eman ta zabal zazu

Universidad del País Vasco | Euskal Herriko Unibertsitatea

# Advances in Streaming Novelty Detection

by

Ander Carreño

Supervised by Iñaki Inza and Jose A. Lozano

Donostia - San Sebastián, September 2022

*In God we trust; all others bring data.*
–W. Edwards Deming

This research was carried out at the University of the Basque Country (UPV/EHU) within the Intelligent Systems Group (ISG) and has been partially supported by the Basque Government (IT1504-22, IT1244-19, Elkartek, BERC 2018-2021 and 2022-2025 programs), and by the Spanish Ministry of Science, Innovation and Universities trough the BCAM Severo Ochoa accreditation SEV-2017-0718, PID2019-104966GB-I00, TIN2016-78365-R. Ander Carreño has been supported by a grant of the Spanish Ministry of Economy and Competitiveness (BES-2017-080016).

## Acknowledgments

I would like to start expressing my appreciation to my supervisors, Iñaki Inza and Jose A. Lozano for their patience and efforts during my Ph.D. This work would not have been completed without their wise guidance. Especially, I would like to point out the vast number of times Iñaki has called me to just ask about my spirit. It has been exceptional support throughout the entire thesis, particularly, in the most stressful moments. Besides, I will never forget some of the sentences that Jose A. has written me these years in his review comments while proofreading papers: *Por aquí pasó María dejando la porquería*, *efecto globo...* or *conejo de la chistera*. Thank you for helping me become the scientist I am today.

I am also taking the opportunity to thank Jesse Read for giving me the outstanding opportunity to share my ideas and collaborate with him and his colleagues as a visiting researcher in the Laboraitoré d'Informatique of the École Polytechnique. Our distended talks have resulted in great contributions that have suppose a step up in my scientific career. I am not only bringing good research but what I would like to be a lifetime relationship. All the best for the future!

The number of people that have supported me during my endeavors as a Ph.D. student could require the length of this dissertation by just trying to thank them all. Hence, I am forced to do a small selection of them. Firstly, I dedicate this thesis to who deserves every drop of ink put in this document, Nerea Martin. I could never have achieved this without you. You have illuminated me in the darkest moments, toasted with me in happiest ones, and encouraged me to keep pushing until the end. Everything I can say is not even close to what you really deserve. The vast part of this thesis is yours. Thank you very much!

I would like to specially thank Josu Ceberio for his incredible support even before this Ph.D started. You have planted the seed of science in me. Thanks for the Skype talks, beers, and astonishing *poteos*. Please, never lose that unique sense of humor.

I could not continue this document without sincerely thanking Verónica Álvarez, Onintze Zaballa, Amaia Abanda, Anton Uranga, and Ioseba I. Alonso. You have served me as pillars throughout these years and I would never have achieved this milestone without you. Thanks for the coffee breaks, lunch times, sports sessions, and for listening when I really needed it. You deserve the best!

I wish to extend my appreciation to Cristina Galán, Jairo Rojas-Delgado, Ioar Casado, Lorenzo Nagar, Martin Parga, Santiago Mazuelas, Leticia Hernando, Aritz Perez, Ekhiñe Irurozki, Fabio Pizzichillo, Borja Calvo, and Jose Antonio Pascual for diversely helping me along this way.

Above all, I would like to give thanks to people who did not scientifically contribute to the dissertation but were my companions in this prolonged journey. Isa, Lucia, Nerea, and Aritz, thanks for helping me clear my mind in some necessary moments when I would lose hope. You have always tried to understand me and tried to shed some light on the foggy pathways.

*Quisiera dar las gracias a mi familia: Fransico Javier Carreño, María Luisa Tobes, Asier, Feli, y Fernando por vuestro apoyo y cariño incondicional, por haberme educado en la perserverancia, la cultura del estudio, del esfuerzo y del trabajo, y por haberos preocupado por mi en los momentos difíciles. Puede que vosotros os sintáis orgullosos de mi, pero no es comparable a cómo yo os admiro y os agradezco que hayáis estado y estéis ahí. Parte de esta tesis es también vuestra. No cabe duda de que también debo agradecer mucho a mi otra familia: Carlos, Tere e Iker. Gracias por haber sido mi apoyo todos estos años.*

As I said, I have had the difficult task of selecting people who explicitly appear in this section. Nevertheless, if you feel that you should have been here and you are not, I sincerely apologize, and please, take a part of this last Thank You.

# Contents

# 0

# Preface

Almost a century ago, *machine learning* started its endeavors to become one of the most intensively researched fields that is today. In its source, it gathered mathematicians, physicists, and statisticians that worked on pattern recognition and artificial intelligence. Their aim was to build a mathematical model that could learn from and make predictions on data.

Nowadays, due to the massive capacity of data storage and process capabilities; along with the impressive results that machine learning and artificial intelligence fields have obtained, machine learning is one of the most competitive fields of science with more than *a million* papers published since this thesis started in 2018. In this social and scientific context, this dissertation tries to provide a humble contribution to the overwhelming state-of-the-art.

In supervised classification, there are some input variables that have influence over other output variables. The aim is to build a statistical model that learns such influence so that it can predict the output given the input. Commonly, the literature knows the inputs as features, attributes or independent variables; and the outputs as classes, labels or targets [Hastie et al., 2009, Bishop, 1995].

Suppose building a production chain where screws are being manufactured. The raw material would flow throughout several steps until the final product, the screw, is completed. In several different steps along the chain, imagine having devices that analyze the process of every single screw by means of a variety of sensors. With sensor data (input), some checkpoints can be created so that anomalous, abnormal, rare, or outlier (output) screws can be eliminated from the chain by an automatic, intelligent system (statistical model). As a result, an increase in production and a decrease in costs could be achieved. In order to create such convenient model, labeled data is needed. This means that, somehow, the inputs and the outputs must be collected into a dataset, so that the correlation can be properly learned. Such intelligent system would be built under the supervised classification paradigm.

The mentioned example is one of the vast majority of applications of machine learning and supervised classification that is widely approved by both social and scientific communities. Essentially, in supervised classification data is composed by multiple examples or instances. Each training instance is defined by a set of features and its associated labels or targets. In this general context, several situations can occur that drastically change the task of learning a classifier. For instance, sometimes labeled data is available to learn a statistical model. In other situations, due to the nature of the problem, the most common case is to predict that an screw is being normally/properly manufactured. Often, the input data is not complete due to some errors in the sensors. Once in a while, there is a type of time correlation between the input variables; for instance, if the temperature of the manufactured screw is measured throughout the production chain. The model must consider that the tools used in the manufacturing process suffer from wear, potentially modifying the correlation between the input and output variables... Nevertheless, correctly defining such problems is a crucial task for progress. The first contribution of this dissertation specifically focuses on this. The key differences among some close, but divergent problems that have been indistinctly referred to with same terminology, hindering the advance of the field have been provided.

As a second contribution, one relevant and challenging problem that gives name to this dissertation has been studied. Concretely, the Streaming Novelty Detection (SND) problem. In this learning scenario, it is considered that new classes can emerge or disappear in a stream fashion. The problem starts from a given set of classes that evolves during time. The task of the classifier is to accurately provide prediction for the unseen instances considering that new classes can emerge or disappear. Furthermore, the generative distribution of the instances can change so the model needs to adapt to these changes.

Few approaches that deal with the SND can be found in the literature [Faria et al., 2016, Carreño et al., 2022, Masud et al., 2013, Mu et al., 2017]. Commonly, these have followed a similar framework that consist of learning an initial model from a given supervised dataset. Then, new cases will timely arrive for prediction. Suppose that one of the samples is widely different from the ones that the model is expecting/has learned. The model would recognize this and store it into a fixed-sized buffer. When there is a sufficient amount of cases i.e., when the buffer is full, the model would automatically seek for new emerging classes among such abnormal instances. At this point, the model is updated to consider new emerging classes and the predictions of the buffered instances are output.

Relating this scenario to the brutal COVID-19 pandemic that has struck the entire world between 2019 an 2021, imagine having a system capable of predicting the diseases of a patient based on data such as blood test, electrocardiograms (ECG) and X-ray images. A model would have been built with data of both healthy people, and patients with common cold, flu, chickenpox and mononucleosis. Such model could classify patients among this set of classes. When some COVID-19 positive patient is analyzed by the model, the system would recognize this as an abnormal case and, after some time, when a sufficient number of abnormal cases are gathered, the system would automatically have discovered COVID-19 disease.

In particular, we have provided a probabilistic modeling approach for the SND problem based on mixture of Gaussian distributions. For each class, a Gaussian distribution is learned and the newcomer instances are predicted based on the probability of belonging to each of the classes. When there is not enough evidence to classify an instance among the previously learned set of classes, it is stored into a fixed-sized buffer. When the buffer is full, new classes are sought among the buffered instances by means of the Expectation Maximization (EM) algorithm.

As a third contribution, the previous work is extended to treat different type of data, and to account for some of the inherent limitations of the SND problem. In particular, the case where data are time series is studied, meaning that there is temporal correlation among the input variables. Furthermore, since the discovering step is done in an unsupervised manner, the model could get into a non-recoverable state if a new class concept is wrongly identified in the discovery process. To overcome this critical issue, we propose to maintain multiple parallel, inherently different models that an expert could evaluate in hindsight.

I hope you enjoy reading the forthcoming pages about such fascinating ideas and contributions as much as I have delighted in exploring these years.

## 0.1 Overview of the Dissertation

This document is organized as follows: Chapter 1 provides a self-explanatory introduction to a variety of base concepts necessary for understanding the following contributions. Particularly, the task of classification is described in Section 1.2. In Section 1.3, the role of *time* when learning classifiers is deeply discussed. Firstly, the relation of time among the features is treated. Secondly, the data that is timely generated is reviewed. Section 1.4 and Section 1.5 describe some literature techniques that learn a model in supervised and unsupervised learning scenarios, respectively. Afterwards, the methodologies to assess the validity of the models are exposed in Section 1.6, and Section 1.7 shows multiple scores that summarize the performance of a model.

In the following chapters, the main contributions that conform this dissertation are described. Chapter 2 focuses on a literature review of some divergent problems that are referred to with same terminology in the literature and vice-versa. Chapter 3 and 4 consist of two methodological contributions to the SND problem. The sooner describes a novel parametric solution based on a mixture of Gaussian distributions while the later is based on a deep neural network approach that considers time series data.

# 1 Background

This chapter introduces the general notation and framework used in the rest of the dissertation. It starts by defining the classification task, followed by the different learning scenarios where the classifiers are learned. Consequently, a discussion about the role of time when learning classifiers is given. Finally, multiple algorithms that learn a classifier, and the evaluation methods and scores are summarized.

## 1.1 The Task of Classification

In machine learning, classification refers to the problem of predicting a set of categorical outputs (classes or labels), given a set of inputs (features). Formally, let $\mathcal{X} \subseteq \mathbb{R}^d, \mathcal{C} \subseteq \mathbb{N}^k$ be the feature and label spaces, respectively; where $d$ represents the number of features and $k$ is the number of classes. Also, let $\mathbf{x} \in \mathcal{X}$ be a feature vector and $\mathbf{c} \in \mathcal{C}$ its corresponding label vector. A labeled instance is then defined as the tuple $(\mathbf{x}, \mathbf{c})$ that is assumed to be generated by a probability distribution $p(\mathbf{x}, \mathbf{c})$ [Mitchell, 1997, Duda et al., 2001]. A classifier consists of learning a function $f$ that maps features to labels defined as $f : \mathcal{X} \rightarrow \mathcal{C}$.

Depending on the length of the label vector $\mathbf{c}$, several major classification problems can be defined. These can be categorized in two groups, firstly, single-output classification problems, where $k = 1$ can be considered. Secondly, multi-dimensional or multi-output classification problems where $k > 1$ are found. Although there are many multi-output classification problems [Read et al., 2011, 2014], single-output classification has attracted more research [Faouzi, 2022, Sellami and Tabbone, 2022]. Partially, because a multi-output classification problem can be dealt with single-output classification approaches [Rivolli et al., 2020].

Considering the number of values each class variables can take, different categorizations can be made. When $k = 1$ and the number of values is two, scenarios where the class variable $c$ is binary are found; while, on the other hand, problems where $c$ takes more than two values exist. In the sooner, the problem is known as binary classification while the latter problem is named as multiclass classification. By combining these two categorizations, 4 different classification paradigms are found that are summarized in Figure 1.1. In this dissertation, the scenario where there is only one class variable and it is binary is treated in the first contribution shown in Chapter 2. Chapter 3 and 4 assume the multiclass setting. Henceforth, a labeled instance is defined as the tuple $(\mathbf{x}, c)$.

According to the Bayesian Decision Theory [Duda et al., 2001], classification can be described by the prior probabilities of the classes $p(c)$ and the class conditional probability density functions $p(\mathbf{x}|c)$. The classification decision is made according to the posterior probabilities of the classes, which for class $c$ can be represented as:

$$p(c|\ \mathbf{x}) = \frac{p(c)p(\mathbf{x}|\ c)}{p(\mathbf{x})}, \tag{1.1}$$

where $p(\mathbf{x}) = \sum_c p(c)p(\mathbf{x}|c)$. Assuming that there is an equal cost of misclassification between the different classes, this classification rule is optimal. The prediction of an instance $\mathbf{x}$ is done as $\hat{c} = \arg\max_{c\ \in \mathcal{C}} p(c|\mathbf{x})$.

In supervised classification, the classifier learns the relation between the features $\mathbf{x}$ and the class $c$ by minimizing the expectation of a loss function over a set of examples. The most common loss is the 0/1 loss function that is defined as:

$$L(c, \hat{c}) = \begin{cases} 1 \text{ if } c \neq \hat{c} \\ 0 \quad \text{otherwise} \end{cases} \tag{1.2}$$

assuming that $c$ is the true label of the instance $\mathbf{x}$.

In this classification problem the data to learn the classifier is annotated. However, there are situations where obtaining labeled data is not

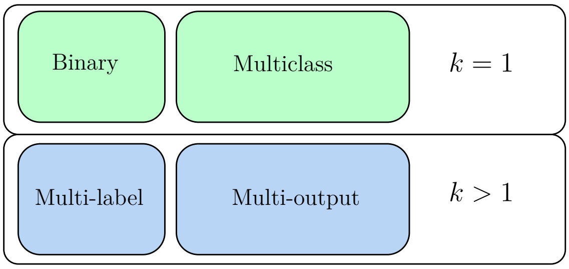


Fig. 1.1: Summary of the different classification problems with respect to the structure of the class variable.

possible. This issue derives into some different classification paradigms that are discussed in the following section.

## 1.2 Main Learning Scenarios

In machine learning data is the key component that allows learning predictive models. This data is sometimes annotated, meaning that the features are accompanied with their corresponding classes. However, gathering this annotated data is not always possible due to time or cost limitations. Therefore, three different dataset configurations with respect to the class variable can occur, resulting in three different problems.

- In **supervised classification** a fully labeled dataset is available, defined as $\mathcal{D}_s = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_n, c_n)\}$ so that $\mathcal{D}_s \subseteq \mathcal{X} \times \mathcal{C}$. Therefore, in supervised classification, an instance $\mathbf{x}_i$ is accompanied with its class value $c_i$, for $i = 1, 2, \dots, n$ .
- In **unsupervised classification** no labeled data is available to learn the model. Therefore, the dataset is defined as $\mathcal{D}_u = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$. The task in this problem consists of finding groups (clusters) of instances in the provided dataset.
- Somewhat in between the aforementioned learning scenarios, the **weakly-supervised classification** problem is found [Hernández-González et al., 2016]. This setting refers to the lack of complete supervision in the dataset. There exists some label information but

it is not complete. From this, the popular semi-supervised problem can be considered that assumes that there is a mix of labeled and unlabeled examples in the given dataset $\mathcal{D}_{sm} = \{\mathcal{D}_s \cup \mathcal{D}_u\}$. Nevertheless, weakly-supervised classification refers to a more general framework where for instance, a set of candidate labels are provided for a group of instances but the one-to-one correspondence is not defined, or to problems where multiple subjective labels are supplied for each instance by a group of non-expert annotators (crowd learning).

As the reader might has been realized already, the number of data samples available, the distribution of instances of each of the classes, the amount of the supervised information in a weakly-supervised setting, or if all the classes are represented in the provided dataset for learning, is also related to the task of learning a classifier. Referring to these characteristics enlarges the described taxonomy.

## 1.3 The Role of Time When Learning Classifiers

Data is constantly being generated. In 2022, more than $2.5 \times 10^{18}$ bytes per day have been created. Unsurprisingly, such data comes in a wide variety of shapes and formats that range from images and table-shaped data to sounds and handwritten texts. Furthermore, nowadays the fact that the data is constantly being generated may imply that there is a temporal relation between such data.

*Time* is a key component that needs to be reviewed to understand the forthcoming pages. On the one hand, the influence of *time* over the measured features is discussed. On the other hand, the timely arrival of instances in an online or stream setting is reviewed.

### 1.3.1 Influence of Time Among the Features

In this section, two situations are reviewed. On the one hand, data that does not have a temporal correlation is considered. Formally, an instance is defined as $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ where $x_v \in \mathbb{R}$ and $v \in \{1, 2, \ldots, d\}$. Let $\sigma$ be the permutation operator so that outputs a permutation of a given vector and $D_\sigma = \{(\sigma(\mathbf{x}_1), c_1), (\sigma(\mathbf{x}_2), c_2), \ldots, (\sigma(\mathbf{x}_n), c_n)\}$ be the resulting dataset after applying $\sigma$ to each of the instances of a given dataset $D$. Let $f$ also be the classifier learned from $D$ and $h$ a classifier learned in $D_\sigma$. When there is no temporal correlation among the variables, the learned classifiers are the same $f = h$. This suggests that changing the order of the features of an instance *does not change what it represents*. For example, in a blood test of a patient, the measured values like cholesterol, glucose or sodium do not have any temporal correlation. Moving the percentage of lymphocytes to the bottom of the document does not alter what the blood test represents. Hence, the resulting classifier is the same no matter the order of the features.

On the other hand, data that is timely measured is found. This data is often referred to as *time series* [Abanda et al., 2019]. In this case, there is a temporal correlation among the measured features. Formally, an instance is defined as $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ where $x_v \in \mathbb{R}$ and $v \in \{1, 2, \ldots, d\}$. Regular supervised classification approaches do not leverage from the temporal correlation. However, it has been shown that considering time is crucial for time series classification [Faouzi, 2022, Abanda et al., 2019, Ghassempour et al., 2014, Oates et al., 1999]. Considering that we use algorithms that do leverage from temporal correlation, following the same idea as in the previous case, $f \neq h$. This suggests that, for instance, if a wind sensor timely monitors the wind speed over a day, the measured values have a temporal correlation, a temporal order. Moving the wind record of 3 p.m. to after 8 p.m. drastically changes what the instance represents. Note that, time series may have an infinite length [Abanda et al., 2019]; however, our definition only considers fixed-length time series since these are the ones that are used in this dissertation.

### 1.3.2 Timely Generated Instances

Similar to the previous section, two scenarios are considered if the instances are timely generated or not.

Firstly, when the instances are not timely generated, we refer to this scenario as an static or standard classification problem. In this setting it is assumed that the data comes from a generative distribution that does not change over time. Therefore, the training set provided to learn a predictive model comes from the same probability distribution as the forthcoming test set.

Secondly, considering that instances timely arrive, the learning scenario drastically changes. This scenario is also known in the literature as *online* or *stream* learning [Lu et al., 2019, Alippi et al., 2017, Sun et al., 2016, Gomes et al., 2017, Bifet et al., 2018].

In this learning scenario, instances are timely being generated from a generative distribution that may change over time. In Figure 1.2, an illustration of how instances are generated in a streaming environment is found.

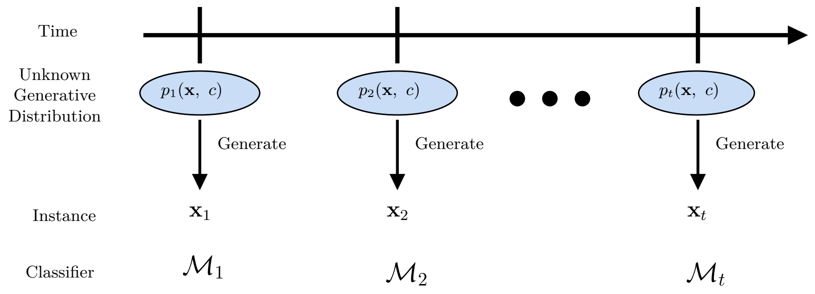


Fig. 1.2: Graphical representation of how the instances are timely generated in a streaming classification learning scenario.

In a screw manufacturing process where raw material is transformed until a screw is obtained, the same sensors record data of screws that are timely being manufactured. Such sensors clearly suffer from wear over time. Furthermore, the raw material used to manufacture the screws may slightly change. This motivating example clearly shows that the joint generative probability distribution $p(\mathbf{x}, c)$ can change over time, occurring the well known **concept drift** [Gama et al., 2014]. Such drift stands for unexpected, and unpredictable changes in the underlying generative distribution of the data. Formally, given two different timestamps $t_1$ and $t_2$, the probability distribution of the data changes $p_{t_1}(\mathbf{x}, c) \neq p_{t_2}(\mathbf{x}, c)$. Although one might think that such concept drift is constrained to be smooth, meaning that the changes from $t_1$ to $t_2$ are small, major changes can also be expected. For example, when a sensor is broken and it needs to be replaced; and the newcomer one is from a different brand (change in $p(\mathbf{x})$). Therefore, the classification process becomes more challenging since the classifier needs to dynamically adapt to these possible changes in order to maintain accurate predictions.

The source of the concept drift has been thoroughly researched [Gama et al., 2014], and it can be summarized as changes in the components of Equation 1.1. In other terms,

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

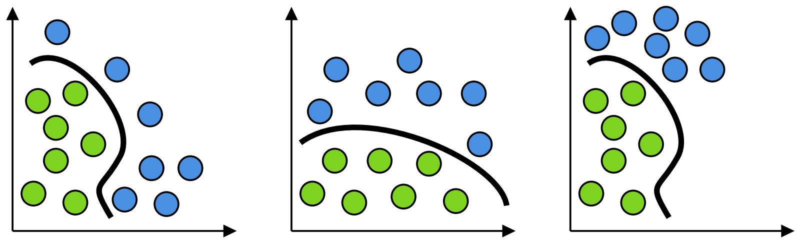

(a) Original data (b) Real concept drift (c) Virtual concept drift

Fig. 1.3: Different types of concept drift according to the relation between the features and the class variable.

For the task of learning a classifier, these changes are relevant if they affect to the decision boundary of the classifier. Such changes can be graphically seen in Figure 1.3. If these changes affect to the decision boundary, the classifier would need to adapt in order to keep providing accurate predictions (see Figure 1.3b). Hence, two different types of concept drift can be defined [Gama et al., 2014]:

- *Real concept drift* refers to the changes in $p(c|\mathbf{x})$. These changes can occur with or without changes in $p(\mathbf{x})$ and they force to adapt the classifier. As can be seen in Figure 1.3b, the decision boundary needs to be adapted to properly classify the instances.

- *Virtual concept drift* happens when there is a change in $p(\mathbf{x})$ but without a modification in $p(c|\mathbf{x})$. Hence, the decision boundary remains valid and the adaptation is not required. Figure 1.3c shows that the location of the instances has changed but there is no need to modify the decision boundary to keep providing accurate predictions.

## 1.4 Learning From Supervised Data

In this section, the most common supervised classification approaches of the literature that will take part in the forthcoming pages are reviewed.

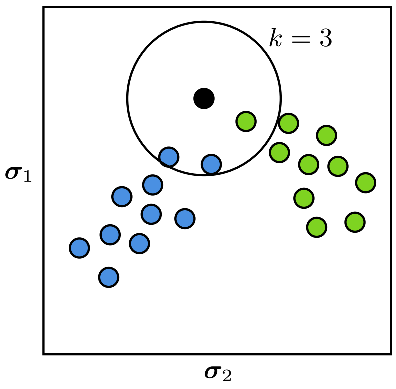


Fig. 1.4: Illustration of a $k$-Nearest Neighbors ($k$-NN) algorithm with $k = 3$ using Euclidean distance.

The most popular algorithm that fits into this category is the **$k$-Nearest Neighbors ($k$-NN)**. It is a distance based classifier that is usually referred to as a lazy-classifier since it does not learn any model from the data. Instead of learning a model from the data, it directly uses the training set to predict the label of unseen samples. It is based on the assumption that inputs that are close in the feature space, have close outputs in the label space. Defining *closeness* leads to different implementations of this algorithm. The most common similarity measure is the Euclidean distance. Figure 1.4 shows an illustration of a $k$-NN algorithm with $k$ set to 3 neighbors. As can be seen, the new instance is predicted as *blue* class since the majority of its $k = 3$ closest neighbors are from such class. Different values of the $k$ parameter may result on different predictions for the same instance. Intuitively, a low $k$ value imposes a more rigid decision boundary. On the contrary, having a large $k$ value derives into a smoother decision boundary. This is related to another widely studied problem called over fitting [Dietterich, 1995].

