidius Constanta*, 23(2):187–199.

Parra, L., Deco, G., and Miesbach, S. (1996). Statistical independence and novelty detection with information preserving nonlinear maps. *Neural Computation*, 8(2):260–269.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Pimentel, M. A., Clifton, D. A., Clifton, L., and Tarassenko, L. (2014). A review of novelty detection. *Signal processing*, 99:215–249.

Queiroz, M., Coelho, F., Torres, L. C. B., Campos, F. V., Lara, G., Alvarenga, W., and Braga,

A. P. (2022). Rbf neural networks design with graph based structural information from dominating sets. *Under review*.

Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285.

Rosenblatt, M. (1956). Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sack, J.-R. and Urrutia, J. (1999). *Handbook of computational geometry*. Elsevier.

Schmidt, W. F., Kraaijveld, M. A., Duin, R. P., et al. (1992). Feed forward neural networks with random weights. In *International Conference on Pattern Recognition*, pages 1–1. IEEE COMPUTER SOCIETY PRESS.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

Shao, Z. and Er, M. J. (2016). An online sequential learning algorithm for regularized extreme learning machine. *Neurocomputing*, 173:778–788.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*, volume 26. CRC Press.

Silvestre, L. J., Lemos, A. P., Braga, J. P., and Braga, A. P. (2015). Dataset structure as prior information for parameter-free regularization of extreme learning machines. *Neurocomputing*, 169:288–294.

Su, P. and Drysdale, R. L. S. (1997). A comparison of sequential delaunay triangulation algorithms. *Computational Geometry*, 7(5-6):361–385.

Sun, P. G., Ma, X., and Chi, J. (2017). Dominating complex networks by identifying minimum skeletons. *Chaos, Solitons & Fractals*, 104:182–191.

Takahashi, C. C. and Braga, A. P. (2020). A review of off-line mode dataset shifts. *IEEE Computational Intelligence Magazine*, 15(3):16–27.

Tax, D. M. and Duin, R. P. (2004). Support vector data description. *Machine learning*, 54(1):45–66.

Torres, L., Castro, C., Coelho, F., Torres, F. S., and Braga, A. (2015a). Distance-based large margin classifier suitable for integrated circuit implementation. *Electronics Letters*, 51(24):1967–1969.

Torres, L., Coelho, F., Castro, C. L., and Braga, A. P. (2014). A graph of gabriel approach for large margin classifiers. In *LA-CCI-The Latin American Congress on Computational Intelligence Co-located with ARGENCON*, volume 1, page 55.

Torres, L. C., Castro, C. L., and Braga, A. P. (2015b). Gabriel graph for dataset structure and large margin classification: A bayesian approach. In *Proceedings of the European Symposium on Neural Networks*, pages 237–242.

Torres, L. C., Castro, C. L., Coelho, F., and Braga, A. P. (2020). Large margin gaussian mixture classifier with a gabriel graph geometric representation of data set structure. *IEEE transactions on neural networks and learning systems*, 32(3):1400–1406.

Vapnik, V. (1992). Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer.

Varsa, G. S. and Sridharan, D. (2019). A balanced energy efficient virtual backbone construction algorithm in wireless sensor networks. *AEU-International Journal of Electronics and Communications*, 107:110–124.

Wanderley, M. F. B., Gardeux, V., Natowicz, R., and de Pádua Braga, A. (2013). Ga-kde-bayes: an evolutionary wrapper method based on non-parametric density estimation applied to bioinformatics problems. In *ESANN*. Citeseer.

Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101.

Wu, W., Du, H., Jia, X., Li, Y., and Huang, S. C.-H. (2006). Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theoretical Computer Science*, 352(1-3):1–7.

Yu, H. and Webb, G. I. (2019). Adaptive online extreme learning machine by regulating forgetting factor by concept drift map. *Neurocomputing*, 343:141–153.

Zhang, W. and King, I. (2002). A study of the relationship between support vector machine and gabriel graph. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 1, pages 239–244. IEEE.