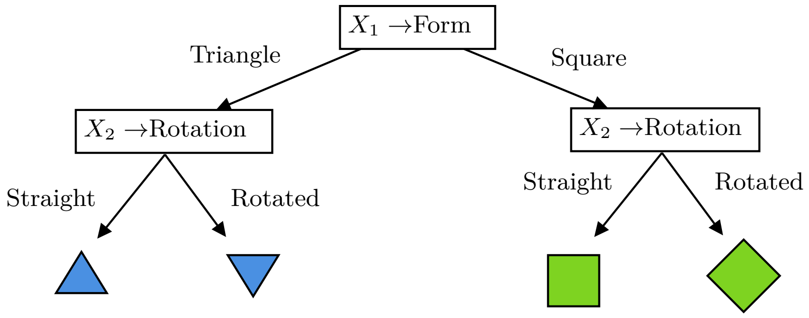


Fig. 1.5: Illustration of a decision tree.

There are methods, founded in information theory [Shannon, 1948], that consist of finding split values of the features so that they can properly discriminate between labels. The most common approach is the **decision tree**. It learns a variety of split points for the features in a tree-like model that are then used to discriminate the instances into the different classes. C4.5 [Quinlan, 2014] is the most common algorithm that learns a decision tree. The nodes in a decision tree are features and, in the branches, the corresponding split values are shown. Figure 1.5 shows an example of a decision tree. Two attributes are used to discriminate between 4 classes. Decision trees offer the main advantage of interpretation since the tree-like model is easy to understand. However, decision trees tend to offer unstable classifiers. A change in the data could imply a major change in the structure of the tree. In order to account for this limitation, the ensemble based methods have been proposed. The most popular one is the **random forest** classifier [Breiman, 2001]; that has offered good results in many scenarios

[Fernández-Delgado et al., 2014]. The motivating idea of ensemble methods is to create a large number of base classifiers with high variance so that their outputs could be combined. In the case of random forests, this is achieved by using multiple decision trees as base classifiers that are learned with a different number of feature subsets. Figure 1.6 illustrates the process of prediction of a random forest classifier.

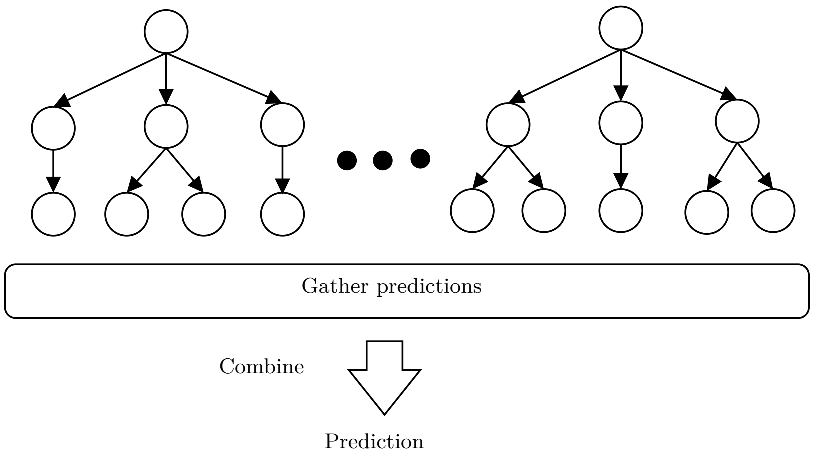


Fig. 1.6: Illustration of a Random Forest classifier. The predictions of the multiple decision trees are combined to provide a single prediction.

Another common learning models are the **Support Vector Machine (SVM)** [Cortes and Vapnik, 1995] and its multiple variants [Tax, 2001], and kernel-based learning methods [Moreno et al., 2003]. Briefly, a SVM finds a linear separation of instance belonging to different labels in a kernel space. It is common that data is not linearly separable, with respect to the class variable, in the input feature space. Therefore, this feature space is enlarged by means of kernel-based methods so that a hyperplane is found that correctly separates the different labels. This process is illustrated in Figure 1.7.

Another highly researched models are the **(artificial) neural networks**. In recent years, an overwhelming number of new research has been developed around these black-box style models that are inspired

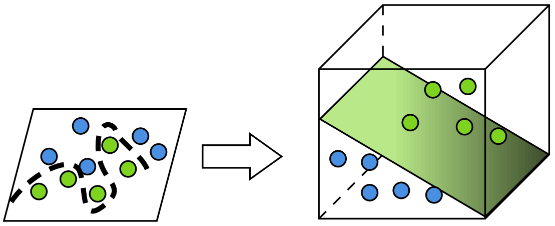


Fig. 1.7: Example of a SVM model. It enlarges the input feature space so that a linear (hyperplane) separation is obtained.

in the human brain. In a nutshell, a neural network can be seen as a multiple chained non-linear functions that are combined to provide a classification. They are composed by a variety of neurons that are inspired in the human neurons (see Figure 1.8). A human neuron is composed by multiple dendrites that receive some stimulus. Such stimulus are processed in the nucleus and then are sent to other neurons through a link called axon. In order to send the stimulus through the axon, there is some force that the nucleus produces to allow the output to arrive to its next destination. In an artificial neuron, the stimulus are feature values. In the nucleus, these inputs are added to produce an output. An activation function is used to mimic the force of the neuron output. Essentially, this is a non-linear function.

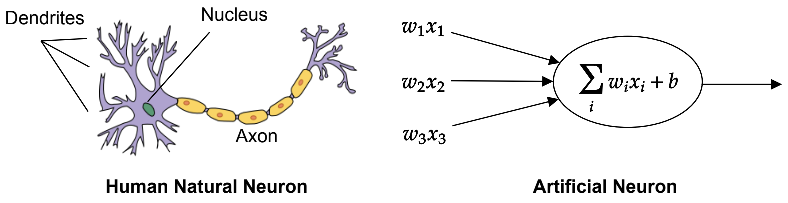


Fig. 1.8: Comparison between the artificial neural network and the natural human neuron.

Depending on the layout of the artificial neurons, different architectures can be created. The most popular one is the deep feed forward neural network shown in Figure 1.9. This architecture is divided in three main parts. Firstly, the input layer receives the features of each instance. Secondly, the outputs of the input layer are sent to a set of hidden layers that are composed by an *arbitrary* number of artificial neurons. Finally, the output layer produces a classification. The number of output neurons corresponds to the number of classes. When these architectures have several hidden layers, these are also framed under the *deep learning* framework [LeCun et al., 2015].

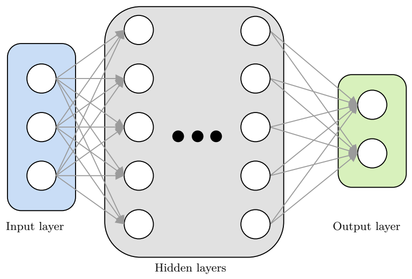


Fig. 1.9: Example of an artificial neural network architecture.

In computer vision, **convolutional neural networks** have achieved outstanding results when classifying images [Gu et al., 2018]. They are based on convolution kernels or filters that slide along the input. It has been discovered that the convolution operations, depending on the kernel selected, extract specific characteristics of the images such as edges. Therefore, the convolutional layers serve as feature extractors of the images [Gu et al., 2018]. Then, these features are used as inputs for a fully connected neural network that outputs a prediction. One of the most famous architectures is the LeNet neural network [LeCun et al., 1989] that is illustrated in Figure 1.10.

## 1.5 Learning From Unsupervised Data

This section focuses on reviewing a subset of unsupervised approaches, also known as clustering techniques, that have been used in the contributions of this dissertation.

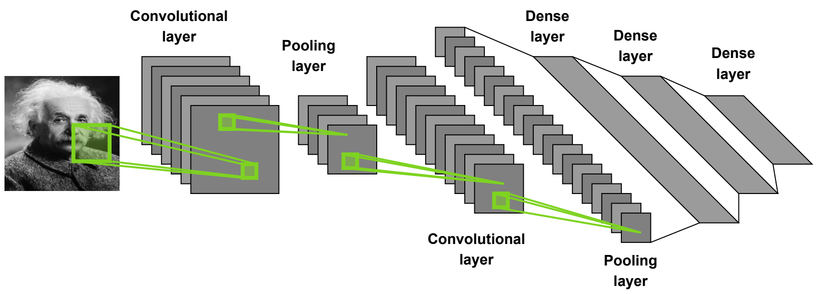


Fig. 1.10: LeNet convolutional neural network.

Learning from unsupervised data is the task of finding similar patterns in an unlabeled dataset. This task is of great importance in several applications. For instance, in marketing when trying to find similar clients based on their shopping interests. Besides, unsupervised classification techniques have also been successfully used to filter noise in a data preprocessing step [Carreño et al., 2020].

Defining *similarity* among two instances is not a trivial task and multiple algorithms derive from such definition. When referring to distance based models, the **$k$-Means** clustering algorithm can be found [Forgy, 1965]. It is an iterative method that finds $k$ centroids among the data instances under the assumption of cohesion and separation. The pseudocode of the $k$-Means clustering algorithm is shown in Algorithm 1. Commonly, the Euclidean distance is used as a similarity measure between the data points. Following this assumption, in Figure 1.11, $k$-Means obtains rounded, sphere-like clusters. In certain scenarios this shape could be a limitation and other approaches that are defined over other similarity definitions are found.

For instance, if density is used to consider similarity between the points, the **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)** algorithm comes across [Ester et al., 1996]. It is also an iterative model that starts placing a radius $\epsilon$ to each of the instances. Then, core points are identified if more than *minPoints* are inside the specified radius. Instances are added to the clusters defined by the core points if they are connected to other points of the same

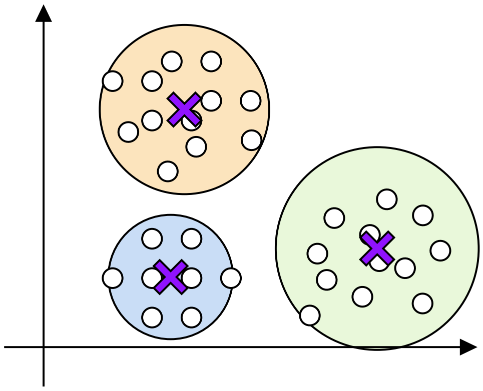


Fig. 1.11: Illustration of a 3-Means clustering algorithm. The crosses represent the cluster centroids.

**Algorithm 1:** Pseudocode of the $k$-Means algorithm.

```
Input: $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=1}^n$, $k$
Initialize cluster centroids $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\} \in \mathbb{R}^d$ randomly
while not converged do
    for $i \in \{1, 2, \ldots, n\}$ do
        $c_i = \arg\min_j ||\mathbf{x}_i - \boldsymbol{\mu}_j||^2$
    for $j \in \{1, 2, \ldots, k\}$ do
        $\boldsymbol{\mu}_j = \frac{\sum_{i=1^n} \mathbb{1}\{c_i=j\}\mathbf{x}_i}{\sum_{i=1}^n \mathbb{1}\{c_i=j\}}$
```

cluster. This process is illustrated in Figure 1.12. If an instance is not connected to any point, it is considered as outlier.

Another probabilistic method to cluster points is to assume that they have been generated by a mixture model, particularly from a mixture of Gaussian distributions. In this case, it is possible to learn the parameters of the model and therefore cluster the data by means of the **Expectation Maximization (EM)** algorithm. Suppose that we are given an unlabeled dataset $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=1}^n$ where $n$ represents the number of examples. Suppose that such data is generated from a $k$

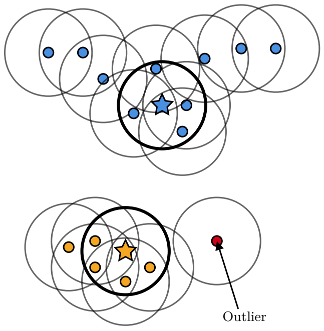


Fig. 1.12: Illustration of DBSCAN algorithm. The starred points are the core points. In this example, the *minPoints* are set to 4 and $\epsilon$ is represented with the radius of the drown circles.

component mixture of Gaussian distributions with a density function defined as

$$p(\mathbf{x}_i|\ \boldsymbol{\omega}) = \sum_{j=1}^{k} \tau_j f_j(\mathbf{x}_i|\omega_j), \tag{1.3}$$

where $\boldsymbol{\omega} = \{\omega_1, \omega_2, \ldots, \omega_k\}$ represents the vector of parameters of the $j^{\text{th}}$ mixture $\omega_j = (\tau_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, $\tau_j$ stands for the weight or the mixing proportion, $\boldsymbol{\mu}_j$ is the mean and $\boldsymbol{\Sigma}_j$ the covariance matrix, and $f_j$ is the probability density function of a multivariate Gaussian distribution, defined as:

$$f_j(\mathbf{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_j|^{1/2}} exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_j)\right) \tag{1.4}$$

We wish to model the data by specifying a joint distribution $p(\mathbf{x}_i, z_i) = p(\mathbf{x}_i\ |z_i)p(z_i)$. Here, $z_{i,j} \sim$ Uniform$(0, 1)$, and $\mathbf{x}_i|z_{i,j} \sim \sum_{j=1}^{k} \tau_j \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$. Note that the $z_{i,j}$ variables are latent random variables so they are not observed; otherwise, the problem would be trivial since $z_{i,j}$ provides the probability of the instance $x_i$ being generated from the mixture component $j$:

$$z_{i,j} = p(\mathbf{x}_i|\omega_j) = \frac{\tau_j f_j(\mathbf{x}_i|\theta_j)}{\sum\limits_{l=1}^{k} \tau_l f_l(\mathbf{x}_i|\omega_l)} \tag{1.5}$$

The estimation of the parameters of a Gaussian mixture is done by means of maximum likelihood estimation. Therefore, the loglikelihood function to optimize can be written as:

$$\log \mathcal{L}(\boldsymbol{\omega}|\mathcal{D}_u) = \sum_{i=1}^{n} p(\mathbf{x}_i|\ \boldsymbol{\omega}) \tag{1.6}$$

Computing Equations 1.5 and 1.6 correspond to the Expectation step of the EM algorithm. We would like to find the parameters of the Gaussian mixture that maximize the loglikelihood function (see Eq. 1.6). Hence, the Maximization step corresponds to obtaining the following parameters values:

$$\tau_j^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{z}_i^{(t)} \tag{1.7}$$

$$\boldsymbol{\mu}_j^{(t+1)} = \frac{\sum\limits_{i=1}^{N} \mathbf{z}_i^{(t)} \mathbf{x}_i}{\sum\limits_{i=1}^{N} \mathbf{z}_i^{(t)}} \tag{1.8}$$

$$\boldsymbol{\Sigma}_j^{(t+1)} = \frac{\sum\limits_{i=1}^{n} \mathbf{z}_i^{(t)} (\mathbf{x}_i - \boldsymbol{\mu}_j^{(t+1)})(\mathbf{x}_i - \boldsymbol{\mu}_j^{(t+1)})^T}{\sum\limits_{i=1}^{n} \mathbf{z}_i^{(t)}} \tag{1.9}$$

where $t$ represents the $t^{\text{th}}$ iteration. The algorithm should run until it converges. To initialize EM algorithm, there are two options. On the one hand, random values to the model parameters can be assigned, i.e. randomly assign values to $\boldsymbol{\omega}$ and $\boldsymbol{\tau}$. On the other hand, instances can be randomly assigned to the different $k$ clusters by randomly assigning values to $\mathbf{z}$. These two initializations correspond to starting either at Expectation or Maximization steps. A pseudocode for the EM algorithm is provided in Algorithm 2.

**Algorithm 2:** Pseudocode of the EM algorithm.

```
Input: $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=1}^n$, $k$
Initialize $\mathbf{z}$ by randomly sampling from $\mathcal{U}(0,1)$
while not converged do
    /* Maximization Step                                        */
    for $j \in \{1, 2, \ldots, k\}$ do
        $\tau_j = \frac{1}{n} \sum_{i=1}^{n} \mathbf{z}_i$
        $\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} \mathbf{z}_i \mathbf{x}_i}{\sum_{i=1}^{N} \mathbf{z}_i}$
        $\boldsymbol{\Sigma}_j = \frac{\sum_{i=1}^{n} \mathbf{z}_i (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j^{(t+1)})^T}{\sum_{i=1}^{n} \mathbf{z}_i}$
    /* Maximization Step                                        */
    $z_{i,j} = p(\mathbf{x}_i | \omega_j) = \frac{\tau_j f_j(\mathbf{x}_i | \omega_j)}{\sum_{u=1}^{k} \tau_l f_l(\mathbf{x}_i | \omega_l)}$
    $\log \mathcal{L}(\boldsymbol{\omega} | \mathcal{D}_u) = \sum_{i=1}^{n} p(\mathbf{x}_i | \ \boldsymbol{\omega})$
```

Clustering in high dimensional spaces is a challenging task [Parsons et al., 2004, Aggarwal and Yu, 2000]. Several techniques try to map the input space to another lower dimensional feature space in order to ease the task of clustering data. Afterwards, the aforementioned solutions could be applied. An approach based on deep learning can be found, named **autoencoder** neural networks, that learn an embedded lower dimensional representation of the data [Kramer, 1991]. An autoencoder consists of three parts, firstly, the *encoder* maps the input feature space into an embedded representation named *code* or *bottleneck*. Afterwards, these embeddings are reconstructed to match the original input data in the *decoder* part. In order to guide the learning process, the reconstruction error between the input and output instances is minimized. Figure 1.13 shows a graphical representation of an autoencoder. Once the network is learned, the encoder part serves as a feature compressor.

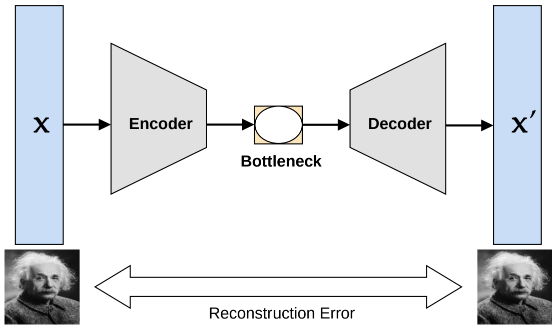


Fig. 1.13: Graphical representation of an autoencoder neural network.

## 1.6 Methods for Evaluating Classifiers

The performance of a classifier is commonly summarized in a evaluation metric, a real value that provides an estimation of its goodness. How to compute this value has been thoroughly researched and it is analyzed in Section 1.7. In this section, some different methodologies to obtain such value are reviewed.

As a first approach to assess the validity of a learned classifier, the same data used for learning can be used [Santafe et al., 2015]. However, this approach is unfair. In this setting, the ability of the model to remember the labels of the provided examples is tested; not its generalization capabilities. This approach is known in the literature as a *dishonest* evaluation method.

As a second simple approach, the provided data to learn the classifier is split into two subsets. The first subset that is commonly larger than the second one is used for learning. This is known as training set. The second subset, known as test set is used to assess the performance of the classifier. In this setting, since the model has never seen the test instances, its ability to generalize from the training set is evaluated. Note that in some situations, the learned model has some parameter that needs to be tuned. In order to tune this parameter, the training set is also split into two subsets, one is used for training a model with some selected hyperparameters, and the second one is used to evaluate the learned model with such hyperparameters. Once the model is learned with some specific hyperparameters, it is tested against the test set.

There are situations where the given training data is scarce. Therefore, splitting the dataset into two subsets *wastes* valuable instances. In order to test the learned classifiers in this scenario, K-Fold Cross Validation technique was proposed. In a K-Fold Cross Validation setting [Stone, 1974], the dataset is split into $k$ folds. Each fold will be used interactively to test a classifier that is learned with the rest of the folds. From each iteration, an score is obtained. This scores are summarized into a single score that represents the validity of the model. This procedure is illustrated in Figure 1.14.

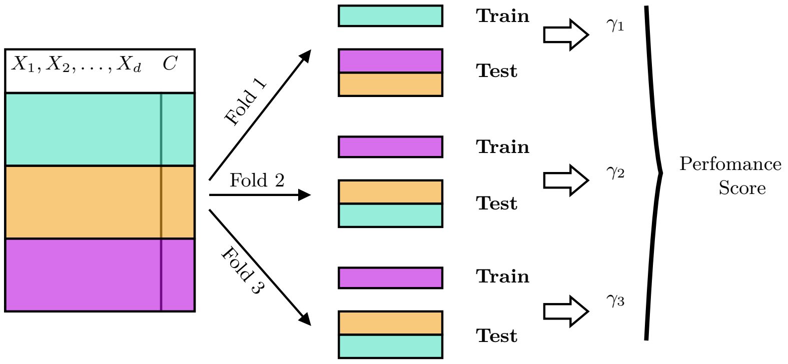


Fig. 1.14: Graphical representation of a 3-Fold Cross Validation evaluation method.

Another common evaluation method is the stratified K-Fold Cross Validation. When splitting the initial dataset, the labels may be underepresented in the created folds, the learned classified over the folds could be biased. In a stratified setting, it is ensured that all the folds have the same distribution of the classes.

## 1.7 Evaluation Measures

When evaluating a classifier over a test set, a confusion matrix can be obtained that shows the performance of a classifier (see Table 1.1). Nevertheless, the interpretability gets reduced when the number of classes increases. Also, comparing the performance of different algorithms is not straightforward from the confusion matrix. Therefore, evaluation measures or scores have been developed to serve as a summary value of the performance of a classifier. Most of these evaluation measures were initially developed for binary classification problems.

Table 1.1: Example of a confusion matrix of a problem with 3 classes.

| | | **Predicted** | | |
|---|---|---|---|---|
| | | **1** | **2** | **3** |
| **Real** | **1** | 50 | 10 | 10 |
| | **2** | 30 | 100 | 6 |
| | **3** | 30 | 10 | 45 |

When performing an evaluation method, a measure is commonly computed. Depending on the problem, and the importance of misclassification [Carreño et al., 2020, Ortigosa-Hernández et al., 2016], different evaluation scores are computed. For instance, in a medical diagnosis, predicting that a patient has cancer when in fact, he/she has not, is better than predicting that the patient is healthy when he/she has cancer for obvious reasons. Hence, computing a measure that explicitly points out this behavior is crucial.

Although many different classification measures have been developed [Carreño et al., 2020, Ortigosa-Hernández et al., 2016], classification *accuracy* is by far the most used one; mainly because its intuitive interpretation. It essentially averages the number of prediction successes that a classifier achieves. Interpreting a success as when the predicted class $\hat{c}_i$ matches the real class label $c_i$ of a given instance $\mathbf{x}_i$, it can be defined as

$$Acc = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\hat{c}_i = c_i) \tag{1.10}$$

As complementary to the accuracy metric, there is the *classification error*, that averages the number of errors that a classifier obtains. It can be defined as:

$$Error = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\hat{c}_i \neq c_i) \tag{1.11}$$

However, there are situation where these evaluation measures are not representative enough. For instance, in the aforementioned problem where the aim is to predict patients with cancer. The accuracy and error scores assume equal misclassification costs and this might not be true in multiple scenarios. Similarly, when dealing with highly unbalanced scenarios [Carreño et al., 2020, Ortigosa-Hernández et al., 2016], the classifier would get a high accuracy value by only outputting the majority class; which would derive into a useless assessment score.

In order to try to avoid erroneous interpretations when one of the described situations occur, some metrics have been proposed. Table 1.3 shows the most common scores that are used to evaluate classifiers. These are defined over a binary classification setting but they can be extended to a multiclass setting. In order to do that, the *one vs all* approach is commonly used, i.e., considering the class of interest as the positive class, and considering the rest as the negative class. Nevertheless, there are specific metrics for multiclass setting that are beyond the scope of this section.

Table 1.2: Description of a binary confusion matrix.

| | | **Predicted** | |
|---|---|---|---|
| | | + | - |
| **Real** | + | TP | FP |
| | - | FN | TN |

Table 1.3: Description of the most common evaluation measures used when assessing binary classifiers. These can be extended to multiclass scenarios if considering *one vs all* approach.

| **Evaluation Measure** | **Formula** |
|---|---|
| Accuracy | $\frac{TP+TN}{TP+TN+FP+FN}$ |
| Error | $\frac{FP+FN}{TP+TN+FP+FN}$ |
| True Positive Rate Recall or Sensitivity | $\frac{TP}{TP+FN}$ |
| True Negative Rate or Specificity | $\frac{TN}{FP+TN}$ |
| Precision | $\frac{TP}{TP+FP}$ |
| F-measure | $\frac{2TP}{2TP+FP+FN}$ |

# 2

# Analyzing Rare Event, Anomaly, Novelty and Outlier Detection Terms Under the Supervised Classification Framework

In this chapter, rare event, anomaly, novelty and outlier detection terms are analyzed under the supervised classification point of view. Associated to these terms, some widely different problems can be found that are often interchangeably referred to with the aforementioned terminology and vice-versa. Under this hindering situation we propose a one-to-one assignment of terms and problems that give a short step towards the standardization of the field. Furthermore, in order to validate our proposed assignment, we perform a set of experiments by retrieving papers from Google Scholar, ACM Digital Library and IEEE Xplore search engines.

## 2.1 Introduction

Numerous applications require filtering or detecting *abnormal* observations in data. For instance, in security, intruders are abnormalities [Ribeiro et al., 2016, Pimentel et al., 2014, Luca et al., 2016, Phua et al., 2010, Yeung and Ding, 2001]; in traffic data, road accidents [Theofilatos et al., 2016]; in geology, the eruption of volcanoes [Dzierma and Wehrmann, 2010]; in food control, foreign objects inside food wrappers [Einarsdóttir et al., 2016]; in economics, bankruptcy of a company [Fan et al., 2017]; or in neuroscience, an unexperienced stimulus is considered an abnormality [Kafkas and Montaldi, 2018]. In some situations, the abnormalities are called rare events, anomalies, novelties, outliers, exceptions, aberrations, surprises, peculiarities, noise or contaminants among others. Of these, the most common terms in the literature are rare event, anomaly, novelty and outlier.

Considering the importance of abnormalities in different areas, a lot of research has been done, mainly in the last 10 years. However, the fact that these contributions have been carried out in different knowledge areas, a mix-up between names and problems has occurred in the literature. Particularly, when the same term is used in distinct disciplines but with other meaning and vice versa. Moreover, the terminology has changed over time and even in the same discipline; a similar problem has been named differently in different time periods. On the one hand, different names have been used for similar problems. For instance, Van Den Eeckhaut et al. [2006] deal with a problem of predicting, in a fixed period of time, the risk factor of a landslide in an area. The authors create a landslide susceptibility map in which each area is scored based on the risk of a landslide. This is done using historical data of either normal and ground which has suffered a landslide (abnormal). In this study, the authors refer to landslides as *rare events* because landslides seldom occur. In Ribeiro et al. [2016] a similar problem is addressed, but with a different term. Here, a study in the railway industry is carried out. Train passenger doors have several subsystems in order to keep them open or closed according to a variety of safety and comfort rules. In some situations these doors fail due to the deterioration of the system. Therefore, the authors predict whether the door is going to fail in a fixed period of time or not. In order to do that, both normal and failure historical data is used to learn a model. In this case, the door failures are referred to as *anomalies.* As can be seen, both problems are very similar and different terms have been used to refer to the abnormalities. In both problems, temporal data of normal and abnormal classes is available to build the prediction models.

On the other hand, the same terms have been used to describe widely different problems. In the following two problems, the authors use the term *novelty* to describe the abnormalities. In Luca et al. [2016] a variety of patients are constantly monitored with a 3D accelerometer. Those patients eventually suffer an epileptic seizure. Due to abrupt movement during a seizure, the patient could became injured. Therefore, detecting this behavior as soon as possible is relevant in order to avoid this harmful situation. In order to predict if a patient is suffering an epileptic seizure, a model is built based on the recorded movement data of several patients. The data consists of 3D accelerometer data divided in fixed time windows in which whether or not an epileptic seizure has occurred is annotated. However, notably less abnormal (seizure) data is available due to the eventuality of these attacks. In the prediction phase, given new information about a currently monitored patient, the classifier detects if the patient is suffering an attack at that moment. Einarsdóttir et al. [2016] detect foreign objects inside food envelopes. A classifier is learned only from food-images without abnormal objects. In other words, the model is learned using information of only one class. However, in the detection phase, the model classifies new instances in two classes, normal (without foreign objects) and abnormal (with foreign objects). While both examples are named with the same term, the problems are widely different. For instance, the former has both normal and abnormal data available to train the model, whereas the latter only learns from a dataset with observations of only one class.

As we have seen in the previous paragraphs, there is an important mix-up between terms and problems. Possibly motivated by the same mix-up detected by us, some papers that present specific learning methods have made an effort in their introduction section to discuss the differences between one or two terms, or to clearly define their learning scenario. However, to the best of our knowledge, no paper in the literature has treated the four rare event, anomaly, novelty and outlier terms under the supervised classification point of view. For instance, in Luca et al. [2016], Dufrenois and Noyer [2016] a brief discussion about the novelty term and one-class classification framework is made. In Weiss and Hirsh [1998], the authors clearly define their rare event learning scenario. In Campos et al. [2016], an effort is made to distinguish between one class classification and outlier detection. Finally, in Ribeiro et al. [2016], three methods related with outlier, anomaly and novelty detection learning scenarios are used to solve the same problem. Also, some insights are given about all these three learning scenarios. However, none of these papers frame the corresponding terms into the supervised classification framework.

This confusion calls for the repetition of research and hinders the advance of the field. Therefore, the aim of this work is to contribute with a first step in the organization of the area. In order to do that, this work underlines the differences between each term, and organizes the area by looking at all these terms under the umbrella of supervised classification. Particularly, for each term, the most frequently used learning scenario is associated.

| **Paper Reference** | **Brief description of the main characteristics of the paper** | **Term used** |
|---|---|---|
| Van Den Eeckhaut et al. [2006] | The risk factor of a landslide is predicted in a fixed period of time. The data consist of historical landslided and normal land features. | Rare event |
| Ribeiro et al. [2016] | Failure of the train passenger doors is predicted in a fixed period of time. The data consist of both normal and failure temporal instances. | Anomaly detection |
| Luca et al. [2016] | Whether the patient is suffering an epileptic seizure is predicted. The data consists of normal and abnormal patient records. The patient records are timely monitored. | Novelty detection |
| Einarsdóttir et al. [2016] | The presence of foreign objects inside food envelopes is predicted. The data consist only of images of food envelopes without foreign objects. | Novelty detection |

Table 2.1: An illustrative example of the mix-up between terms and problems in the literature.

This chapter is organized as follows. Each section describes a supervised learning scenario: Section 2.2 describes rare event detection, Section 2.3, anomaly detection and Section 2.4, novelty detection. In each section, the objective of the classification task, the characteristics of the input data and the most popular techniques for the described learning scenario are reviewed. In Section 2.5, the related outlier term is treated. In Section 2.6, the one-to-one assignment of terms to learning scenarios is described coupled with a brief discussion about the main evaluation techniques of each learning scenario. In Section 2.7, the experimental validation is described. Finally, in Section 2.8, the conclusions of this work are exposed.

## 2.2 Rare Event Detection

Almost all the papers that use the term *rare event* to describe the abnormalities of the problem to be solved share the time dimension as a common characteristic [Theofilatos et al., 2016, Dzierma and Wehrmann, 2010, Heard et al., 2010]. For instance in Theofilatos et al. [2016], a road accident study in the Attica Tollway (Greece) is performed. The authors divide the tollway into different sections and they detect the occurrence of an accident in a certain section of the highway. A model is built based on recorded data from ground-sensors and traffic-cameras. More specifically, the data is sliced into one-hour time intervals and manually labeled by experts. Therefore, given a new one-hour time interval, the model detects an accident occurrence. In Dzierma and Wehrmann [2010], a geomorphological study is performed. The authors predict if a new volcano eruption is going to happen in a fixed period of time. A Poisson Process is learned with the historical Volcanoes Explosivity Index (VEI) of two volcanoes. Next, given a new VEI of one of the two volcanoes, the occurrence of the eruption in a fixed time interval is predicted.

In the previously described problems, the goal consists on the prediction of occurrence of a rare event in a bound period of time. A genuine characteristic of the rare event learning scenario, from a supervised classification point of view, is that the instances are time series [Hamilton, 1994]. From this perspective, the objective is to classify new incoming time series as rare (when the rare event has occurred) or normal (no event has occurred) using a previously learned model. This approach is known in machine learning as supervised time series classification [Esling and Agon, 2012]. However, due to the temporal nature of the problem, two different classification approaches can be found in the literature. Firstly, the *full length supervised time series classification* is dealt with. For example, in Murray et al. [2005], the SMART[1] dataset is used to detect if a hard-drive is faulty in a fixed period of

[1] The SMART dataset is hard-drive self-monitoring recovered data in which both normal and failure behaviors are collected.

time. The authors learn a model using recorded hard-drive sensor measurements at different times. Then, given new hard-drive sensor data, failure is detected. In Zhang et al. [2017], a termo-technology dataset which contains information gathered over time about heating systems is used. The objective is to detect if the heating system has failed in a fixed period of time. Secondly, another type of classification of rare events can be found in the literature, in which the objective is to classify new observations (time-series) as early as possible, preferably before the full time series is available. This approach is known as *early supervised time-series classification* in machine learning literature [Mori, 2015]. For example, in Ogbechie et al. [2017] a prediction of faulty metal bars is studied. During the bar melting process, several sensors monitor the characteristics of each bar. These measurements, recovered from both normal and faulty bars, are used to learn a model. Next, given information about a new bar, the classifier predicts if the bar is going to be faulty. The early detection of a faulty bar is crucial because, depending on when it is detected, it can be fixed during the rest of the process.

According to the characteristics of the data, in most of the problems referred to with the *rare event* term, instances are time series and are labeled in two categories: normal ($\mathcal{N}$) and rare ($\mathcal{R}$). Furthermore, in many papers, the data shows an unbalanced distribution of classes. Formally, assuming that the data is generated by a generative mechanism $P(\mathbf{x}, c)$ [Mitchell, 1997], $P(C = \mathcal{R}) \ll P(C = \mathcal{N})$. Considering the instances during the training stage, both normal and abnormal instances are available to learn the classifier. Therefore, rare event classification can be formalized as a (highly) unbalanced supervised time series classification problem [Köknar-Tezel and Latecki, 2011, Cao et al., 2011]. Formally, this scenario can be described as follows:

A time series (TS) is an ordered pair (timestamp, value) of fixed length $m$:

$$\begin{aligned} TS = \{(t_1, x_1), \ldots, (t_i, x_i), \ldots, (t_m, x_m)\} \\ \text{with } t_i \in \mathbb{N}, \text{ for } i = 1, \ldots, m \end{aligned} \tag{2.1}$$

Time series classification is a supervised data mining task in which giving a training set of time series, $\mathbf{TR} = \{(\mathbf{TS}_1, y_1), \ldots, (\mathbf{TS}_n, y_n)\}$, in which $y$ represents the label of the corresponding time series, the objective is to build a classifier that is able to predict the class label of any new time series as accurately as possible [Mori, 2015]. In the particular case of a rare event, it is common to have a scenario where $P(C = \mathcal{R}) \ll P(C = \mathcal{N})$. A common classification process can be seen in Figure 2.1.

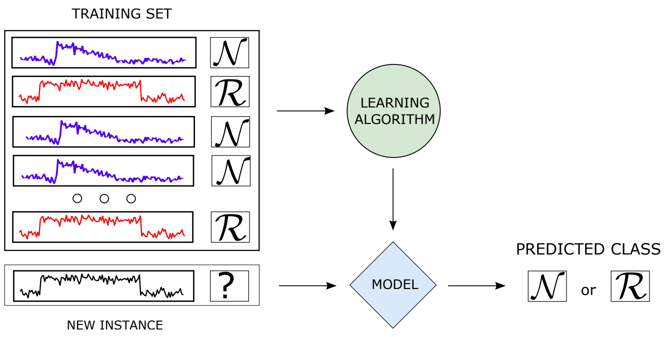


Fig. 2.1: A flowchart of the supervised time series classification data mining task (Mori [2015]).

Besides, there are some problems in the literature in which the prediction must be output as soon as possible. This learning scenario is known as early time series classification [Mori et al., 2018].

However, even though the problem itself has the time dimension as a key component, in some rare event detection applications, instances are transformed without considering this genuine characteristic. Therefore, the approach treats the problem as an unbalanced non-temporal classification task, similar to those found in the anomaly detection learning scenario (further described in Section 2.3). For instance in Murray et al. [2005], the data is composed of several hard drive sensor measurements at different time intervals. Therefore, for the same drive, many readings of the same sensors are available. However, the authors do not consider the order in which the measures have been recorded, and, given new hard-drive unordered measurements, the model classifies the drive as faulty or normal. Hence, the temporal nature of the data is not leveraged.

Regarding the rare event literature, the objective of most of the related papers is focused on classifying the rare class. Therefore, in order to evaluate the performance of the classification task, popular metrics such as AUC [Zhang et al., 2017, Xu et al., 2016, Ren et al., 2016] and the recall of the rare class [Zhang et al., 2017, Ren et al., 2016] have been commonly used.

Among the most frequently used techniques in time series classification, rare event logistic regression, an adaptation of the logistic regression for this learning scenario, is a popular choice [Theofilatos et al., 2016, Van Den Eeckhaut et al., 2006, Ren et al., 2016, King et al., 2001]. However, techniques such as Kullback-Leibler divergence to discriminate between rare and normal events [Xu et al., 2016], long-short term neural networks [Zhang et al., 2017], rule-based classification learned with genetic algorithms [Weiss and Hirsh, 1998], multiple-instance naïve Bayes [Murray et al., 2005], Poisson Processes [Dzierma and Wehrmann, 2010], support vector data regression with surrogate functions [Bourinet, 2016], Bayesian networks [Cheon et al., 2009] or support vector machines [Khreich et al., 2017] have been successfully adapted for this learning scenario.

Taking into account the unbalanced distribution of classes, most of the previous methods are coupled with techniques specifically designed to deal with unbalanced time-series classification. Some of these techniques include: the Structure Preserving Over Sampling (SPO) technique [Cao et al., 2011], or an adaptation of the classical Synthetic Minority Over-sampling TEchnique (SMOTE) [Köknar-Tezel and Latecki, 2011].

Finally, another widely different rare event related learning scenario can be found in the literature. The estimation of the probability of occurrence of a rare event [Wu et al., 2003, Cadini et al., 2017, Dessai and Hulme, 2004, Dueñas-Osorio and Vemuru, 2009, Bedford and Cooke, 2001]. This approach is mainly used in engineering and physics and some illustrative examples of rare event probability estimation include: the estimation of the probability of infrastructure failure in a fixed period of time [Dueñas-Osorio and Vemuru, 2009], the estimation of the probability of failure of technical systems in a fixed period of time [Bedford and Cooke, 2001], or the estimation of the probability of extreme climate developments in a specific time window [Dessai and Hulme, 2004]. Since this learning scenario is beyond the supervised classification framework, it is not considered in this chapter. Among the most frequently used techniques in order to estimate the rare event probability, importance sampling, Monte Carlo simulations [Balesdent et al., 2016, Auffray et al., 2014], kriging [Auffray et al., 2014] or first order reliability method (FORM) [Straub et al., 2016] are found in the literature.

## 2.3 Anomaly detection

Most of the problems which describe the abnormalities with the anomaly term are non-temporal. The data is labeled in two categories: normal ($\mathcal{N}$) and anomaly ($\mathcal{A}$). For instance, in Miri Rostami and Ahmadzadeh [2018], the authors detect breast cancer using the Surveillance Epidemiology and End Results (SEER) [1] dataset. This dataset consists of patients which have been examined for cancer diseases. The patients which suffer from cancer are described with anomaly term. Hence, the data consists of cases of both normal and anomalous instances. A model is then learned which classifies new unseen cases as anomalous or normal. Fiore et al. [2017] detects credit-card transactions using a public dataset with legal and notably less fraudulent transactions. A neural network is learned to classify new incoming transactions as legal or fraudulent.

In anomaly detection learning scenario, anomalous instances are scarce due to the unbalanced distribution between normal and anomaly classes [Chandola et al., 2009]. Therefore, this scenario can be formalized as (highly) unbalanced supervised classification. Formally, an instance is defined as $\mathbf{x} = (x_1, \ldots, x_m)$. Given a training set $\mathbf{TR} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, in which $y$ represents the label of the corresponding instance, the objective is to learn a classifier that is able to predict a new class label of any new instance as accurately as possible. Regarding the probability distribution of the class variable, $P(\mathcal{A}) \ll P(\mathcal{N})$. Where $\mathcal{A}$ represents the anomaly class label and $\mathcal{N}$ the normal class label. An illustrative example can be seen in Figure 2.2.

In order to evaluate the performance of the classifiers, due to the (highly) unbalanced distribution of classes, common metrics such as accuracy are not informative enough. Therefore, authors focus on the correct classification of abnormalities. A popular evaluation measure used is the maximization of the recall of the minority class [Ribeiro et al., 2016, Miri Rostami and Ahmadzadeh, 2018].

[1] Available here: https://seer.cancer.gov/data/

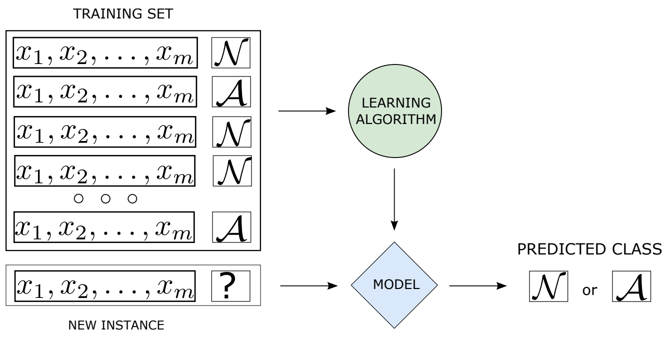

TRAINING SET
$x_1, x_2, \ldots, x_m$ $\mathcal{N}$
$x_1, x_2, \ldots, x_m$ $\mathcal{A}$
$x_1, x_2, \ldots, x_m$ $\mathcal{N}$
$x_1, x_2, \ldots, x_m$ $\mathcal{N}$
$x_1, x_2, \ldots, x_m$ $\mathcal{A}$
LEARNING ALGORITHM
$x_1, x_2, \ldots, x_m$ ?
NEW INSTANCE
MODEL
PREDICTED CLASS
$\mathcal{N}$ or $\mathcal{A}$

Fig. 2.2: A flowchart of a supervised classification task. This learning scenario is assigned to the anomaly detection term.

For anomaly detection, popular supervised classifiers have been adapted obtaining competitive results. For instance, support vector machines [Zhou et al., 2017], neural networks [Noto et al., 2012] or Gaussian mixture models [Reynolds, 2015] present genuine algorithms to deal with anomaly detection domains. Note that, since anomaly detection can be formalized as a (highly) unbalanced supervised classification problem, techniques that specifically deal with unbalanced domains can be used for anomaly detection. Similar to the rare event oversampling techniques, SMOTE [Miri Rostami and Ahmadzadeh, 2018, Araujo et al., 2018], is widely used to synthetically generate instances from the minority class.

## 2.4 Novelty detection

In most of the papers that use the term novelty to describe the abnormalities, the model is learned using a dataset that contains only one class. For instance, in Khreich et al. [2017], system call traces are classified as novel or normal. A novel instance corres2ponds to an unsupported or unexpected system call trace. To learn the model, only normal system call traces which have been gathered in a secure environment are used. When a new system call arrives, the classifier predicts it as normal or novel. Similarly, in Einarsdóttir et al. [2016], a study in food control is carried out. Specifically, in some cases, foreign objects can be found inside food envelopes. Since this situation can result in bad customer experience and legal issues, the detection of foreign objects is crucial. The authors learn a classifier using X-ray images only from normal food (without foreign objects inside). Next, giving a new unseen X-ray image, the classifier predicts it as novel (with foreign object inside) or normal. The novelty term has also been commonly used in streaming scenarios. Masud et al. [2013] start from a labeled dataset, where an initial model is learned. This model classifies the new incoming instances either among the normal known classes or as novel (the instance is not similar to any known class). If this new instance is classified as novel, it is kept in a buffer because it is considered as a candidate for a new class. When this buffer is full, new classes are sought in this buffer. The classifier is updated with new emerging novel classes for future predictions.

Regarding the two aforementioned problems, two different learning scenarios can be considered. What we call the *static novelty detection* learning scenario is considered. Here, the problem can be cast as a binary supervised classification problem. Given a dataset composed by only one class, a model is built. This model learns a decision boundary that isolates the normal behavior. For prediction, when a new instance arrives, it is classified as novel or as normal. In this framework, the efforts are focused on correctly classifying the normal class [Pimentel et al., 2014, Einarsdóttir et al., 2016, Kafkas and Montaldi, 2018]. Therefore, in order to evaluate the performance of the classifiers, the recall of the normal class is commonly maximized [Luca et al., 2016, Swarnkar and Hubballi, 2016]. Formally, the training set is generated only from $P(\mathbf{x}|C = \mathcal{N})$. At the training stage, even though the classifier is learned using information about only one class (normal class), it is built considering that another behavior exists which is different that which is normal.

Formally, an instance is defined as $\mathbf{x} = (x_1, \ldots, x_m)$. Given a training dataset, $\mathbf{TR} = \{(\mathbf{x}_1, y_1 = \mathcal{N}), \ldots, (\mathbf{x}_n, y_n = \mathcal{N})\}$, the objective is to learn a classifier that will be able to predict between normal $\mathcal{N}$ and *novel*. Note that, in this learning scenario, only one class, the normal class $\mathcal{N}$, is available to train the model. An illustrative example can be graphically seen in Figure 2.3.

Besides, what we call *dynamic novelty detection* is considered. In some situations, in the literature, it is also known as *evolving classes*,

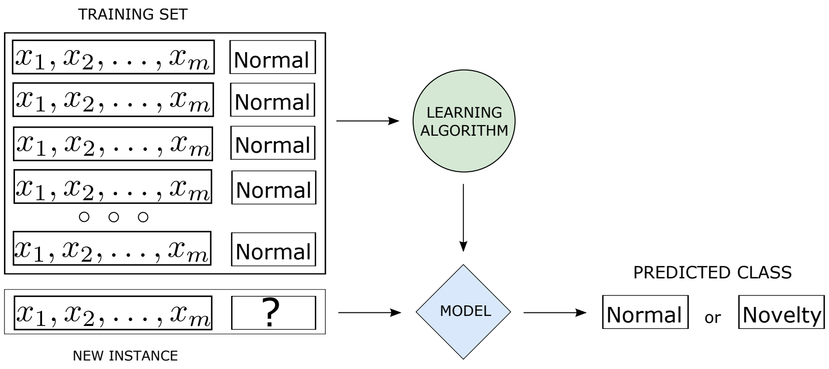
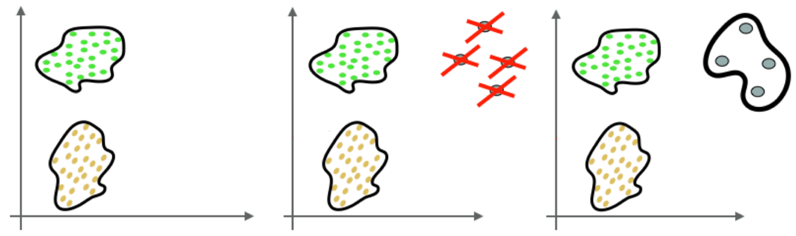


Fig. 2.3: A flowchart of the supervised classification framework task in which only one class is available to learn the classification model. This learning scenario is assigned to the the static novelty detection term.

*future classes* or *novel class detection* [Faria et al., 2016, Masud et al., 2013, Mu et al., 2018]. This learning scenario can be formalized as a supervised classification problem in which the number of labels for the class variable is unknown. In other words, the generative probability distribution dynamically changes during the classification process. Therefore, the classifier has to adapt to these changes. When a new instance arrives, the model has to classify among the current classes or it stores it in a buffer [Masud et al., 2013, Spinosa et al., 2007, Zhu et al., 2018]. Considering the life-cycle of the classes, these can drift, be born, die or reappear. Hence, the classifier must be updated for those changes, considering that the adaptation time is relevant in a streaming environment. Note that most of the existing approaches consider a dynamically (highly) unbalanced supervised classification scenario [Masud et al., 2013, Spinosa et al., 2007, Zhu et al., 2018, Chen et al., 2008] since a few instances may constitute a new emerging class (Figure 2.4). To evaluate the performance of the classifier in this environment, genuine metrics have been proposed. For instance, Masud et al. [2013] use the percentage of novel class instances classified as a current class; the percentage of existing class instances falsely identified as novel; and, the total misclassification error. Zhu et al. [2018] use the average precision among all classes. Chen et al. [2008] output the evolution of the classification error as new events occur: the emergence of a class, disappearance or drift.

The dynamic novelty detection learning scenario can be divided in two stages. Firstly, the initial learning stage (also known as offline stage), in which given a labeled training dataset, a model is built. Secondly, the prediction stage (also known as online stage), in which new classes may emerge and disappear, and the old classes may also drift. These two phases are formalized as follows:



(a) Initial training example. (b) Classification of instances. (c) Class discovery.

Fig. 2.4: Flowchart of the dynamic novelty detection learning scenario. At the beginning, the given classes are modeled. When new instances arrive, they are classified among known classes or they are rejected as not belonging to any existing class (see the crossed instances in Figure 5b). Finally, the new emerging class is sought.

**Initial training phase (offline):** In the offline phase a classifier $C_0$ is learned considering a set of labels $L_0$.

**Prediction phase (online):** The online phase can be described as a prediction and adaptation stage in which a data stream ($DS$) is observed. A $DS$ is a possibly infinite sequence of instances. At time $t$, the current classifier $C_t$ predicts a new instance. If the instance is classified as one of the current labels, the classifier is adapted with this knowledge to create $C_{t+1}$. If the new instance can not be classified in the current set of labels, it is kept apart in a buffer and the model does not modify. Once the buffer is full the classifier is updated and the set of labels $L_t$ modified.

An illustrative flowchart of this learning scenario can be seen in Figure 2.5.

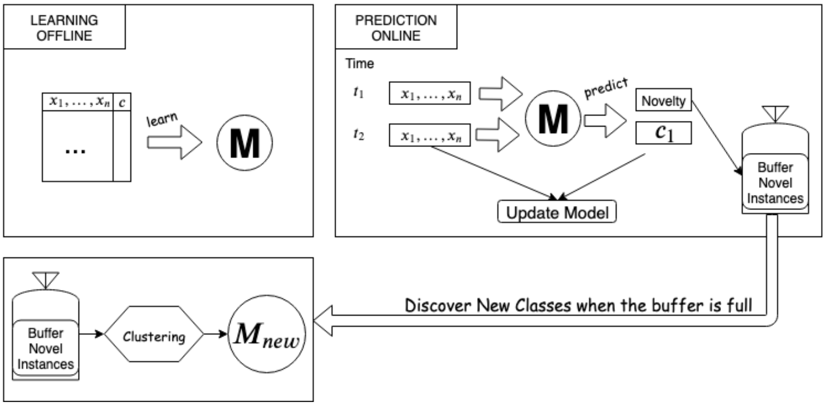


Fig. 2.5: A flowchart of the dynamic novelty detection problem. In this problem, the number of labels for the class variable is unknown, and dynamically changes over time.

According to the techniques used in *static novelty detection*, one class classification techniques are those which are the most representative ones in this learning scenario. For instance, one class SVM [Dufrenois and Noyer, 2016, Khreich et al., 2017, Erfani et al., 2016], K-Nearest Neighbors data description [Tax, 2001], graph embedded one class classifiers [Mygdalis et al., 2016], one class Random Forests [Désir et al., 2013] and Isolation Forest [Zhang et al., 2011] have been successfully applied under the *static novelty detection* learning scenario. Besides, in *dynamic novelty detection*, techniques such as OLINDDA [Spinosa et al., 2007], a sphere-based novelty detection algorithm, in which clustering is done with the k-means algorithm; MuENLForest [Zhu et al., 2018] which discovers new labels in a multi-label classification framework by creating an ensemble of Random Forest and Isolation Forest classifiers to discover emerging new classes; or the ensemble proposed in Masud et al. [2013], which creates an ensemble of decision trees which, in each leaf node, runs a k-means algorithm to discover sphereshaped emerging new classes, have been successfully proposed in the literature.

## 2.5 The related outlier detection scenario

The outlier term also comes up when seeking related works with rare event, anomaly and novelty terms. While the term is mainly associated with an unsupervised framework, the literature shows examples where the term is used to name other previously explained scenarios [Hodge and Austin, 2004, Zhang and Zulkernine, 2006, Billor et al., 2000]. Therefore, it is briefly considered in this section.

In some papers, the term outlier has been related with *noise*, linking these observations with incorrect or inconsistent behaviors [Aggarwal, 2017]. Consequently, the outlier detection task forms part of a preprocessing phase [Teng et al., 1990, Rousseeuw et al., 2011]. For instance, when human errors are introduced retrieving data, these erratic observations are considered outliers [Barai and Lopamudra, 2017]. In other situations, the detection of instances with high deviation are considered outliers [Radovanović et al., 2014, Dang et al., 2014]. In Radovanović et al. [2014], the authors detect all-star players in an unlabeled dataset composed by NBA players between 1973 and 2003. The outstanding NBA players are considered outliers. In order to detect them, clustering is pursued and those points which deviate significantly from others are considered outstanding NBA players.

Regarding the characteristics of the data in the outlier detection scenario, it can be either temporal (time-series) [Gupta et al., 2013] or non-temporal [Campos et al., 2016, Aggarwal, 2017, Radovanović et al., 2014].

An outlier detection task can be formalized as an unsupervised classification problem. Formally, given a dataset $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the objective is to find the instance that (highly) deviates from others. An example of an outlier detection task can be seen in Figure 2.6.

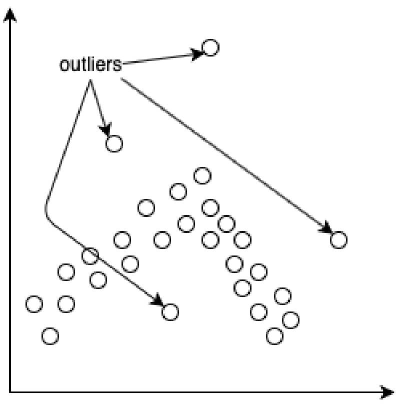


Fig. 2.6: An example of unsupervised classification. This learning scenario is assigned to the outlier detection task. As can be seen the outliers are deviated instances without a clear pattern.

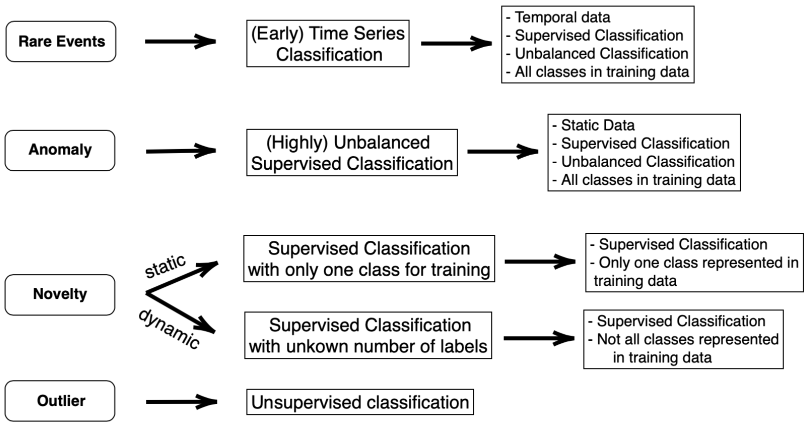

Rare Events
(Early) Time Series Classification
- Temporal data
- Supervised Classification
- Unbalanced Classification
- All classes in training data
Anomaly
(Highly) Unbalanced Supervised Classification
- Static Data
- Supervised Classification
- Unbalanced Classification
- All classes in training data
Novelty
static
dynamic
Supervised Classification with only one class for training
- Supervised Classification
- Only one class represented in training data
Supervised Classification with unkown number of labels
- Supervised Classification
- Not all classes represented in training data
Outlier
Unsupervised classification

Fig. 2.7: Assignment of terms to learning scenarios. The main characteristics of each learning scenario have been summarized.

## 2.6 The proposed assignment of terms and learning scenarios

In this chapter, based in our experience and initial approach to the literature, we did discover two major issues: a) the existence of a problematic mix-up between terms and learning scenarios. And b) we realize that most of these problems can be put in the same learning framework. Furthermore, we based on the assignment of terms to problems in these key papers to design our taxonomy. For each paper, we have reviewed the goal of the paper, the characteristics of the input data and the most representative techniques used in each rare event, anomaly, novelty and outlier detection works. Concretely, for each term related paper, the problem that the authors want to solve, such as, whether it is a time series classification, has an unbalanced learning characteristic, it is a classification task or a regression task, which the evaluation measures are, and, if it is a supervised or unsupervised classification problem has been reviewed.

In Figure 2.7, the assignment of terms to learning scenarios is graphically explained. As can be seen, each term is associated with one learning scenario. Moreover, the genuine characteristics of each learning scenario are shown in this figure. Also, an extended summary is exposed in Table 2.2.

In the case of the rare event term, the most relevant learning scenario under the supervised classification point of view is the (early) time series classification. In most of the papers described with the rare event term, there is a temporal nature in the problem, the classes are unbalanced and all the classes are represented in the training set.

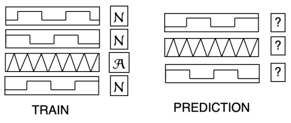
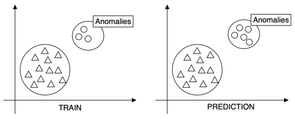
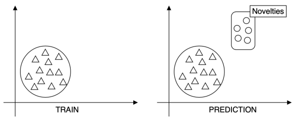
In the problems described with the anomaly term, the most relevant learning scenario is the (highly) unbalanced supervised classification. In this learning scenario, the data is static, the distribution of classes is unbalanced, and all the classes are represented in the training set.

Regarding the problems described with the novelty term, two different learning scenarios are considered. On the one hand, the static novelty detection in which the objective is to classify an instance between novelty or normal based on a model which has been trained with only the normal class. On the other hand, the dynamic novelty detection is considered. In this learning scenario, the objective is to discover new emerging classes in an streaming environment. However, both learning scenarios share some common characteristics, such as: both of the learning scenarios are supervised, and, both of them try to discover instances from classes that were not available in the training set. Hence, both of the learning scenarios do not have all the classes represented in the training data (in the case of static novelty detection, the novel class is not available. In the case of dynamic novelty detection, the new novel classes are neither available in the training set).

Finally, the outlier detection term has been mostly associated with the unsupervised classification framework in the literature.

All these learning scenarios require specific measures in order to evaluate the performance of the classifiers that solve the related problems. Therefore, depending on the objective of the classification task, different measures are commonly computed in the literature. In Table 2.3, the most common evaluation measures for each term are presented. Regarding the evaluation techniques used to validate the performance of the classifier, in the majority of the papers the k-fold cross validation, stratified k-fold cross validation and the train and test split are used.

Table 2.2: Summary of the main characteristics of each term along with the key references of the literature.

| Term | Description | Key References |
|---|---|---|
| **Rare Event**  (Early) Supervised Time Series Classification | • Temporal data<br>• All classes represented in the training set<br>• Unbalanced class distribution<br>• Supervised classification | Theofilatos et al. [2016], Dzierma and Wehrmann [2010], Heard et al. [2010], Hamilton [1994], Zhang et al. [2017], Ogbechie et al. [2017] |
| **Anomaly**  (Highly) Unbalanced Supervised Classification | • All classes represented in the training set<br>• Unbalanced classification<br>• Supervised classification | Miri Rostami and Ahmadzadeh [2018], Fiore et al. [2017], Chandola et al. [2009] |
| **Novelty**  Supervised Classification only one class for training | • Possible unbalanced classification<br>• Supervised classification | Masud et al. [2013], Pimentel et al. [2014], Luca et al. [2016], Einarsdóttir et al. [2016], Kafkas and Montaldi [2018], Khreich et al. [2017], Swarnkar and Hubballi [2016], Spinosa et al. [2007], Zhu et al. [2018], Chen et al. [2008], Masud et al. [2009] |

*Continues in the next page.*

Table 2.2: Summary of the main characteristics of each term along with the key references of the literature (cont.).

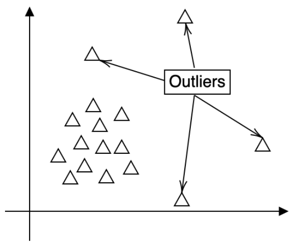
| **Term** | **Description** | **Key References** |
|---|---|---|
| **Outlier**  Unsupervised Classification only one class for training | • Possible temporal data. • Unsupervised classification | Campos et al. [2016], Hodge and Austin [2004], Zhang and Zulkernine [2006], Billor et al. [2000], Aggarwal [2017], Teng et al. [1990], Rousseeuw et al. [2011], Barai and Lopamudra [2017], Radovanović et al. [2014], Dang et al. [2014], Gupta et al. [2013] |

Table 2.3: Summary of the most used evaluation measures of each term related learning scenario.

| Term | Evaluation Measure | Formula | References |
|---|---|---|---|
| **Rare Event** | Accuracy | $\frac{\eta_{\mathcal{A}}+\eta_{\mathcal{N}}}{\|D\|}$ | Zhang et al. [2017], Xu et al. [2016], Ren et al. [2016] |
| | Recall of Rare Events | $\frac{\eta_{\mathcal{A}}}{\eta_{\mathcal{A}}+\eta_{\mathcal{A}\to\mathcal{N}}}$ | |
| | Earliness | $\frac{1}{\|D\|}\sum_{x\in D}\frac{t_x^*}{L}\cdot 100\%$ | Mori et al. [2018] |
| **Anomaly** | Accuracy | $\frac{\eta_{\mathcal{A}}+\eta_{\mathcal{N}}}{\|D\|}$ | Luca et al. [2016], Swarnkar and Hubballi [2016] |
| | Recall of Anomalies | $\frac{\eta_{\mathcal{A}}}{\eta_{\mathcal{A}}+\eta_{\mathcal{A}\to\mathcal{N}}}$ | |
| **Static Novelty** | Accuracy | $\frac{\eta_{\mathcal{A}}+\eta_{\mathcal{N}}}{\|D\|}$ | Pimentel et al. [2014], Einarsdóttir et al. [2016], Kafkas and Montaldi [2018] |
| | Recall of Normal | $\frac{\eta_{\mathcal{N}}}{\eta_{\mathcal{N}}+\eta_{\mathcal{N}\to\mathcal{A}}}$ | |
| **Dynamic Novelty** | EN_Accuracy | $\frac{A_0+A_n}{\|D\|}$ | Masud et al. [2013], Mu et al. [2017], Zhu et al. [2018], Chen et al. [2008], Masud et al. [2009] |
| | F-measure | $\frac{2*P*R}{P+R}$ | |
| | Miss New | $\frac{\eta_{\text{new}\to\text{old}}}{\eta_{\text{new}}+\eta_{\text{new}\to old}}$ | |
| | False New | $\frac{\eta_{\text{old}\to\text{new}}}{\eta_{\text{old}}+\eta_{\text{old}\to\text{new}}}$ | |
| | Global Error | $\frac{\eta_{\text{new}\to\text{old}}+\eta_{\text{old}\to\text{new}}}{\|D\|}$ | |
| | Correct Between Known | Accuracy between known instances | |
| **Outlier** | Number of outliers detected | | Campos et al. [2016], Radovanović et al. [2014], Dang et al. [2014] |

*Definitions of the variables are shown in the next page.*

Table 2.4: Description of the variables used in Table 2.3.

| Variable | Description |
|---|---|
| $\|D\|$ | Number of instances. |
| $L$ | Length of the Time Series. |
| $t_x^*$ | Time at which the prediction is made. |
| $\eta_{\mathcal{A}}$ | Number of instances correctly classified from the abnormal class. |
| $\eta_{\mathcal{N}}$ | Number of instances correctly classified from the normal class. |
| $\eta_{\mathcal{A}\rightarrow\mathcal{N}}$ | Number of instances from the abnormal class classified as normal. |
| $\eta_{\mathcal{N}\rightarrow\mathcal{A}}$ | Number of instances from the normal class classified as abnormal. |
| $\eta_{\text{old}\rightarrow\text{new}}$ | Number of instances from a new class classified as an old class. |
| $\eta_{\text{new}\rightarrow\text{old}}$ | Number of instances from an old class classified as a new class. |
| $\eta_{new}$ | Number of instances correctly classified as a new class. |
| $\eta_{old}$ | Number of instances correctly classified as an old class. |
| $P$ | Precision of the emerging class. |
| $R$ | Recall of the emerging class. |
| $A_0$ | Number of known instances classified as an old label. |
| $A_n$ | Number of new instances classified as a new label. |

## 2.7 Validation of the proposed assignment

In order to validate this proposal of assignment, an experiment has been carried out considering two different scenarios. In the first scenario, the most cited papers after the year 2000 have been gathered; while in the second scenario, the first search-results after 2014 have been considered. In both scenarios, for each paper, two terms are obtained. On the one hand, that used by the authors to describe the problem, and on the other hand, that which would have been assigned with our taxonomy. In this way, a confusion-like matrix has been formed for every scenario.

In order to retrieve these papers, Google Scholar, ACM Digital Library and IEEE Xplore search engines have been used individually. Hence, the experiment is replicated for each individual search engine. In this way, the possible differences between these three communities have been checked.

The goal of the experiment is two-fold. Firstly, we would like to validate the presented proposal of assignment of terms to learning scenarios, and check when it matches the majority of the literature papers. Secondly, we would like to identify the most frequently confused learning scenarios between pairs of terms. Finally, we have also tested if the confusion varies between different communities and, hence, different search engines have been considered.

According to the confusion matrix of the most cited papers (Tables 2.5, 2.7 and 2.9), the terms used to describe the different types of abnormalities mostly match our proposal of assignment. However, in some situations, we have found discrepancies. Particularly, the highest discrepancies are found between the anomaly and rare event terms. In the case in which the authors use the anomaly term, it is frequently confused with our standardization of the rare event term. After checking the related literature, we realize that this happens when the problem has a temporal nature. Therefore, these problems would have been described with the rare event term regarding our proposal of assignment of terms. Similarly, these discrepancies are found in problems described with the rare event term but on the opposite side. When the novelty term is used by the authors to refer to the abnormalities of their problems, a minor set of papers are confused with our concept of anomaly term. In these works, instances of the novelty class are available during the training stage. Consequently, according to the presented proposal of assignment, their learning scenario is associated with the anomaly term. Finally, considering the outlier term, only a few situations are found in which the outlier detection learning scenario has been confused with the novelty detection one. In these mismatched works, a normality model is learned from labeled data. Then, instances non-conforming the normal behavior are rejected and considered outliers. Based on our proposal, this learning scenario corresponds with novelty detection.

In the second scenario with the first search-results of each term after 2014 (Tables 2.6, 2.8 and 2.10), a similar trend can be seen. However, there is some increase in the discrepancies. The confusion of the use of the terms novelty and anomaly is noticeable. For instance, the anomaly and the novelty problem descriptors have been confused in more situations than in the previous experiment with the subset of most cited works.

Regarding the different search engines, it can be seen that the mix-up is more prominent in the ACM community. Particularly, in the first 50 search-results (Table 2.8), it can be seen that the mix-up between the outlier term is considerably higher than in other communities. However, this trend can not be seen in the 50 most cited papers (Table 2.7).

Moreover, the novelty term also shows a slightly higher confusion in this community.

It can be concluded that the proposed assignment of terms to learning scenarios is supported by the literature. In addition, the confusion matrices reveal the mix-up between terms and learning scenarios. This clearly promotes the repetition of works and hinders the progress of the field. Furthermore, due to the popularity and increase of contributions in these term-related fields in recent years, this confusion is increasing. Therefore, we think that the standardization of the field is necessary and, with this review, we try to take a short step towards the solution of this mix-up.

Table 2.5: The confusion-like matrix formed from the results obtained from Google Scholar. For each term, the 50 most cited search-results (papers) have been analyzed after the year 2000. The terminology used by the authors is compared with respect to our proposal of assignment of terms to learning scenarios.

| | | **Problem descriptor used in the searched paper** | | | | |
|---|---|---|---|---|---|---|
| | | Rare Events | Anomaly | Novelty | Outlier | **Total** |
| **Our proposed approach** | Rare Events | 35 | 15 | 5 | 1 | 56 |
| | Anomaly | 13 | 24 | 16 | 1 | 54 |
| | Novelty | 0 | 7 | 26 | 3 | 36 |
| | Outlier | 2 | 4 | 3 | 45 | 54 |
| | **Total** | 50 | 50 | 50 | 50 | 200 |

Table 2.6: The confusion-like matrix formed from the results obtained from Google Scholar. For each term, the first 50 search-results (papers) after the year 2014 have been analyzed. The terminology used by the authors is compared with respect to our proposal of assignment of terms to learning scenarios.

| | | **Problem descriptor used in the searched paper** | | | | |
|---|---|---|---|---|---|---|
| | | Rare Events | Anomaly | Novelty | Outlier | **Total** |
| **Our proposed term** | Rare Events | 35 | 15 | 5 | 1 | 56 |
| | Anomaly | 13 | 24 | 16 | 1 | 54 |
| | Novelty | 0 | 7 | 26 | 3 | 36 |
| | Outlier | 2 | 4 | 3 | 45 | 54 |
| | **Total** | 50 | 50 | 50 | 50 | 200 |

Table 2.7: The confusion-like matrix formed from the results obtained from the ACM Digital Library. For each term, the 50 most cited search-results (papers) have been analyzed after the year 2000. The terminology used by the authors is compared with respect to our proposal of assignment of terms to learning scenarios.

| | | **Problem descriptor used in the searched paper** | | | | |
|---|---|---|---|---|---|---|
| | | Rare Events | Anomaly | Novelty | Outlier | **Total** |
| **Our proposed term** | Rare Events | 34 | 10 | 3 | 2 | 49 |
| | Anomaly | 13 | 24 | 12 | 4 | 53 |
| | Novelty | 2 | 6 | 24 | 4 | 36 |
| | Outlier | 1 | 10 | 11 | 40 | 62 |
| | **Total** | 50 | 50 | 50 | 50 | 200 |

Table 2.8: The confusion-like matrix formed from the results obtained from the ACM Digital Library. For each term, the first 50 search-results (papers) after the year 2014 have been analyzed. The terminology used by the authors is compared with respect to our proposal of assignment of terms to learning scenarios.

| | | **Problem descriptor used in the searched paper** | | | | |
|---|---|---|---|---|---|---|
| | | Rare Events | Anomaly | Novelty | Outlier | **Total** |
| **Our proposed term** | Rare Events | 30 | 15 | 4 | 6 | 55 |
| | Anomaly | 10 | 23 | 17 | 12 | 62 |
| | Novelty | 1 | 3 | 20 | 4 | 28 |
| | Outlier | 9 | 9 | 9 | 28 | 55 |
| | **Total** | 50 | 50 | 50 | 50 | 200 |

Table 2.9: The confusion-like matrix formed from the results obtained from the IEEE Xplore search engine. For each term, the 50 most cited search-results (papers) have been analyzed after the year 2000. The terminology used by the authors is compared with respect to our proposal of assignment of terms to learning scenarios.

| | | **Problem descriptor used in the searched paper** | | | | |
|---|---|---|---|---|---|---|
| | | Rare Events | Anomaly | Novelty | Outlier | **Total** |
| **Our proposed term** | Rare Events | 24 | 15 | 6 | 4 | 49 |
| | Anomaly | 16 | 25 | 8 | 5 | 54 |
| | Novelty | 5 | 2 | 25 | 5 | 54 |
| | Outlier | 5 | 8 | 11 | 36 | 60 |
| | **Total** | 50 | 50 | 50 | 50 | 200 |

Table 2.10: The confusion-like matrix formed from the results obtained from the IEEE Xplore search engine. For each term, the first 50 search-results (papers) after the year 2014 have been analyzed. The terminology used by the authors is compared with respect to our proposal of assignment of terms to learning scenarios.

| | | **Problem descriptor used in the searched paper** | | | | |
|---|---|---|---|---|---|---|
| | | Rare Events | Anomaly | Novelty | Outlier | **Total** |
| **Our proposed term** | Rare Events | 30 | 17 | 8 | 2 | 57 |
| | Anomaly | 11 | 24 | 11 | 4 | 50 |
| | Novelty | 1 | 1 | 21 | 5 | 28 |
| | Outlier | 8 | 8 | 10 | 39 | 65 |
| | **Total** | 50 | 50 | 50 | 50 | 200 |

## 2.8 Conclusions

In this chapter, we have underlined those genuine characteristics of each rare event, anomaly, novelty and outlier terms that are shared by the majority of the papers in the literature and have been assigned to a learning scenario. In order to do that, we have reviewed the different aims of each paper, the characteristics of the input data and the most representative techniques used in each rare event, and anomaly and novelty detection works. Each term has been accompanied with a set of illustrative applications to highlight the different learning scenarios. We have argued that the learning scenarios associated to the reviewed terms can be formalized under a supervised classification framework. Finally, we hope that the discussion with the closely related outlier term can enrich the comprehension of each scenario. Finally, the main characteristics of terms and problems have been summarized in Table 2.11. In this table, both the features related with the available data and the characteristics of the problem have been distinguished.

With this work, we take a short step towards the standardization of the rare event, anomaly, novelty and outlier terms. We think that our proposed assignment of terms to learning scenarios can help to resolve the muddle which hinders the progress in the term-related fields. Also, we think that the standardization of the terms and learning scenarios can strongly help to improve the progress in the field by letting the

| **Relative to** | **Characteristics** | **Rare Events** | **Anomaly** | **Novelty** | **Outlier** |
|---|---|---|---|---|---|
| Data | Temporal data | Yes | No | No | Possible |
| Data | All classes represented in training data | Yes | Yes | No | - |
| Problem | Unbalanced Classification | Yes | Yes | Possible | - |
| Problem | Supervised Classification | Yes | Yes | Yes | No |

Table 2.11: Summary of the principal characteristics, extracted from the literature, of the reviewed terms and learning scenarios.

community (and especially young, newcomer researchers) to easily find what they are looking for, and by avoiding the repetition of works.

# 3

# SNDProb: A Probabilistic Approach for Streaming Novelty Detection

In this chapter we focus on the Streaming Novelty Detection (SND) problem. In SND, new classes may emerge and disappear throughout a stream. Hence, the model must provide accurate predictions to the newcomer instances and also, discover new emerging classes. We propose a new methodological solution based on probability distributions that deals with the Streaming Novelty Detection (SND) problem. Concretely, we propose a novel solution based on a Gaussian mixture distributions in which each mixture component models one class.

## 3.1 Introduction

With the rapid growth of the Internet of Things (IoT), massive sensor data are available, creating new demanding problems for both researchers and engineers. Many of these problems have a temporal nature and evolve during time. For instance, in agriculture, plants are constantly monitored by a large variety of sensors. These sensors gather information about the moisture, air pressure and temperature, among other factors. The plants can be affected by some well-known plagues. However, in some situations, a certain plague can change (drift), be eradicated (disappear) and a new one can be discovered (emerge). This evolving learning scenario is known in the literature as streaming novelty detection, evolving classes or concept evolution [Masud et al., 2013, Mu et al., 2017, 2018, Haque et al., 2016a,b].

In streaming novelty detection, a model is initially learned from a fully-labeled dataset. Afterwards, instances arrive in a streaming fashion. The model classifies the instances among the learned classes. At the same time, the model is updated with the newcomer instances and their newly made predictions. Once in a while, an instance does not belong to any learned class and it is inserted into a buffer. When the buffer is full, an update process that searches for new classes is run. At this point, the model is updated with the new class or classes and the predictions of the buffered instances are output. A general pipeline of the framework can be seen in Figure 3.1.

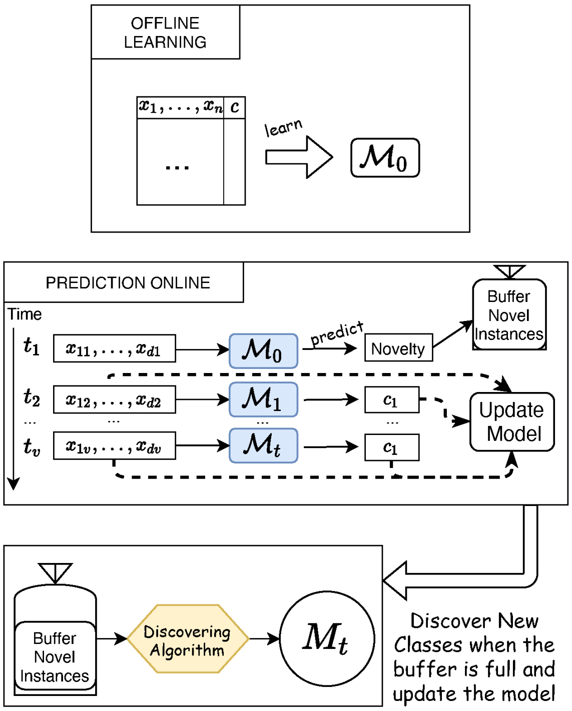


Fig. 3.1: General pipeline of a streaming novelty detection algorithm.

Streaming novelty detection can be seen as a combination of different learning scenarios such as *streaming classification* and *novelty detection*. In *streaming classification* unlabeled instances arrive either in batches or one by one, and not necessarily in equally-spaced time intervals. The aim is to classify these instances considering that changes may occur during the stream, the so-called concept drift [Lu et al., 2019, Alippi et al., 2017]. Therefore, the model needs to adapt to those changes in order to give accurate predictions Gomes et al. [2019]. In static *novelty detection* [Carreño et al., 2020, Pimentel et al., 2014, Khan and Madden, 2014], the problem consists of discarding the observations that do not follow a given (learned) pattern. Briefly, given a fully labeled dataset, a model is learned. At the prediction stage, instances are either classified as a known class or discarded [1].

Note that in Masud et al. [2013], Sun et al. [2016], Masud et al. [2009], Abdallah et al. [2016], a problem similar to the one tackled in this chapter is presented. However, at some point of the stream, true labels of the predicted instances are supplied, and the model is updated accordingly. This scenario is called class incremental learning in the literature, and it is different to the streaming novelty detection approached here. Besides, other works such as Haque et al. [2016a,b], Abdallah et al. [2016] address a semi-supervised streaming novelty detection problem where a limited amount of labeled instances to maintain and update the classification model are provided. Finally, approaches can be found from the fuzzy set theory. Particularly, da Silva et al. [2018] and da Silva and de Arruda Camargo [2020] propose an extension of the MINAS approach [Faria et al., 2016], in the fuzzy paradigm. In da Silva et al. [2018] and da Silva and de Arruda Camargo [2020], the micro clusters are learned using a Fuzzy C-Means clustering technique. Hence, the newcomer instances are associated to more than one micro-cluster, resulting on more flexible decision boundaries. These algorithms are compatible with the streaming novelty detection problem presented in this chapter. In spite of the solid results of these fuzzy algorithms, our choice is to remain in the supervised classification paradigm instead of moving to the fuzzy framework.

Other key works are related to the streaming novelty detection scenario addressed in this chapter. For instance in Tan et al. [2011], half-

[1] Note that the discarded instances can be used to some other specific objective such as failure detection or intrusion detection.

space trees are used to learn a model that assigns an anomaly score to newcomer instances. The model is continuously updated without any true label. In Lee et al. [2013], anomaly data is detected and over-sampled. Then a PCA analysis is done to discover the direction of the anomaly data in the feature space and hence, properly discover the new-comer anomalous points. In Deng [2016], outliers are detected using a combination of PCA techniques. However, none of these approaches discover new emerging classes. In this learning scenario addressed in this chapter it is assumed that the anomalous points have enough entity to be recognized as new classes. On the contrary to the aforementioned works, where these anomalous points are not subject to form new classes.

In order to deal with streaming novelty detection, the literature has commonly divided the solution into the following phases. The *offline phase* refers to the stage in which, given a fully-labeled dataset, an initial predictive model is learned. In the *online phase*, new instances arrive and the model classifies these instances either as a known class, or as a new class candidate. The new class candidate instances are stored in a fixed-sized buffer. As part of the *online phase*, a two-fold *update process* is run. On the one hand, it uses the classified instances as a known class to update the current models. In this way, the concept drift is tackled. On the other hand, when the buffer is full, the *update process* searches for new classes among the buffered instances.

This learning scenario presents a variety of challenges. For instance, the problem starts from a supervised classification problem at the offline phase, and, as long as the stream flows, only unsupervised data arrives. With these unsupervised data, new classes are discovered. Therefore, there is a transition from supervised to unsupervised classification. Besides, when a new class emerges, the number of instances that represent that class may be low with respect to the rest of classes. Therefore, while the predictions for a known class are supported by a possibly larger number of instances, the predictions for the new class are supported by fewer instances [Sun et al., 2016].

Two different methodologies can be seen in the literature that deal with streaming novelty detection. On the one hand, approaches that rely on a single classification model that is dynamically updated [Faria et al., 2016, Mu et al., 2017]. On the other hand, approaches that use an ensemble of classifiers have been developed [Masud et al., 2013, 2009, Garcia et al., 2019]. In this second strategy, a variety of models are used to represent the existing classes. Regarding the model update, at some point of the stream, the ensemble becomes outdated and a new model is learned to replace the outdated one. Both aforementioned approaches share a similar drawback. In order to speed up the computations, a compression of the data points is pursued. Particularly, clustering is carried out and the classes are represented by the union of (many) cluster boundaries. Each cluster is denoted by a center and a radius. In order to predict the class of an unlabeled instance, a (large) number of Euclidean distances are computed between the instance and the center of each cluster to check whether or not the instance belongs to any of the classes. Instances that do not belong to any class are stored in a fixed-sized buffer. In order to discover new classes, clustering is performed over the buffered instances when the buffer reaches its maximum capacity.

As previously explained, literature approaches are based on the idea of spherical clusters. These clusters are generated by leveraging the concept of cohesion and separation between classes. This methodology has some drawbacks. For example, it does not allow overlapping between classes. In addition, some literature works, such as Masud et al. [2013], Mu et al. [2017], do not consider the emergence of multiple classes in the same buffer.

In order to overcome these limitations, a parametric approach is proposed. The model consists of a mixture of Gaussian distributions. Each class is represented by a mixture component. No instances are kept throughout the stream except those stored in the fixed-sized buffer. These instances consists of observations that the model can not classify as any known class, i.e., unexplainable examples by the current model. In the *offline phase*, given a fully labeled dataset, a mixture of Gaussian distributions is learned. In the *online phase*, newcomer instances are predicted based on the probability of belonging to each class. The instances classified as a known class are used by the *update process* to update the parameters of the known mixture components. Besides, instances that can not be classified as any known class are stored in a fixed-sized buffer. When the buffer is full, new classes are sought among the buffered instances. If a new class is discovered, new components are added to the mixture and, at the same time, the existing mixture components are allowed to update their parameters. If no class is discovered, the parameters of the mixture are updated with the buffered instances. This process is done using the Expectation Maximization (EM) algorithm. However, both probability distributions and data are available at this point. In order to deal with this unusual situation, the probability distributions are weighted. This weighting process is done by a meta-regression model, which outputs an adequate weights for each mixture component.

This chapter is organized as follows: Section 3.2 refers to the proposed method. In Section 3.3, the experimental results are discussed. Finally, in Section 3.4, the conclusions are exposed.

## 3.2 Proposed method

In streaming novelty detection, instances arrive in an infinite stream fashion $DS = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_t, c_t), \ldots\}$, where $t$ represents the timestamp in which an instance arrives. Commonly, the literature has addressed this problem by dividing it into two different phases. The *offline phase* and the *online phase*. The later is then divided into the *prediction* and the *update process*.

### 3.2.1 Offline phase

In the *offline phase*, denoted as time $t = 0$ for simplicity, a probability distribution is learned from the data to create the initial classifier. In order to do that, a supervised set of observations $S_0 = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_m, c_m)\}$ is available. The learned probability distribution consists of a mixture of Gaussian distributions $\sum_{j=1}^{|C_0|} \phi_j f_j(\boldsymbol{\mu}_j, \Sigma_j)$, where $\boldsymbol{\mu}_j$ and $\Sigma_j$ represent the mean and covariance matrix of the $j^{th}$ mixture component $f_j$ at time $t = 0$, respectively; $\phi_j$ is the prior, or the mixing proportion, of the $j^{th}$ mixture component; and $|C_0|$ represents the number of classes at time $t = 0$. A graphical representation of this phase can be seen in Figure 3.2, and the pseudocode for this phase is exposed in Algorithm 3 ($u_j$ represents the number of instances belonging to class $j$).

The computational complexity of the offline phase is $\mathcal{O}(|C_0|md^2)$, where $m$ represents the number of observations of the training set, $|C_0|$

**Algorithm 3:** Pseudocode of the offline phase.

```
Input: DS=$\{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_m, c_m)\}$
for $j \in \{1, \ldots, |C_0|\}$ do
    $\boldsymbol{\mu}_j = \frac{1}{u_j} \sum_{i=1}^{u_j} \mathbf{x}_i$
    $\Sigma_j = \frac{1}{u_j} \sum_{i=1}^{u_j} (\mathbf{x}_i - \boldsymbol{\mu}_j)^2$
```

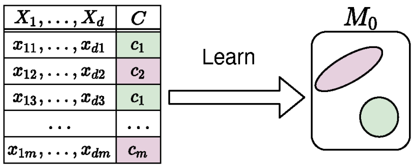


Fig. 3.2: Flowchart of the offline phase of the proposed parametric framework. A mixture of Gaussian distributions is learned from a labeled dataset.

are the number of classes at time $t = 0$, and $d$ are the number of features. The memory complexity of this phase corresponds to: $\mathcal{O}(d^2|C_0|)$.

### 3.2.2 Online phase

In the *online phase*, a sequence of unsupervised instances arrives, one instance at a time, and not necessarily in equally-spaced time intervals. For each instance $\mathbf{x}_t$, it is decided whether there is enough evidence to classify it as one of the current known classes or, otherwise, to insert into a buffer. Considering the proposed probabilistic framework, it is decided that an instance can not be classified if, for each mixture component $\mathcal{N}_j(\boldsymbol{\mu}_j, \Sigma_j)$, the instance is out of the sets $\Omega_j = \{\mathbf{x} | f_j(\mathbf{x}) \geq r_\alpha\}$ such that $p(\Omega_j) = 1 - \alpha$, for $j = 1, \ldots, |C_0|$; i.e., if the instance has a low probability of belonging to any mixture component, it is inserted in a buffer. This threshold $\alpha$ is a user parameter. The computation of the $\Omega_j$ set is generally a complex task. However, in the case of the Gaussian distribution, it can be computed using the Mahalanobis distance. Hence, the set can be equivalently rewritten as $\Omega_j = \{\mathbf{x} | d_M^2(\mathbf{x}, \mathcal{N}_j(\boldsymbol{\mu}_j, \Sigma_j)) \leq h_\alpha\}$ where the Mahalanobis distance is defined as:

$$d_M^2(\mathbf{x}, \mathcal{N}_j(\boldsymbol{\mu}_j, \Sigma_j)) = (\mathbf{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \tag{3.1}$$

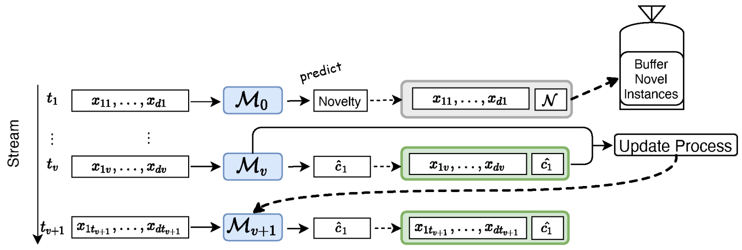


Fig. 3.3: Flowchart of the online phase of the proposed parametric framework. A stream of instances arrive, and for each instance a label is predicted. If the instance is predicted as novel, it is stored in the buffer. Otherwise, the instance, its predicted class and the current model are used to update the model for the next iteration.

Considering $d_M^2$ as a function of a Gaussian random variable, then $d_M^2 \sim \chi_d^2$, where $d$ represents the number of dimensions. Hence, computing the $h_\alpha$ is performed as $h_\alpha = \chi^2_{d,1-\alpha}$.

When an instance is inside $\Omega_j$, we assume that there is enough evidence to classify it as a known class, for some $j$. Hence, the prediction is made by assigning the $c_t = \arg\max_j \{f_j(\mathbf{x}_t), \forall j \ / \ x_t \in \Omega_j\}$. This can be computed using the Mahalanobis distance as:

$$c_t = \arg\min_j d_M^2(\mathbf{x}_t, \mathcal{N}_j(\boldsymbol{\mu}_j, \Sigma_j)) \tag{3.2}$$

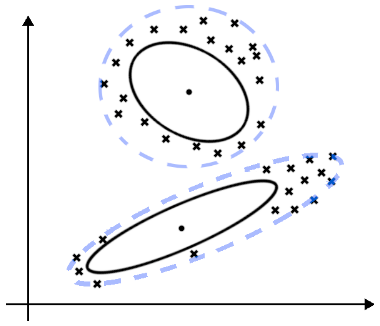
The pseudocode for this phase can be seen in Algorithm 4 ($\mathbf{p}$ represents the vector of predictions). A graphical representation of this phase is shown in Figure 3.3.

### 3.2.2.1 Update Process

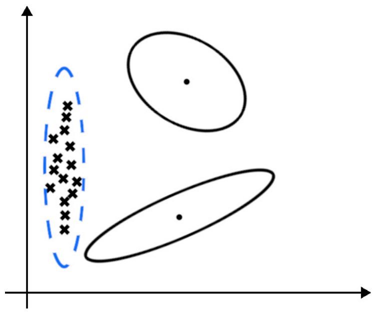
The goal of the update process is twofold. On the one hand, when an instance is classified as a known class $j$, the update process renews the parameters of that class $\{\boldsymbol{\mu}_j, \Sigma_j\}$ incrementally. On the other hand, the update process manages the disappearance and emergence of new

**Algorithm 4:** Pseudocode of the online phase.

```
Input: $\mathcal{M}_t$, $\mathbf{x}_t$, $\alpha$
for $j \in \{1, \ldots, |C_t|\}$ do
    $\mathbf{dst}_j = d_M^2(\mathbf{x}_t, \mathcal{N}_j(\boldsymbol{\mu}_j, \Sigma_j))$
if $\mathbf{dst}_j \geq \chi^2_{d,1-\alpha}$ for all $j$ then
    $\mathbf{buffer} = \mathbf{buffer} \bigcup \mathbf{x}_t$
else
    $c_t = \arg\min_j \mathbf{dst}_j$
    $\mathbf{p} = \mathbf{p} \bigcup c_t$
updateProcess$((\mathbf{x_t}, c_t), \mathcal{M}_t, \mathbf{p})$
```



(a) Extension of the current models.



(b) Detection of a new class.

Fig. 3.4: Two different situations that the update process could face during the discovering of new classes. Black points represent the buffered instances.

classes. In the former, a check is made to see whether a class can be considered as outdated. In order to do that, the number of predictions of that class, with respect to the number of predictions of other classes, is computed. If this ratio is low, according to a user defined parameter, the mixture component that represents the outdated class is removed from the mixture. Note that the proposed algorithm does not address the reappearance of removed classes. Therefore, the reappearance of a removed class will be treated as a new emerging class. Regarding the

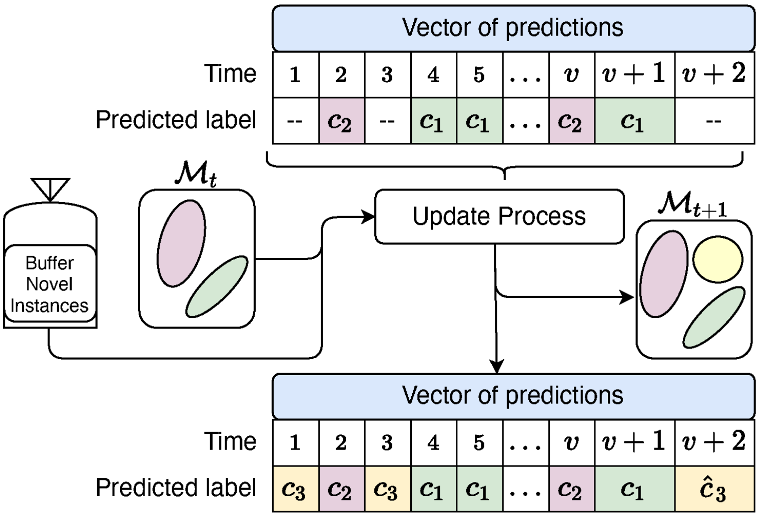

Vector of predictions
Time
1 2 3 4 5 ... $v$ $v+1$ $v+2$
Predicted label
-- $c_2$ -- $c_1$ $c_1$ ... $c_2$ $c_1$ --
$\mathcal{M}_t$
Buffer Novel Instances
Update Process
$\mathcal{M}_{t+1}$
Vector of predictions
Time
1 2 3 4 5 ... $v$ $v+1$ $v+2$
Predicted label
$c_3$ $c_2$ $c_3$ $c_1$ $c_1$ ... $c_2$ $c_1$ $\hat{c}_3$

Fig. 3.5: Flowchart of the update process in the online phase of the proposed parametric framework. Given the vector of predictions, the buffered instances and the current model, the update process outputs the new updated model and the predictions for the buffered instances.

emergence of a new class, it searches for new emerging classes among the buffered instances. Note that, depending on the threshold selected by the user, some instances may belong to one of the known classes. However, they are introduced into the buffer. Therefore, the buffered instances may not contain new emerging classes but a drift or an extension of existing ones (see Figure 3.4). A graphical representation of this part of the algorithm is shown in Figure 3.5.

In order to discover new classes, the EM algorithm is used. The maximum number of new emerging classes that are sought are selected by the user. One run of the EM is carried out for each possible number of components (classes). The Bayesian Information Criterion (BIC) is used to evaluate the configuration of each EM run and select the proper mixture components. This criterion penalizes the likelihood by the number of parameters of the model. Therefore, it finds a balance between the number of parameters and the likelihood. It is computed as:

$$BIC = p\log(n) - 2\log(\hat{L}) \tag{3.3}$$

where $p$ are the number of parameters of the model, $n$ the number of data points, and $\hat{L}$ represents the likelihood.

The EM algorithm has to deal with an scenario where both probability distributions that represent the known classes, and data (instances from the buffer) are given. We consider that this scenario can be addressed in two different ways. Firstly, in order to work only with data, the probability distribution associated to each class can be sampled. Nevertheless, this approach is not suitable in a streaming environment in which memory and time limitations exist. Furthermore, the amount of instances to be sampled from each distribution is also unknown. The second approach consists of assigning weights to the probability distribution associated to each class. This is the approach taken in this chapter. The weights are incorporated in the EM algorithm with the following adaptation in the maximization step to update the parameters of the $j^{th}$ class component:

$$\boldsymbol{\mu}_j^{(l+1)} = \frac{\overbrace{\sum_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\mu}_j^l, \Sigma_j^l)\mathbf{x}_i}^{\text{buffer}} + \overbrace{\sum_{k=1}^{|C_t|} w_k p(\boldsymbol{\mu}_k^0|\boldsymbol{\mu}_j^l, \Sigma_j^l)\boldsymbol{\mu}_k^0}^{\text{prob. distr.}}}{\sum_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\mu}_j^l, \Sigma_j^l) + \sum_{k=1}^{|C_t|} w_k p(\boldsymbol{\mu}_k^0|\boldsymbol{\mu}_j^l, \Sigma_j^l)} \tag{3.4}$$

$$\begin{aligned}\Sigma_j^{(l+1)} = {} & \frac{\sum_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\mu}_j^l, \Sigma_j^l)(\mathbf{x}_i - \boldsymbol{\mu}_j^{(l+1)})(\mathbf{x}_i - \boldsymbol{\mu}_j^{(l+1)})^T}{\sum_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\mu}_j^l, \Sigma_j^l) + w_j p(\boldsymbol{\mu}_j^0|\boldsymbol{\mu}_j^l, \Sigma_j^l)} + \\ & + \frac{\sum_{k=1}^{|C_t|} w_k p(\boldsymbol{\mu}_k^0|\boldsymbol{\mu}_j^l, \Sigma_j^l)\Sigma_k^0}{\sum_{i=1}^{n} p(\mathbf{x}_i|\boldsymbol{\mu}_j^l, \Sigma_j^l) + \sum_{k=1}^{|C_t|} w_k p(\boldsymbol{\mu}_k^0|\boldsymbol{\mu}_j^l, \Sigma_j^l)}\end{aligned} \tag{3.5}$$

where we denote as $\boldsymbol{\mu}_j^0$ and $\Sigma_j^0$ the parameters of the $j^{th}$ class at the beginning of the EM algorithm; and $w_j$ represents the assigned weight.

The result of the EM algorithm strongly depends on the weights assigned to each of the mixture components (see Figure 3.6). For instance, if the weight assigned to a component $j$ is high in comparison to the rest of the weights and number of buffered instances, the EM algorithm naturally reduces the covariance of that mixture component $j$. When the opposite situation occurs (when the weight for one mixture component $j$ is low with respect to other weights and number of buffered instances), the mixture component $j$ can significantly drift, considering the new emerging class instances as its own members. Therefore, in order to obtain a proper solution, a clever balance between the weights of the components and the size of the buffer is necessary.

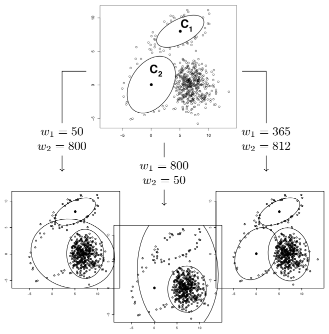


Fig. 3.6: The top figure shows the feature space when the buffer is full. The buffered instances are represented by black points. Three bottom scenarios show different results depending on the weights assigned to each class. The leftmost figure shows when $w_1 \ll w_2$. $C_1$ drifts and considers some of the new emerging class instances as its own class. The middle figure shows when $w_1 \gg w_2$, then the EM minimizes to almost null the covariance of $C_1$. Also, $C_2$ drifts and assumes new emerging class instances. The rightmost figure shows the adequate weights and its result.

Finding adequate weights depends on many factors. For instance, different weights are necessary when facing a concept drift or a new emerging class scenario (see Figure 3.6). Furthermore, aspects such as the ratio of instances predicted of each class, or the filling speed of the buffer strongly affect the adequate weights, among others. Therefore, due to the complexity of the selection of an adequate weight set, a meta-regression approach has been used for this task.

*A. Meta-regression approach*

This meta-regression model is pre-trained and supplied in the online phase with a cascade classification hierarchy. A diagram of the cascade hierarchy can be seen in Figure 3.7.

The meta-regressor uses a variety of features that are extracted from the stream. The stream is naturally split into epochs. Each epoch starts when the buffer is empty and ends when it is full. The aim of these features is to provide summarized information about the behavior of the stream. Particularly, they give information about the distribution of the instances throughout the current epoch and the degree of overlap among the classes. Hence, during each epoch, the following features are extracted:

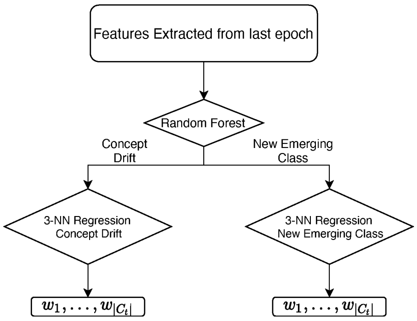

Features Extracted from last epoch
Random Forest
Concept Drift
New Emerging Class
3-NN Regression Concept Drift
3-NN Regression New Emerging Class
$w_1, \ldots, w_{|C_t|}$
$w_1, \ldots, w_{|C_t|}$

Fig. 3.7: Flow diagram of the meta-regression approach used to predict the proper weight set for the mixture components used in the EM algorithm.

- The number of instances classified as $c$, divided by the number of instances classified as a different class.
- The entropy of the mixture model at the time that the buffer reaches its maximum capacity. This feature represents the existing degree of overlapping between classes. The entropy Garcia et al. [2019] is calculated as

$$Ent(\mathcal{M}_t) = -\sum_{j=1}^{|C_t|}\sum_{i=1}^{|B|} t_{ij}(\mathbf{x}_i)\log t_{ij}(\mathbf{x}_i) \tag{3.6}$$

where $|B|$ represents the buffer size and

$$t_{ij}(\mathbf{x}_i) = \frac{\phi_k f_k(\mathbf{x}_i|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{h=1}^{|C_t|}\phi_h f_h(\mathbf{x}_i|\boldsymbol{\mu}_h, \Sigma_h)} \tag{3.7}$$

- For each class, the mean and variance of the inter-arrival times of two consecutive instances of the same class $c$ are computed. This feature represents the arrival frequency of instances of each class.
- For each class, the mean and variance of the density values that the Gaussian distribution of that class assigns to the instances of the buffer.

As a first stage of the classification hierarchy, a random forest classifier is used to differentiate between a concept-drift and (see Figure 3.4a) a new emerging class (see Figure 3.4b) scenario. The random forest classifier has been selected due to its demonstrated solid performance [Fernández-Delgado et al., 2014]. The second stage of the meta-regressor approach consists of running a 3-Nearest Neighbors classifier in order to obtain the adequate weight set. The $k$ of the NN classifier has been selected from empirical tests. The $k$-NN algorithm is a natural multi output classifier that empirically has shown good results in our tests. Two different datasets are used for the 3-NN approach, one containing only situations where new emerging classes have emerged, and a second dataset where concept-drift situations have occurred. The output of the random forest classifier selects the corresponding dataset for the 3-NN approach. As the reader might have realized already, as many regression models as the number of mixture components need to be supplied.

In order to generate the training data for the meta-regression approach and pre-train it, a battery of synthetically generated realistic random scenarios has been created. These scenarios are given to SNDProb for classification. When the buffer is full, the aforementioned features are extracted from the current epoch. Then, multiple weight sets are tested in a greedy-search approach. For each weight set (one weight for each mixture component), the EM algorithm is run and the buffered instances are predicted. The best weight set is selected based on the accuracy that the model obtains after releasing the buffer. This process is run many times and a consistent dataset is obtained.

Two different situations can occur when running the experiments. On the one hand, a situation where a new class emerges in the buffer and, on the other hand, a situation where concept-drift occurs. These two situations generate two different datasets.

In the generation of the datasets and in order to control the emergence of new classes, the arrival-strategies presented in the experimental section have been used. To manage the position of the classes in the feature space, and take into account the overlapping that may occur, random points have been sampled that are considered the means of the Gaussian distributions that form the classes. The covariance matrix is retrieved by sampling a Wishart distribution with random parameters. In the offline phase, 2500 instances of each of the offline classes are used to train the initial model. The experiment finishes when the buffer is released and a new emerging class is discovered. If the buffer is filled with extensions of known classes and hence, a concept drift needs to be modeled, the experiment continues until a new class emerges.

**Algorithm 5:** Pseudocode of the update process.

```
Input: $\mathcal{M}_t$, $(\mathbf{x}_t, c_t)$, metaregressor
if Buffer is full then
    features = extractFeatures($\mathbf{p}$, buffer, $\mathcal{M}_t$)
    $\mathcal{M}_t$ = removeOutdatedClasses($\mathcal{M}_t$, $\mathbf{p}$)
    $\mathbf{w}$ = metaregressor(features)
    $\mathcal{M}_{t+1}$, $\mathbf{p}$ = emAlgorithm(buffer, $\mathcal{M}_t$, $\mathbf{w}$)
else
    # Update model with the newly predicted $(\mathbf{x}_t, c_t)$
    $\mathcal{M}_{t+1}$ = updateParameters($\mathcal{M}_t$, $(\mathbf{x}_t, c_t)$)
return $\mathcal{M}_{t+1}$, $\boldsymbol{p}$
```

The computational complexity of the online phase, in the worst case scenario when the update process is run is computed as follows. Firstly, when predicting a newcomer instance, the inverse of the covariance matrix needs to be obtained, which has a cubic complexity being $\mathcal{O}(d^3|C_t|)$ where $d$ represents the number of features and $|C_t|$ the number of classes at time $t$. Secondly, the computational complexity for the extraction of features for the metaregression model can be seen as:

- Calculating the entropy of the current epoch: $\mathcal{O}(|C_t|d^2|B|)$, where $|B|$ represents the buffer size.
- Extracting the inter-arrival times of two consecutive instances of the same class: $\mathcal{O}(u^3)$, where $u$ represents the number of instances arrived in the current epoch.
- The cost of predicting the weights with the metaregression approach is $\mathcal{O}(M + DT)$, where $M$ represents the number of instances of the metaregressor, $D$ the depth of the random forest trees and $T$ the number of random trees.
- Predicting with the 3NN model has a computational complexity of $\mathcal{O}(K(|B|k))$ where $K$ are the different EM configurations that are considered, and $k$ the number of components of each EM run.

To sum up, the global computational complexity of the proposed algorithm is: $\mathcal{O}(d^3|C_t| + |C_t|((d^2|B|) + (uu^2)) + M + DT + K(|B|k))$. Besides, the memory complexity of this phase is $\mathcal{O}(d^2|C_t| + l)$, where $l$ is the length of the vector of predictions.

## 3.3 Experimental study

This section describes the experiments carried out in this study and analyzes the results. In these experiments, the proposed method is compared with two state-of-the-art techniques such as MINAS [Faria et al., 2016] and SENCForest [Mu et al., 2017]. A supervised version of the proposed algorithm SNDProb is also used (Supervised SNDProb). This illustrates a low classification error bound for the proposed SNDProb algorithm. Supervised SNDProb uses the true labels to update the model, i.e. stream instances come annotated. Instances that are classified as novelty are introduced into the fixed sized buffer annotated. When the buffer reaches its maximum capacity, new classes are discovered, and the existing ones are updated accordingly. Note that the buffer is supervised so the new class discovering process consists of computing the parameters of the Guassian mixtures of each of the buffered classes. The experiments have been run with both a battery of synthetic datasets and two public datasets commonly used in novelty detection and streaming literature. Particularly, the Cover Forest and Poker datasets from the UCI repository have been used. A brief description of these datasets is shown in Table 3.1. Regarding the synthetic scenarios, multiple variations have been used to test the proposed algorithm. The amount of classes ranges from 3 to 5, and the number of offline classes used is all the classes but one emerging class. The real-world Cover Forest dataset, has been modified according to [Mu et al., 2017]. Table 3.1 shows the number of features of the datasets, the number of examples, the number of classes that are available in each dataset, and the specific class labels that are used in the initial offline phase.

Table 3.1: Brief description of the used synthetic and real-world datasets.

| **Dataset** | **# Features** | **# Inst.** | **# Classes** | **# Offline Classes** |
|---|---|---|---|---|
| Cover Forest | 10 | 551443 | 4 | 2 |
| Poker dataset | 10 | 1017261 | 4 | 2 |
| 36 Synthetics | 2 | 17000 | 3, 4 or 5 | $\|C\| - 1$ |

# Inst. represents the number of instances in the dataset.

The parameters of the SNDProb, MINAS and SENCForest algorithms are exposed in Table 3.2. Regarding the proposed algorithm, SNDProb, the prediction threshold $\alpha$ is fixed to 0.02, the maximum number of new classes to appear at a time is set to 2 and the buffer size is set to 500 instances. These same parameters are set for the Supervised SNDProb algorithm. Regarding the parameters of the MINAS and SENCForest approaches, these have been taken from the original papers according to their best results and comparative study.

### 3.3.1 Performance metrics

In order to evaluate the performance of the classifiers in streaming novelty detection, classical scores from supervised classification are not representative enough in this streaming environment. In streaming novelty detection, the performance is not only based on the accuracy but aspects such as the misclassification among the known classes, or the misclassification among the new emerging classes are also important. Therefore, researchers have proposed alternative evaluation measures to deal with

Table 3.2: Parameters of the algorithms used in the experimental study.

| Algorithms | Parameters | Settings |
|---|---|---|
| **SNDProb** | $\alpha$ | 0.02 |
| | Buffer size | 500 |
| | Max new classes | 2 |
| | Class forgetting parameter | 0.05 |
| **MINAS** | Threshold past (buffer) | 2000 |
| | Min number of examples in the cluster | 20 |
| | $k$ = Number of micro-clusters | 100 |
| | Window size to forget outdated data | 1000 / k |
| | Threshold (see Faria et al. [2016]) | TV1 |
| | Clustering algorithm | clustream |
| **SENC Forest** | Subsample size | 100 |
| | Buffer size | 300 |
| | Number of trees | 100 |

this dynamic scenario. For instance, Masud et al. [2013] proposed as a measure the percentage of misclassified instances belonging to a new class classified as a known class (Miss New), or the percentage of misclassified instances from a known class classified as a new class (False new). In addition, Masud et al. [2013] used the classification error to analyze the results. Similarly, in Zhu et al. [2018], the average precision among all classes is used. Mu et al. [2017] proposed the F-measure and EN_accuracy evaluation measures. EN_Accuracy measures the misclassification among offline classes but does not measure the misclassification among the new emerging class instances. For instance, in a problem with two new emerging classes, when an instance from one new emerging class is classified as the other new emerging class, it is considered a success in this metric. On the contrary, if an instance belongs to one of the known classes and it is incorrectly classified as another known or new class, it is considered an error. F-measure evaluates the harmonic mean for new emerging classes. All the aforementioned measures are summarized in Table 3.3. Note that when computing the Miss New evaluation measure, the number of new classes can vary throughout the stream. Meaning that a new emerging class is no longer new after it is discovered. Therefore, when the number of new classes is 0, we output 0 for convenience.

In order to fairly compare the state-of-the-art techniques with SNDProb, some aspects must be considered. For instance, the SENCForest algorithm classifies any new emerging class with the same label. Hence, the classification error can not be computed in the case when there is more than one emerging class, as it is not possible to know whether or not a new class instance is correctly classified among the new classes. For example, consider a problem with 2 new emerging classes. When an instance of one emerging class arrives, it is predicted with the *new* class label. Later on, an instance from the other new emerging class arrives and it is predicted with the same *new* class label. Therefore, it can not be known if it is predicted as the former new emerging class or as the later. In order to overcome this issue, the EN_Accuracy evaluation measure has been used (see Table 3.3). Regarding the MINAS algorithm, it marks instances as *unknown* when the model can not classify them as any other known class. These instances are inserted into the buffer. Contrary to SENCForest and SNDProb algorithms, MINAS does not output a prediction for these buffered instances. Referring to the computation of the evaluation measures, these *unknown* instances do not compute as an error or success for MINAS. Therefore, we remark in the experiments how many of these *unknown* instances are marked by the MINAS algorithm. Furthermore, we remark the number of classified instances at each iteration of SNDProb, MINAS and SENCForest algorithms. Note that SNDProb and SENCForest do classify the buffered instances.

SNDProb and SENCForest output predictions for the buffered instances when the buffer is released. Hence, while these buffered instances are stored in the buffer, it is considered that they are not predicted and hence, they do not count as an error or success in the calculation of the evaluation measures.

All the evaluation measures shown in Table 3.3 are used in this experimental study. Their computation is made at every iteration in both synthetic and real datasets. However, in order to illustrate the comparative study, only the classification error and EN_Accuracy measures are presented. Note that, since in the synthetic datasets only one class emerges in the stream, the EN_Accuracy matches the classification Accuracy and hence, it corresponds with the inverse of the classification error. Therefore, only the classification error is shown in these experiments. The rest of measures can be found in the web based

Table 3.3: Summary of the state of the art evaluation measures.

| **Measure** | **Computation** |
|---|---|
| Classification Error | $\frac{\text{\# Incorrect classified instances}}{\text{\# Instances}}$ |
| EN_Accuracy | $\frac{A_n + A_0}{\text{\# Instances}}$ |
| F-measure | $\frac{2 * P * R}{P + R}$ |
| Miss New | $\frac{\text{\# Instances from a new class classified among KNOWN classes}}{\text{\# NEW class instances}}$ |
| False New | $\frac{\text{\# Instances from a KNOWN class classified as a NEW class}}{\text{\# KNOWN class instances}}$ |

$A_n$: # emerging class instances identified as new class.
$A_0$: # known class instances correctly classified.
$P$: Precision of the emerging class.
$R$: Recall of the emerging class.

supplementary material available in: https://andercarreno.shinyapps.io/SNDProb/. This website contains a RShiny Application that allows SNDProb to be interactively run over the synthetic datasets. To the best of our knowledge, this is the first GUI offering the possibility to test a streaming novelty detection algorithm. Furthermore, in this web application, it is possible to modify some parameters of the SNDProb before running the experiments, giving more flexibility to the user. The code and the data used in this experimental section can also be found on the following Github repository: https://github.com/andercarreno/SNDProb/.

### 3.3.2 Synthetic scenarios

A battery of synthetic datasets has been generated to test a variety of realistic situations, and control a set of different characteristics. The following aspects have been taken into account:

- the new class emergence timestamp
- the probability of arrival of instances of new or known classes at each iteration of the stream
- the overlap between the newcomer class and the existing ones
- the shape of the classes

In order to control the arrival rate of the instances, 6 different *arrival strategies* have been created. An *arrival strategy* consists of a variety of exponential functions, one per class, that models the probability of sampling one of the classes at a certain timestamp. These arrival strategies can be seen in Figure 3.8. The Figure shows the initial 15000 iterations. The same trend remains for the next iterations. This period is a stationary period that starts at iteration 5000, holding for the rest of iterations (up to 15000). Note that an arrival strategy can be seen as a sequence of probability distributions $\{p_t(C)\}_{t=1}^{\infty}$, one at each iteration of the stream $t$, where $|C|$ is fixed [Gama et al., 2014]. For new emerging classes, the probability is close to 0 at the beginning until they emerge at some point of the stream.

For the purpose of controlling the *overlapping* and *shape* of the classes, 6 different synthetic domains have been created. Figure 3.9 shows the different scenarios that are used for 3 classes. Instances from 2 classes are supplied at the offline phase to learn the initial model. Instances of the remaining class arrive according to one of the proposed arrival strategies in order to be discovered by the algorithm. As can be seen, 5 scenarios have been generated using a Gaussian distribution for each class. The mean and the covariance matrices have been adequately chosen to generate different overlapping and shape scenarios. In order to test our algorithm in a scenario that does not fulfill the assumption of Gaussianity of the SNDProb, Scenario 6 has been created. This last scenario has been generated by sampling each class from a 4-component mixture of Gaussian distributions.

A synthetic data stream consists of one of the 36 combinations created by one arrival strategy and one scenario (see Figures 3.8 and 3.9). These 36 combinations have also been extended to 4 and additionally to 5 classes, conforming another 72 synthetic data streams. All experimental results can be found in the supplementary material that is available at the following website: https://andercarreno.shinyapps.io/SNDProb/. From all these synthetic data streams, 3 have been selected to show the performance of the SNDProb and its comparison with MINAS and SENCForest approaches. The selection has made done due to their similarity with possible real-world situations. Particularly, in the first experiment, the 1st scenario and the 1st arrival strategy are used to provide a detailed comparative study between the algorithms. In the second experiment, the of the different arrival strategies over the 1st synthetic scenario is discussed. In the third experiment, the impact of the different arrival strategies over the 4th synthetic scenario is exposed. Finally, the non-Gaussian synthetic scenario with the 2nd arrival strategy is tested and the results are extensively discussed. In all synthetic data streams, the number of instances from each offline class used to learn the initial model is 2500 instances.

In each data stream, only one new emerging class appears throughout the stream. Although the proposed algorithm is capable of discovering more than one class at a time, the lack of evaluation measures that account for the emergence of more than an emerging new class motivates this criteria. Furthermore, SENCForest is not able to discover more than one class at a time and hence, the comparison would not be possible.

#### 3.3.2.1 Scenario 1 & arrival strategy 1: Non-overlapping classes with gradual emergence of the new class.

In Figure 3.10, a screenshot of the stream experimented in the 1st scenario and 1st arrival strategy is shown. The framed subfigures represent the iterations when the buffer is full, before the update process is run. The 1st scenario consists of 3 separated, round-shaped Gaussian distributions. According to the 1st arrival strategy exposed in Figure 3.8, a new class (blue line) emerges around iteration 1500. This class gradually emerges and at iteration 2000 there is a probability higher than 0.4 of sampling an instance from that new emerging class. At iteration 2276, the buffer reaches its maximum capacity and the model is renewed by the update process. At this point, the new class (blue class) is discovered. Referring to the arrival strategy (Figure 3.8), the first class drastically disappears around iteration 1700 (red class). At iteration 4626, when the buffer is full again, SNDProb considers one of the known classes (red class) as outdated and henceforth, the model stops considering it for further predictions.

Figure 3.11 shows the classification error and the number of predicted instances at each iteration of SNDProb, MINAS and SENCForest algorithms over the 1st arrival strategy and 1st synthetic scenario. The Supervised SNDProb approach illustrates the low classification error bound that SNDProb can achieve in this data stream. Table 3.4 shows the confusion matrix of the Supervised SNDProb, SNDProb, MINAS and SENCForest algorithms in percentage.

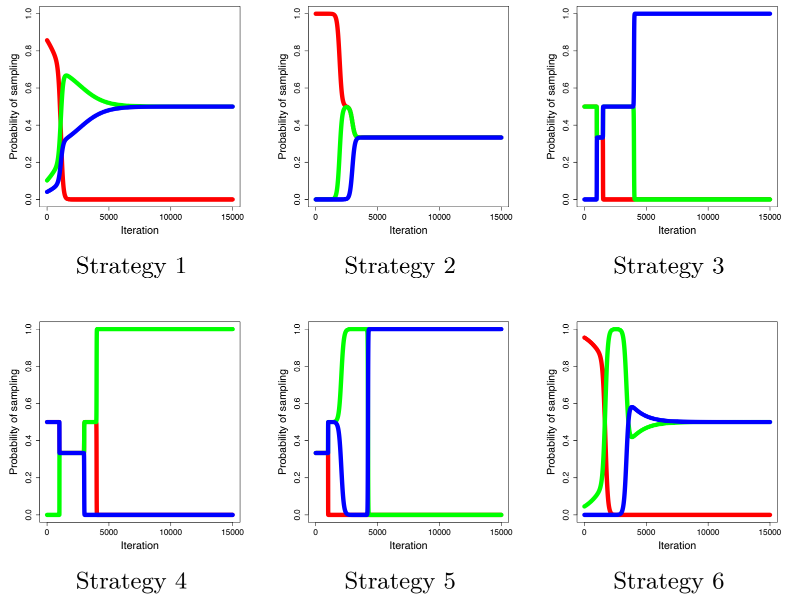
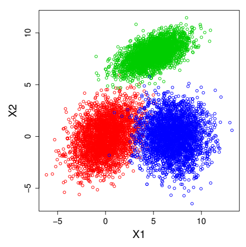
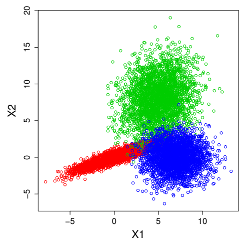


Fig. 3.8: Arrival strategies for 3 classes. With these strategies the arrival and disappearance of the classes are controlled. The lines represent the probability of sampling from that class at each iteration of the stream. Red and green classes are supplied in the offline phase to train the initial model. The blue line corresponds to the new emerging class.

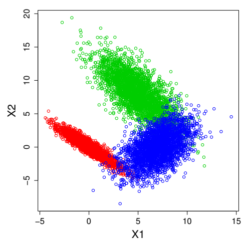
SNDProb outperforms state-of-the-art approaches offering the lowest classification error throughout the stream (see Figure 3.11b). Some events that correspond with the release of the buffered instances are worth analyzing. The new class emerges around iteration 1500 and SNDProb introduces instances from that class in the buffer. The buffer is released at iteration 2276. At this point, a tiny increase in the classification error is found for the SNDProb algorithm. This is due to the missclassification of some of the buffered instances. However, the new emerging class is perfectly discovered as shown in Figure 3.10, in subfigure 3.10c. The second key event corresponds with iteration 4626, when the buffer is again released. This time, the model considers the red class



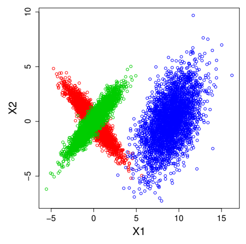
Scenario 1



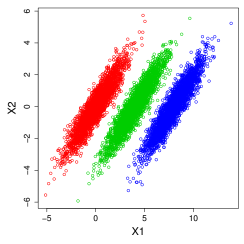
Scenario 2



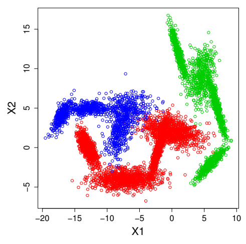
Scenario 3



Scenario 4



Scenario 5



Scenario 6

Fig. 3.9: Synthetic scenarios for 3 classes. Two classes are given at the offline phase (red and green) and one (blue) emerges throughout the stream.

as outdated and removes it from the model. At this point, the buffer consists of instances from green and blue classes and SNDProb correctly discriminates between these two. As a result, a decrease in classification error can be seen. At iteration 10912, another extension of blue and green classes can be found that the model correctly accounts for. Regarding the MINAS algorithm, an increasing trend is found in classification error throughout the stream. SENCForest shows an abrupt increase at the beginning of the stream until it stabilizes around iteration 1000. Afterwards, a nearly constant classification error curve can be seen.

In Figure 3.11a, the number of predicted instances at every iteration of the stream is shown. The supervised version of the SNDProb only shows one step, when the first class is discovered. The rest of the algorithms show several steps that correspond with the different buffer releases. These steps can also be seen in the classification error curves of SNDProb and SENCForest algorithms. Regarding the MINAS approach, the number of predicted instances is noticeably lower than for the rest of the algorithms. Particularly, MINAS does not output prediction for 2538 instances, 16.92% of the total number of instances.

Considering the number of times that the buffer is filled, a major difference is found between the SNDProb and SENCForest. Note that the buffer size is different for these two algorithms. However, even though the buffer size of SNDProb is almost twice that of the SENCForest buffer size, the number of times that the buffer is filled in the SENCForest does not correspond with the number of times (twice) that SNDProb fills its buffer. This behavior suggests that SNDProb is able to learn the shape of the classes with a lower number of instances than the SENCForest approach. This is a positive result, since while instances are in the buffer, their class is not predicted.

Table 3.4: Confusion matrix of the SNDProb, MINAS and SENCForest approaches for the 1st scenario and 1st arrival strategy. The values are shown in percentage and $B$ represents the instances stored in the buffer at the end of the experiment

| | | **Supervised SNDProb** | | | | **SNDProb** | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\hat{C}_1$ | $\hat{C}_2$ | $\hat{C}_3$ | $B$ | $\hat{C}_1$ | $\hat{C}_2$ | $\hat{C}_3$ | $B$ |
| **REAL CLASSES** | $C_1$ | 1.00 | 0.00 | 0.00 | 0.00 | 0.98 | 0.00 | 0.01 | 0.00 |
| | $C_2$ | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.98 | 0.00 | 0.02 |
| | $C_3$ | 0.02 | 0.01 | 0.97 | 0.00 | 0.01 | 0.00 | 0.95 | 0.03 |
| | | **MINAS** | | | | **SENCForest** | | | |
| | $C_1$ | 0.98 | 0.00 | 0.00 | 0.02 | 0.96 | 0.00 | 0.04 | |
| | $C_2$ | 0.00 | 0.96 | 0.00 | 0.04 | 0.00 | 0.97 | 0.03 | |
| | $C_3$ | 0.15 | 0.11 | 0.40 | 0.34 | 0.04 | 0.00 | 0.96 | |

### 3.3.2.2 Scenario 1: Impact of the different arrival strategies

In this experiment, the impact of the different arrival strategies over the 1st scenario (see Figure 3.9) is analyzed. Figure 3.12 shows the evalua-

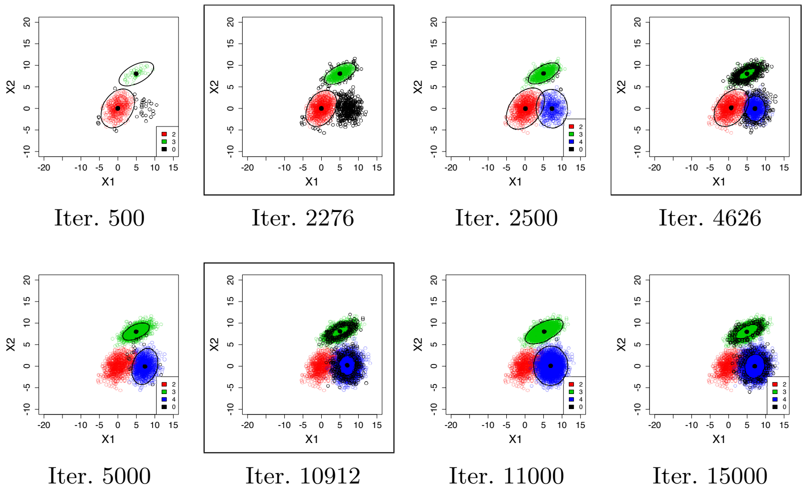


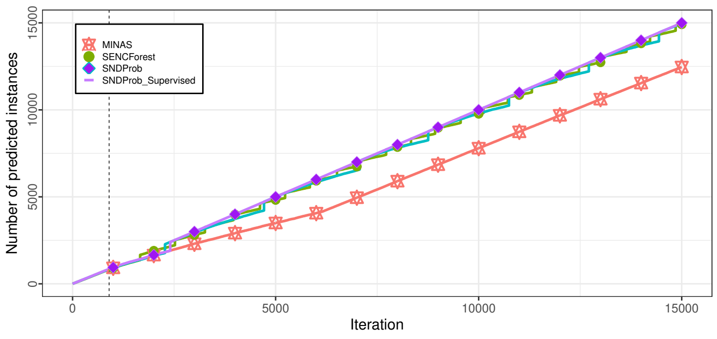
Fig. 3.10: Stream screenshots of the SNDProb in the 1st synthetic scenario with the 1st strategy. The red and the green classes are supplied in the offline phase and the blue class is discovered. The points colored in black are the buffered instances. The framed figures correspond to the iteration when the buffer is full; before the update process is run.

tion of the classification error of SNDProb, MINAS, and SENCForest algorithms over each of the proposed arrival strategies (see Figure 3.8)

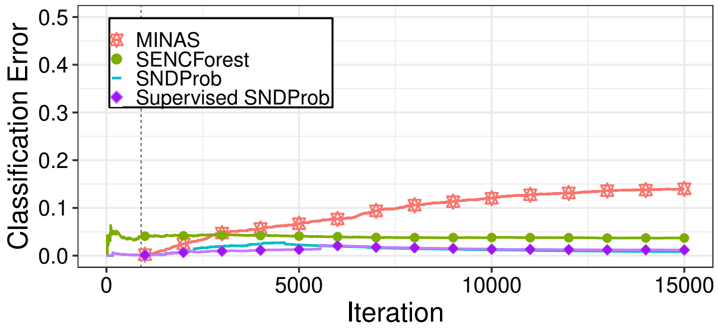
In each of the scenarios, the Supervised SNDProb has been run to gather a classification error bound that can be achieved by SNDProb in each of the data streams.

As can be seen, the performance of all the algorithms is severely affected by the different arrival strategies. SNDProb and SENCForest show the smallest degree of variability. However, SNDProb outperforms the rest of the approaches in every data stream except in the 4th data stream. Nevertheless, the differences in such data stream are negligible.

Regarding the MINAS approach, it needs to be mentioned that the number of instances that are not predicted by this algorithm vary from 9% to 19% of the total amount of 15000 instances. It can be concluded that the MINAS approach is the most affected algorithm by the different

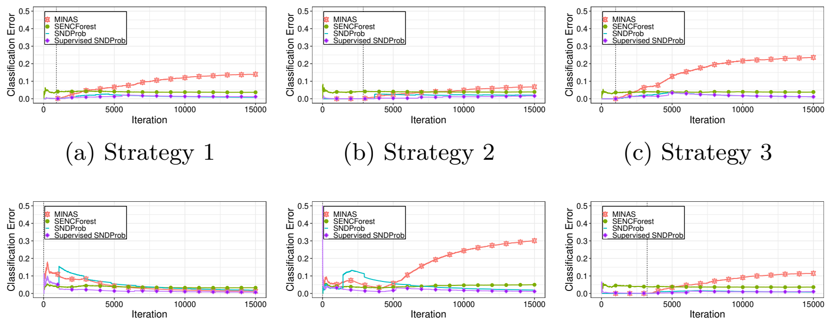


(a) Number of predicted instances throughout the stream



(b) Classification Error

Fig. 3.11: Number of predicted instances and classification error at each iteration of the stream of SNDProb, MINAS and SENCForest algorithms over the 1st scenario and 1st arrival strategy. The vertical dashed line shows when there is more than 0.1 probability of sampling from the new emerging class distribution. A Supervised SNDProb version is also presented to illustrate the low classification error bound that SNDProb can achieve in this data stream.



(a) Strategy 1 (b) Strategy 2 (c) Strategy 3

(d) Strategy 4 (e) Strategy 5 (f) Strategy 6

Fig. 3.12: Impact of the different arrival strategies in the performance of the SNDProb, MINAS and SENCForest algorithms over the 1st scenario. The vertical dashed line shows when the new class emerges. A Supervised SNDProb version is also presented to illustrate the low classification error bound that SNDProb can achieve in this data stream.

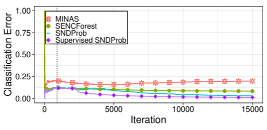
arrival strategies. In many data streams, such as the 1st, 2nd, 3rd, and 6th it can not even discover the new emerging class.

Considering the performance of the SENCForest approach, it can be concluded that the initial model learned by the SENCForest approach at the offline phase performs weakly due to the high classification error shown at the beginning of the experiments.

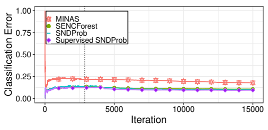
#### 3.3.2.3 Scenario 4: Impact of the different arrival strategies

In this experiment, the impact of the different arrival strategies over the 4th scenario (see Figure 3.9) is analyzed. Figure 3.13 shows the evaluation of the classification error of SNDProb, MINAS and SENCForest algorithms over each of the proposed arrival strategies (Figure 3.8).

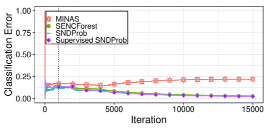
The 4th scenario (Figure 3.9) corresponds with 3 Gaussian shaped classes. Two of them are supplied in the offline phase (red and green): these classes overlap, forming a cross. The new emerging class (blue) is separated from the offline classes.



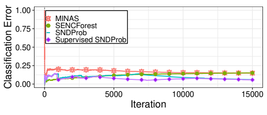
(a) Strategy 1



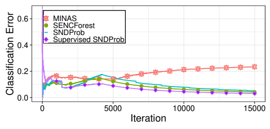
(b) Strategy 2



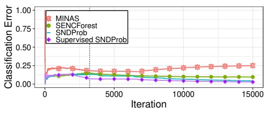
(c) Strategy 3



(d) Strategy 4



(e) Strategy 5



(f) Strategy 6

Fig. 3.13: Impact of the different arrival strategies in the performance of the SNDProb, MINAS and SENCForest algorithms over the 4th scenario. The vertical dashed line shows when the new class emerges. A Supervised SNDProb version is also presented to illustrate the low classification error bound that SNDProb can achieve in this data stream.

In each of the scenarios, the Supervised SNDProb has been run to gather a classification error bound that can be achieved in each of the data streams by the SNDProb algorithm.

As can be seen in Figure 3.13, the different arrival strategies strongly affect the performance of the algorithms. The least affected algorithm is SNDProb, which outperforms the rest of the algorithms by obtaining a classification error below 0.15 in every data stream.

The performance of the MINAS algorithm is worse than the rest of the algorithms used. In every data stream, the classification error is above the rest of the analyzed methods. It is worth mentioning that the number of not predicted instances by the MINAS approach ranges from the 10% to the 21% of the total amount of instances in the data stream.

Finally, the performance of the SENCForest approach is outperformed by the rest of the algorithms in every data stream except in the 5th. In this data stream, SENCForest beats the rest of the approaches but the difference is negligible.

#### 3.3.2.4 Scenario 6 & arrival strategy 2: Non-Gaussian scenario.

In Figure 3.14, a screenshot of the stream using the 6th scenario and 2nd arrival strategy is shown. Regarding the 2nd arrival strategy (see Figure 3.8), no class disappears. Instances of the different classes arrive sequentially, one after another, and once they have emerged, the instances from all the classes have equal probability of being sampled.

The classes of this scenario do not follow a Gaussian distribution. Therefore, SNDProb is not expected to perform as well as in Gaussian scenarios. Nevertheless, the classification error is still lower than 0.15 for every algorithm. As can be seen in the screenshots of the stream in Figure 3.14, the buffer is full of instances of the new emerging class at iteration 5122 (black points). Regarding the arrival strategy presented in Figure 3.8, the new class emerges around iteration 3000. In Table 3.5 the confusion matrix of the Supervised SNDProb, SNDProb, MINAS and SENCForest approaches is shown in percentage.

The Supervised SNDProb has been run to gather a low classification error bound that can be achieved in this data stream by the SNDProb algorithm. In this data stream, the assumptions of the probabilistic model are not fulfilled and therefore, this is reflected in the performance of both SNDProb and Supervised SNDProb. As can be seen, an increase in classification error is found after the new emerging class appears in the stream and even in the supervised version, the use of a single Gaussian distribution is not sufficient for dealing with this scenario. Nevertheless, the classification performance of the SNDProb is close to other competitors such as MINAS and SENCForest approaches.

In Figure 3.15, the evaluation measures of SNDProb, MINAS and SENCForest approaches are shown. In this scenario SENCForest offers the best results. It can be seen an increase in classification error at the beginning of the stream that suggests that the initial model learned on the offline phase performs weekly. Afterwards, the classification error remains constants throughout the stream. Furthermore, there is no increase when the new emerging class emerges. Regarding the MINAS approach, it does not output a prediction for 2290 instances, 15.27% of the data stream. Around the iteration 9000 an increase in classification error can be found that suggests that the model has drifted to a wrong one. Regarding SNDProb, the classification error is almost constant until the iteration 3000, when the new class emerges. Henceforth, the classification error increases. This behavior corresponds to the misclassification of this new emerging class instances as known classes. However, new emerging class is discovered in spite of the limitations of the Gaussian shape (see Figure 3.14c).

In Figure 3.15a the number of predicted instances at each iteration of the stream by SNDProb, Supervised SNDProb, MINAS and SENCForest algorithms is found. Several steps can be found in the SENCForest and SNDProb algorithms, meaning that the buffer is being filled and released several times.

Table 3.5: Confusion matrix of the Supervised SNDProb, SNDProb, MINAS and SENCForest approaches for the 6th scenario and 2nd arrival strategy. The values are shown in percentage and $B$ represents the instances stored in the buffer at the end of the experiment

| | | **Supervised SNDProb** | | | | **SNDProb** | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\hat{C}_1$ | $\hat{C}_2$ | $\hat{C}_3$ | $B$ | $\hat{C}_1$ | $\hat{C}_2$ | $\hat{C}_3$ | $B$ |
| **REAL CLASSES** | $C_1$ | 0.91 | 0.03 | 0.07 | 0.00 | 0.9 | 0.03 | 0.07 | 0.00 |
| | $C_2$ | 0.00 | 1.00 | 0.00 | 0.00 | 0.04 | 0.92 | 0.03 | 0.01 |
| | $C_3$ | 0.09 | 0.00 | 0.91 | 0.00 | 0.26 | 0.00 | 0.71 | 0.09 |
| | | **MINAS** | | | | **SENCForest** | | | |
| | $C_1$ | 0.97 | 0.00 | 0.00 | 0.03 | 0.98 | 0.01 | 0.02 | |
| | $C_2$ | 0.00 | 0.96 | 0.00 | 0.04 | 0.00 | 0.98 | 0.02 | |
| | $C_3$ | 0.10 | 0.11 | 0.30 | 0.49 | 0.04 | 0.00 | 0.96 | |

### 3.3.3 Cover Forest dataset

Table 3.6: Distribution of classes of the Cover Forest dataset.

| | **Online Classes** | | | |
|---|---|---|---|---|
| | **Offline Classes** | | | |
| **Class** | **1** | **2** | **3** | **7** |
| **# Instances** | 211988 | 283277 | 35702 | 20476 |
| **% of instances** | 38.44 | 51.37 | 6.48 | 3.71 |

* In the online phase all the instances are supplied

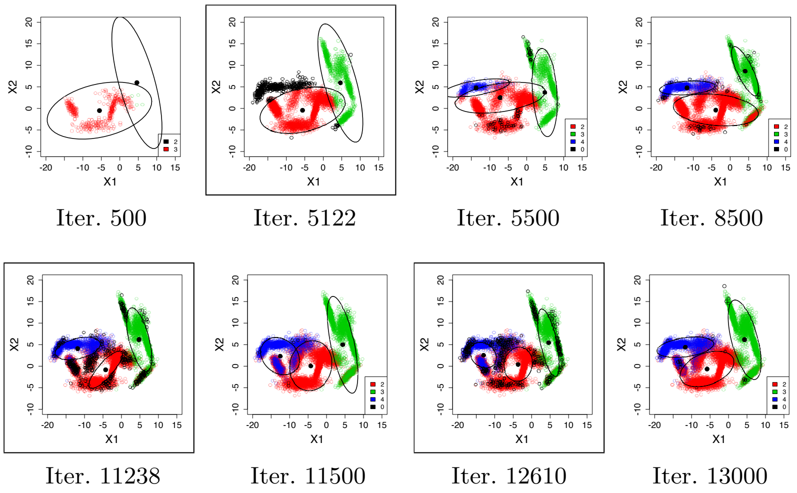


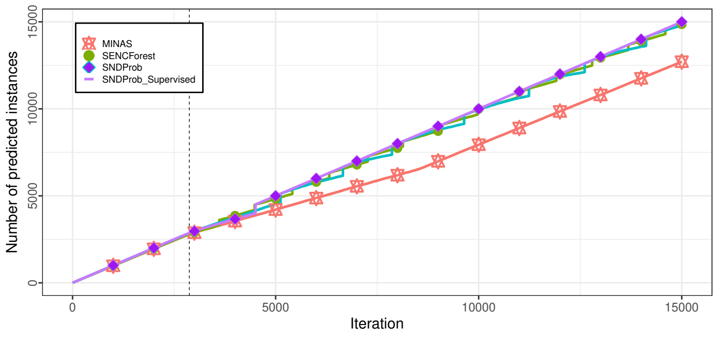
Fig. 3.14: Stream screenshots of the SNDProb in the 6th synthetic scenario with the 2nd strategy. The red and the green classes are supplied in the offline phase and the blue class is discovered. The points colored in black are the buffered instances. The framed figures correspond to the iteration when the buffer is full; before the update process is run.

This dataset is composed of cartographic variables of two American forests. Each instance corresponds to a variety of features extracted from every $30 \times 30$ meter cell of a map. The objective is to predict the cover forest species type of each instance. This is a public dataset that can be gathered from the UCI repository[1]. A summary of this dataset can be found in Tables 3.1 and 3.6.

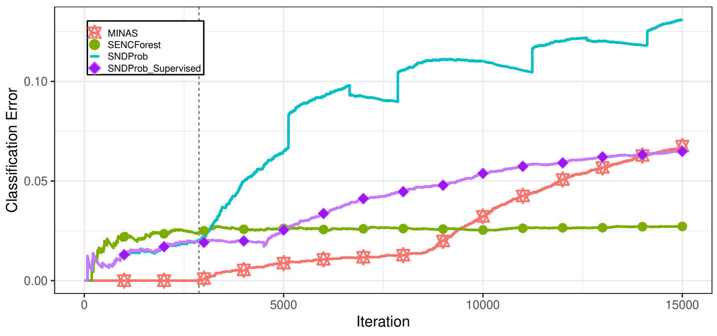
According to the modifications made in Faria et al. [2016], the number of instances from each offline class used to learn the initial model is 500.

In Figure 3.16, the evaluation measures of SNDProb, MINAS and SENCForest algorithms are shown. In addition, in order to illustrate a low Error bound that the SNDProb algorithm can achieve, the Supervised SNDProb approach has been used. In this problem, Super-

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



(a) Number of predicted instances throughout the stream



(b) Error

Fig. 3.15: Number of predicted instances and classification error at each iteration of the stream of SNDProb, MINAS and SENCForest algorithms over the 6$^{th}$ scenario and 2$^{nd}$ arrival strategy. The vertical dashed line shows when there is more than 0.1 probability of sampling from the new emerging class distribution. A Supervised SNDProb version is also presented to illustrate the low classification error bound that SNDProb can achieve in this data stream.

vised SNDProb has a minimal classification error throughout the entire stream. In Table 3.7 the confusion matrix of the Supervised SNDProb, SNDProb, MINAS and SENCForest approaches is shown in percentage.

As can be seen, both SNDProb and MINAS algorithms provide low classification error. Slight increases of the classification error can be seen when the new class emerges in both the MINAS and SNDProb curves. Regarding the SENCForest approach, it shows a decreasing trend in the classification error at the beginning of the experiment. This is due to the fact that only 500 instances from each offline class are given to the algorithm to learn the initial model. Therefore, the model is not robust enough at these first iterations of the stream. When instances from the first class arrive (the first class is an offline class), they are used to update the model and, therefore, the classification error is reduced. At iteration 211490, instances from the second offline class arrive. However, the classification error of the SENCForest algorithm increases.

Considering the EN_Accuracy shown in Figure 3.16c, SNDProb and MINAS show similar results. However, SENCForest shows a different trend in EN_Accuracy. Particularly, it can be concluded that the SENCForest classifies instances from known classes as new.

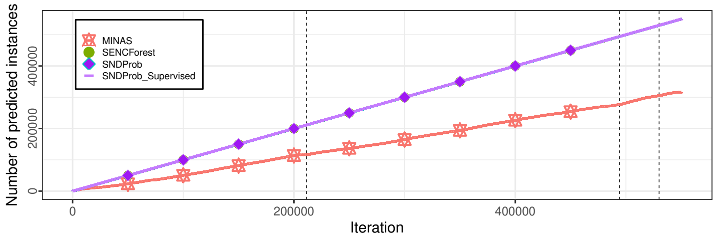
Figure 3.16a shows the number of predicted instances at each iteration by SNDProb, Supervised SNDProb, MINAS and SENCForest algorithms. The number of predicted instances by the MINAS algorithm largely differs from the SNDProb and SENCForest approaches. Specifically, MINAS does not output a prediction for 234079 instances, 42.5% of the dataset.

### 3.3.4 Poker dataset

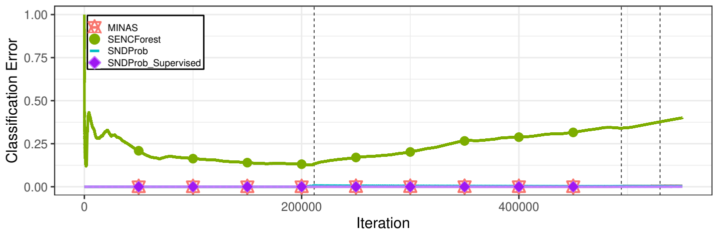
In this dataset, each record is an example of a poker hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes. The class represents the *poker hand*. This is a public dataset that can be obtained from the UCI repository[1]. A summary of this dataset can be found in Tables 3.1 and 3.8.

This dataset has been modified and only the first 4 major classes have been used. In the offline phase, 250000 and 200000 instances from the 1st and 2nd classes have been used respectively to learn the initial model. The rest of the instances are supplied in the online phase for prediction. Regarding the arrival order of the instances, they are randomly shuffled and supplied to the SNDProb for prediction. Note that

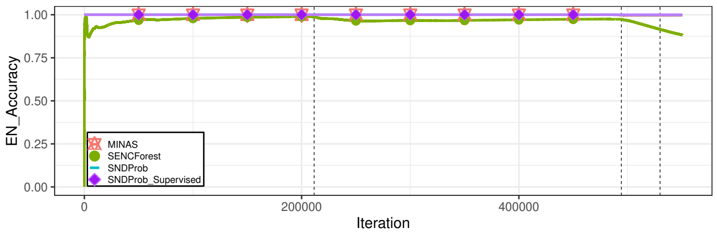
[1] https://archive.ics.uci.edu/ml/datasets/Poker+Hand



Number of predicted instances throughout the stream



Error



EN_Accuracy

Fig. 3.16: Number of predicted instances, classification error and EN_Accuracy at each iteration of the stream of SNDProb, MINAS and SENCForest algorithms over the Cover Forest real-world dataset. In this experiment, one class arrives at a time. Vertical dashed lines show when a class arrives in the stream. The new emerging classes correspond with the last 2 vertical dashed lines.

Table 3.7: Confusion matrix of the Supervised SNDProb, SNDProb, MINAS and SENCForest approaches for the Cover Forest dataset. The values are shown in percentage and $B$ represents the instances stored in the buffer at the end of the experiment.

| | | Supervised SNDProb | | | | | SNDProb | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\hat{C}_1$ | $\hat{C}_2$ | $\hat{C}_3$ | $\hat{C}_7$ | $B$ | $\hat{C}_1$ | $\hat{C}_2$ | $\hat{C}_3$ | $\hat{C}_7$ | $B$ |
| **REAL CLASSES** | $C_1$ | 0.02 | 0.98 | 0.00 | 0.00 | 0.00 | 0.99 | 0.01 | 0.00 | 0.00 | 0.00 |
| | $C_2$ | 0.01 | 0.99 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 |
| | $C_3$ | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| | $C_7$ | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.99 | 0.00 |
| | | **MINAS** | | | | | **SENCForest** | | | | |
| | $C_1$ | 0.55 | 0.00 | 0.00 | 0.00 | 0.45 | 0.55 | 0.13 | | 0.18 | |
| | $C_2$ | 0.00 | 0.56 | 0.00 | 0.00 | 0.43 | 0.39 | 0.72 | | 0.56 | |
| | $C_3$ | 0.00 | 0.00 | 0.02 | 0.78 | 0.20 | 0.01 | 0.15 | | 0.21 | |
| | $C_7$ | 0.00 | 0.00 | 0.02 | 0.51 | 0.47 | 0.06 | 0.00 | | 0.05 | |

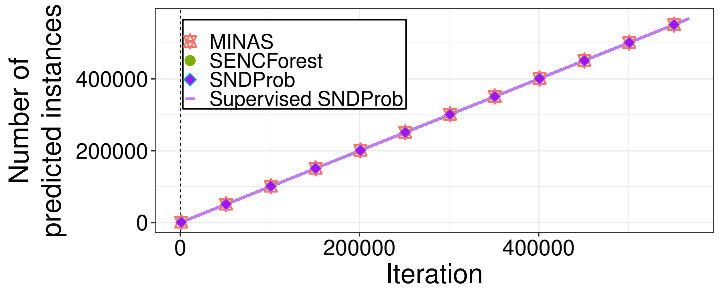
Table 3.8: Distribution of classes of the Poker dataset.

| | **Online Classes** | | | |
|---|---|---|---|---|
| | **Offline Classes** | | | |
| **Class** | **1** | **2** | **3** | **7** |
| **# Instances** | 513702 | 433097 | 48828 | 21634 |
| **% of instances** | 50.50 | 42.58 | 4.80 | 2.12 |

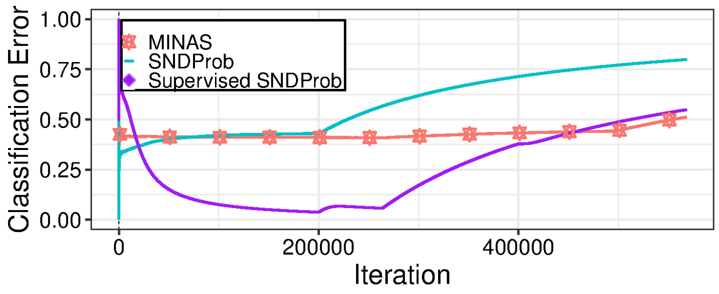
* In the online phase all the instances are supplied

2 classes emerge during the stream and since a random shuffle has been pursued, they are likely to appear at the same buffer release.

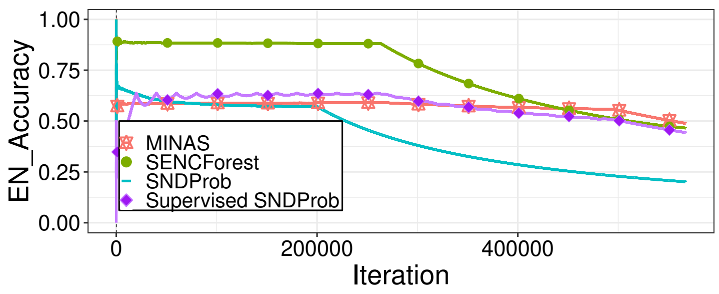
In Figure 3.17, the evaluation measures of SNDProb, Supervised SNDProb, MINAS and SENCForest algorithms are shown. The Supervised SNDProb algorithm has been used to illustrate a low classification error bound that the SNDProb algorithm can achieve. In this problem, SNDProb has a higher classification error with respect to MINAS approach. This is due to the cause that the assumptions of the probabilistic model are not fulfilled. However, the Supervised SNDProb shows a low classification error until the new emerging class emerges. Therefore it can be said that SNDProb can not discover the new emerging class correctly. In Table 3.9 the confusion matrix of the Supervised SNDProb, SNDProb, MINAS and SENCForest approaches is shown in percentage.



Number of predicted instances throughout the stream



Error



EN_Accuracy

Fig. 3.17: Number of predicted instances, classification error and EN_Accuracy at each iteration of the stream of SNDProb, MINAS and SENCForest algorithms over the Poker real-world dataset. In this experiment, multiple classes arrive at a time. A Supervised SNDProb version is also presented to illustrate the low classification error bound that SNDProb can achieve in this data stream.

Table 3.9: Confusion matrix of the Supervised SNDProb, SNDProb, MINAS and SENCForest approaches for the Poker dataset. The values are shown in percentage and $B$ represents the instances stored in the buffer at the end of the experiment.

| | | **Supervised SNDProb** | | | | | **SNDProb** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\hat{C}_1$ | $\hat{C}_2$ | $\hat{C}_3$ | $\hat{C}_7$ | $B$ | $\hat{C}_1$ | $\hat{C}_2$ | $\hat{C}_3$ | $\hat{C}_7$ | $B$ |
| **REAL CLASSES** | $C_1$ | 0.97 | 0.03 | 0.00 | 0.00 | 0.00 | 0.64 | 0.36 | 0.00 | 0.00 | 0.00 |
| | $C_2$ | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.98 | 0.02 | 0.00 | 0.00 |
| | $C_3$ | 0.00 | 0.67 | 0.33 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.96 | 0.04 |
| | $C_7$ | 0.00 | 0.01 | 0.08 | 0.92 | 0.00 | 0.00 | 0.00 | 0.00 | 0.85 | 0.14 |
| | | **MINAS** | | | | | **SENCForest** | | | | |
| | $C_1$ | 0.59 | 0.41 | 0.00 | 0.00 | 0.00 | 0.88 | 0.08 | | 0.04 | |
| | $C_2$ | 0.48 | 0.52 | 0.00 | 0.00 | 0.00 | 0.09 | 0.89 | | 0.09 | |
| | $C_3$ | 0.38 | 0.00 | 0.62 | 0.00 | 0.00 | 0.11 | 0.76 | | 0.13 | |
| | $C_7$ | 0.27 | 0.00 | 0.00 | 0.73 | 0.00 | 0.11 | 0.70 | | 0.19 | |

As can be seen, SNDProb has the highest classification error at the end of the data stream. An increase in classification error is found after the iteration 200000. This suggests that a new emerging class has appeared but the SNDProb is not able to properly discover it. Regarding the MINAS approach, a constant classification error is found throughout the entire stream. Finally, the SENCForest approach shows the lowest classification error at the beginning of the stream, but after the iteration 270000, an increase is found. Two conclusions can be extracted from this behavior. On the one hand, SENCForest learns a good initial model in the offline phase that provides a low classification error. On the other hand, the SENCForest approach is unable to discover the new emerging classes and hence, an increase in classification error is found.

Considering the EN_Accuracy in Figure 3.17c, SNDProb shows a similar pattern to the one shown in the classification error. After the iteration 200000, a decrease is found. This means that instances from new classes are incorrectly being classified as a known class. MINAS and SENCForest show similar results at the end of the stream. However, the SENCForest approach outperforms the rest of the algorithms in terms of the EN_Accuracy at the beginning of the data stream.

Figure 3.17a shows the number of predicted instances at each iteration by SNDProb, Supervised SNDProb, MINAS and SENCForest algorithms. In this data stream, all the algorithms predict all the instances.

## 3.4 Conclusions and future work

A novel probabilistic framework to deal with streaming novelty detection is proposed. It accounts for the current limitations of the state-of-the-art methods and offers a more flexible tool based on a robust mathematical background.

SNDProb uses a mixture of Gaussian distributions to model the set of classes in order to illustrate the proposed framework. Newcomer instances arrive in a stream fashion and they are predicted based on the probability of belonging to each of the classes. Instances for which the model cannot provide confident predictions are introduced into a fixed-sized buffer. For the purpose of discovering new emerging classes, an Expectation Maximization algorithm is used. In order to balance the relevance between the new data stored in the buffer and the probabilistic models that represent the known classes, a meta-regression approach has been developed. It has been tested that when the data fulfills the assumptions of the Gaussian distribution, SNDProb outperforms state-of-the-art algorithms such as MINAS and SENCForest. Furthermore it obtains competitive results in the case of non-Gaussian classes.

SNDProb provides a robust solution to the streaming novelty detection problem. However, there are some limitations that need to be taken into account. For instance, recurring concepts are not considered in this approach. When SNDProb does not predict instances as one of the known classes for a sufficient period of time, it considers this class as outdated and removes it from the known classes. However, it is known in the literature that classes can reappear throughout the stream. SNDProb does not account for this event and treats the newcomer class as a new one. Besides, SNDProb uses a single Gaussian distribution to model each of the classes. This may not be enough to model more complex shaped-classes. However, when the assumptions of the parametric model are fulfilled, SNDProb outperforms other literature approaches. Finally, the high number of parameters that SNDProb and the rest of the algorithms of the literature have, such as MINAS and SENCForest, is a relevant aspect to consider due to the difficulty of finding their appropriate values.

Regarding the experimental study, synthetic scenarios accompanied with a variety of arrival strategies have been presented. With these combinations of scenarios and arrival strategies, a wide range of realistic data streams have been tested, including significant features that has not been studied before in the related literature. Firstly, the scenarios set up a benchmark of different classes shapes and overlapping situations. Secondly, the arrival strategies control the arrival rate of the different offline and online classes throughout the stream. The experimental results confirm that the overlap between classes and the arrival strategies deeply affects the performance of the novelty detection algorithms. For the first time a streaming novelty detection algorithm is analyzed under the aforementioned conditions.

As future work, we plan to extend the current parametric approach to other probabilistic models. For instance, more complex probability models such as a mixture of Gaussians to represent each class is interesting and challenging future work, as well as other more complex probability models such as copulas models. Regarding the threshold value to introduce the novel instances of the online phase into the buffer, there is a challenging line of research in order to make it adaptive. This will ensure a better management of instances of known classes that are accidentally introduced into the buffer.

Finally, we think that there is a need in the community to establish robust evaluation metrics to properly evaluate this learning scenario and set the basis in order to compare the proposed algorithms. We think that other aspects such as the reactivity of the algorithm to discover new emerging classes, or the reactivity to recognize outdated classes are important aspects that should be considered in this scenario, for instance, with novel evaluation measures.

# 4

# Time Series Streaming Novelty Detection with Emerging New Classes

In this chapter, we extend the previous work shown in Chapter 3 to deal with time series data. In order to do that, we propose a novel methodological solution based on deep learning. In particular, we propose using a deep autoencoder and a Deep Support Vector Data Description (Deep SVDD) network to model each of the classes. Newcomer instances are checked whether they belong to the classes or not. In a positive case, the corresponding class is assigned and the model of that class is updated. Instances that do not belong to any of the learned set of classes, are stored into a fixed-sized buffer for further analysis. New classes are discovered by performing a hierarchical agglomerative clustering using Dynamic Time Warping (DTW) distance. Since this is done in an unsupervised manner, the model could get into a non-recoverable state if a new class concept is wrongly identified in the discovery process. To overcome this issue, we propose to maintain multiple parallel, inherently different models, that an expert could evaluate in hindsight. This allows us to use SND in real-world applications with expert knowledge. Our results show that the model can effectively discover new emerging classes and also provides a global explanation of the stream and the evolution of concepts within.

## 4.1 Introduction

Time-dependent data continues to be a hot-topic between both researchers and engineers. As a result, a variety of successful applications can be seen in the literature that deal with time series data [Dennis et al., 2019, Bai et al., 2021, Chen et al., 2021]. For instance, in cybersecurity, users are constantly monitored when using digital resources. These sequences of actions can be seen as time series of different length depending on the usage-time of the users. Commonly, this data is used by a previously learned model to detect anomalous behaviors, such as attackers or intruders. These problems are known in the literature as streaming time-series classification [Dennis et al., 2019]. Briefly, in streaming time series classification, a classifier $f$ predicts the class of newcomer time series and afterwards, the corresponding true label is supplied and used to dynamically update the current model. Formally, let $\mathcal{X} \subseteq \mathbb{R}^L$, $\mathcal{C} \subseteq \mathbb{Z}$ be the feature and label spaces, respectively; also, let $\mathcal{S}_t = \{(\mathbf{x}_t, c_t)\}_{t=1}^{\infty}$ represent the stream where $\mathbf{x}_t \in \mathcal{X}$ is the $t^{\text{th}}$ sequential data point with $\mathbf{x}_t = [x_{t,1}, x_{t,2}, \dots, x_{t,L}] \in \mathbb{R}^L$, $x_{t,l}$ denotes the $l^{\text{th}}$ time-step data value and, $c_t \in \mathcal{C}$ is the corresponding label of $\mathbf{x}_t$. Given $\mathcal{S}_t$ the aim is to dynamically update the current classifier $f : \mathcal{X} \to \mathcal{C}$.

As occurs in any streaming classification setting, the generative distribution at time $t$, $p_t$ on $\mathcal{X} \times \mathcal{C}$ may vary throughout the stream, making the task of accurately classifying the time series more difficult. Concretely, the model must react to concept-drift [Gama et al., 2014]. This concept drift can be seen as a change on the underlying generative distribution of the data over time. Formally, we say there is a concept-drift between two timestamps $t_0$ and $t_1$ if:

$$p_{t_0} \neq p_{t_1} \tag{4.1}$$

In SND another challenging event is considered: new classes can emerge or disappear throughout the stream. This can be seen as an abrupt change in $p_t$. The model needs to adapt to these changes by unsupervisedly learning new classes or removing the learned ones [1]. Formally, in a SND problem an initial offline dataset, $\mathcal{S}_0 = \{(\mathbf{x}_i, c_i)\}_{i=0}^{n}$ is used to learn an initial classifier that predicts newcomer instances

[1] Correctly removing outdated classes minimizes future potential classification error.

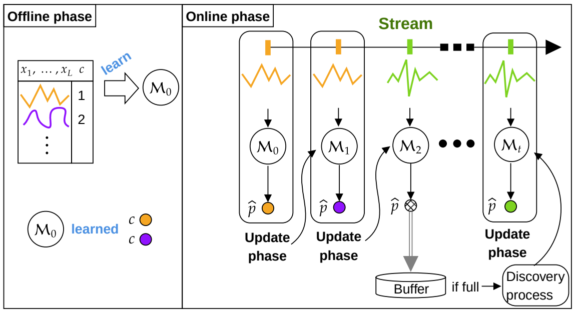


Fig. 4.1: A graphical representation of the offline and online phases of a SND approach. Different classes are represented with different colors. Each time an instance is classified, the instance and the prediction are used to update the current model. Instances for which the labels are unknown are introduced into a fixed-sized buffer.

among a set of known classes $C_t \subset \mathcal{C}$ or as novelties. The classifier is then defined as $f : \mathcal{X} \rightarrow \{C_t, -1\}$, where $-1$ denotes the novelty. Unlabeled instances arrive in a stream fashion $S_t = \{\mathbf{x}_t\}_{t=1}^{\infty}$. The classifier must predict the class of the newcomer instances $\mathbf{x}_t$ considering that the aforementioned changes in $p_t$ can occur at any time. For discovering new classes, the set $\beta = \{\mathbf{x}|f(\mathbf{x}) = -1\}$ is used when this reaches a fixed number of instances.

Referring to the previous motivating example in cybersecurity, in this SND setting, the system would not only classify the clients between anomalous or normal, but among the different categories that are previously known. Moreover, the system would discover new usage-patterns that may potentially be new types of attacks or vulnerabilities of the system[1]. In order to do that, a SND approach starts learning a model from a fully supervised dataset and then predicts instances that arrive in a stream fashion. Both the instances and the predictions are used to update the current model and tackle the concept drift. Once in a while, the model has not enough evidence to predict an instance among the previously learned set of classes. The model keeps aside in a

[1] Other normal usage-patterns can also be discovered.

fixed-sized buffer these unpredicted instances. When the buffer reaches its maximum capacity, new classes are sought among the buffered instances. This workflow is shown in Figure 4.1. Note that multiple classes can emerge when analyzing the buffer. As a result of such an analysis, both the updated model and the corresponding labels for the buffered instances are obtained. The workflow of the discovery process is illustrated in Figure 4.2.

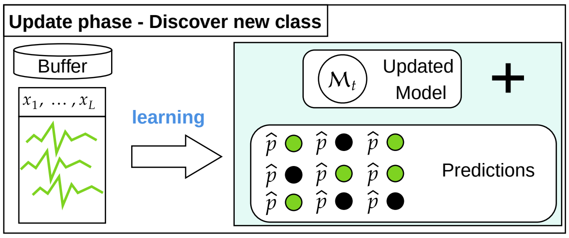


Fig. 4.2: A graphical representation of the output of the update process when the buffer is full of a SND approach. Here the model is updated and two new emerging classes have been discovered (black and green classes). As a result of the release, both the updated model and the corresponding predictions are output.

Regarding the data used in SND, the literature has proposed a set of approaches using non-temporal data [Faria et al., 2016, Carreño et al., 2022, Mu et al., 2017]. However, there are no approaches that explicitly consider time-dependent data. In this paper, we use fixed-length time series as instances that arrive in a stream fashion and we propose a new methodology based on deep neural networks to leverage from such time-dependency.

Nevertheless, SND approaches share a key drawback. Since the discovering process is pursued in an unsupervised manner, wrong decisions may potentially lead to a non-recoverable model. In other words, when the model wrongly infers the emergence of a new class, it is unlikely to recover from this mistake, in terms of forgetting or removing the wrongly discovered class. Similar conclusions apply to when a class is wrongly considered as outdated.

To address this problem, we propose a new solution towards a global explainability of the stream and a robust human-evaluation alternative for the SND solution. Specifically, we propose keeping multiple models whenever new classes are sought among the buffered instances. When the buffer is released, new models are created that consider the emergence of new classes. Besides, the model that assumes that no new class has emerged is also kept. All of them continue being dynamically updated throughout the stream. In order to avoid the exponential grow of the number of models, every time a new model is created, it is compared with respect to the rest and removed if it is considered to be duplicated. This framework allows to a) provide a global explanation of the stream when comparing the accuracy values of the different models at each iteration of the stream, when testing the models in a controlled scenario and, b) allow an expert to select the best SND model in a real-world applications where no labeled data is available throughout the stream. In this scenario, an expert could deal with a reduced set of samples and evaluate the model to select the most appropriated one.

To sum up, our paper contributes to the state-of-the-art in the following aspects:

- We propose a new methodological solution based on deep learning to the SND problem that leverages from time series data.
- We account for the exposed *non-recovery* limitation of SND problem by proposing the maintenance of multiple, inherently different, models at every iteration of the stream.
- By keeping multiple models, we also provide a global explanation of the stream when referring to the accuracy values of every model along the stream, in a testing scenario.

The rest of the paper is organized as follows. In Section 4.2, the related work of the SND problem is discussed. In Section 4.3, we precisely explain the new methodological solution based on deep learning that leverages from time series. Section 4.4 describes the proposed framework of keeping multiple models throughout the stream. In Section 4.5 we discuss about the experimental results and, in Section 4.6, the conclusions are exposed.

## 4.2 Related Work

There are some related scenarios to the SND problem that share a close objective but with different specifics. The differential key component is the availability of true class labels throughout the stream. Firstly, in supervised stream learning, true labels are always received after the prediction is made [Cutkosky, 2020]. This true label is used to update the model. Secondly, in Class Incremental Learning, annotated data is used for learning a new class and update the current model [Zhu et al., 2021, Rebuffi et al., 2017]. Thirdly, other works such as Masud et al. [2013, 2009] also update the current model to consider a new class but in an unsupervised manner. However, true labels are received after some period of time, and always after the class discovering phase. With this new information, the model gets corrected or updated if necessary. This scenario is also related with the late-labelling problem [Grzenda et al., 2020].

The literature has proposed several approximations that deal with the SND problem. Faria et al. [2016] proposed the MINAS approach; a non-parametric solution based on an ensemble of k-means clustering results. It models each of the classes by the union of a large number of micro-clusters that summarize the data points. For prediction, Euclidean distances to these micro-clusters are computed to test whether new points fit to any of the classes. In case they do not fit, they are introduced into a fixed-sized buffer and when it is full, a new clustering is formed from these instances, thus producing emerging classes.

Mu et al. [2017] proposed SENCForest, based on a combination of isolation trees, and k-means clustering. Briefly, a k-means clustering is pursued in each of the leaves of the isolation trees. For prediction, each of the instances is evaluated into the leaves and then within the clustering configurations of that leave. A similar procedure as described in the MINAS approach is done to test whether an instance belongs to one of the learned classes. New classes are incorporated to the current model by extending the isolation trees. One limitation of this approach is that only considers that a single new class can emerge in each buffer release.

Carreño et al. [2022] proposed a parametric approach by means of a mixture of Gaussian distributions. Each of the classes is represented by one mixture component and the predictions are made based on the probability of the instances to belong to each of the classes. For discovering new classes, an EM algorithm is used when the buffer is full.

All of these approaches are designed to deal with fixed-length vector inputs without temporal correlation among the variables (i.e., tabular data). This is not the case of our approach which considers that the instances are time series. Our novel methodology is based on deep learning to handle time series instances. Furthermore, literature solutions do not account for the potential possibility of deriving into a invalid non-recoverable solution due to the absence of supervised data. Hence, the usability of such approaches in real-world scenarios is limited. We account for this limitation by providing a novel framework of maintaining multiple models throughout the stream.

## 4.3 Methodology

The literature has commonly split the solutions of SND in two phases: *offline* and *online*. The *offline phase* learns a model from a fully-labeled dataset. In the *online phase*, unlabeled instances will arrive via a stream and the model needs to accurately classify these among the previously learned classes, or as novelty. If the instances are not predicted as novelties, both the instances and the newly made predictions are used to update the model via the *update process*. Moreover, when the user defined buffer is full of novel instances, the *update process* manages the buffer release and updates the model with the addition of new emerging classes if necessary. Note that, in this work, the instances are time series.

In the following sections, the proposed methodology is detailed in each of the phases that conforms a SND algorithm.

### 4.3.1 Offline phase

The proposed solution consists of learning an ensemble of models each representing one class (one model per class). For each class, a symmetric autoencoder is learned from a fully labeled dataset $S_0 = \{(\mathbf{x}_t, c)\}_{i=1}^{n}$ where each instance $\mathbf{x}_i$ is a time series. Two 1-dimensional convolution layers followed by hyperbolic tangent activation functions are used previous to a dense layer that maps the time series into an embedding space (bottleneck). The quadratic loss is minimized among the input and reconstructed series. In this problem we use convolution layers since the goal is to extract the representative sequences of each class, also known in the literature as shapelets [Li et al., 2021]. Although we have tested RNN layers, convolutional layers have exhibited better results.

As a second step, the encoder is fine-tuned to map the embedded features into a hypersphere. This approach is known in the literature as Deep SVDD [Ruff et al., 2018]. A Deep SVDD network consists on a deep learning approach inspired by kernel-based one-class classification and minimum volume estimation. Specifically, it trains a neural network while minimizing the volume of a hypersphere that encloses the network representations of the data. Let $\mathcal{F} \subset \mathbb{R}^h$ be the output space of $h$ hidden dimensions, $\phi_c(\cdot;\mathcal{W}_c) : \mathcal{X} \rightarrow \mathcal{F}$ is a neural network learned for class $c$ with $K \in \mathbb{N}$ hidden layers and set of weights $\mathcal{W}_c = \{\mathbf{W}_c^1, \mathbf{W}_c^2, \ldots, \mathbf{W}_c^K\}$, where $\mathbf{W}_c^k$ are the weights of layer $k \in \{1, \ldots, K\}$. That is, $\phi_c(\mathbf{x}, \mathcal{W}_c) \in \mathcal{F}$ is the feature representation of $\mathbf{x}$ given by the network $\phi_c$ with parameters $\mathcal{W}_c$. The aim of the Deep SVDD network is to jointly learn the network parameters $\mathcal{W}_c$ while minimizing a hypersphere with radius $R > 0$ and center $\mathbf{o}_c \in \mathcal{F}$ which is fixed to the medoid of the instances of class $c$ in this work. The Deep SVDD minimizes the following objective function:

$$\min_{R_c,\mathcal{W}_c} R_c^2 + \frac{1}{\nu n_c} \sum_{i=1}^{n_c} \max\{0, ||\phi_c(\mathbf{x};\mathcal{W}_c) - \mathbf{o}_c||^2 - R_c^2\} + \frac{\lambda}{2} \sum_{k=1}^{K} ||\mathbf{W_c}^k||_F^2 \tag{4.2}$$

where $\mathbf{o}_c$ and $R_c$ are the center and the radius of the hypersphere of class $c$, respectively. $\mathrm{W}_c = \{W^1, W^2, \ldots, W^K\}$ are the set of weights of the neural network, $n_c$ is the number of instances of class $c$, $\nu \in (0, 1]$ is a penalizing factor that controls the trade-off between the volume of the hypersphere and violations of the boundary. $|| \cdot ||_F$ denotes the Frobenius norm.

Deep SVDD offers two key solutions. Firstly, it works as a one class classifier by checking whether an instance fall inside the hypersphere or not. Secondly, it naturally provides an anomaly score based on the distance of the points to the center of the hypersphere:

$$s_c(\mathbf{x}) = ||\phi_c(\mathbf{x};\mathcal{W}_c^*) - \mathbf{o}_c||^2 \tag{4.3}$$

where $\mathcal{W}_c^*$ are the weights of a pretrained Deep SVDD network. Note that Deep SVDD was not proposed for an online environment where $p_t$ could potentially vary throughout the stream. Hence, the learned value of $R_c$ at some time $t$ could not be appropriate to classify instances at time $t+1$. Furthermore, in the SND setting, the absence of labeled data makes the task of updating the threshold, i.e. the radius of the hypersphere, more difficult. As a first approach, one might think to add a slack to the $R_c$ term. However, the intuition of adding a value in a deep output space is also difficult. Instead, we opted to model the anomaly score $s_c$ with a Gaussian distribution:

$$s_c \sim \mathcal{N}_c\left(\mu_c = \frac{1}{n_c}\sum_{i=1}^{n_c} s_c(\mathbf{x}), \quad \sigma_c^2 = \frac{1}{n_c}\sum_{i=1}^{n_c}(s_c(\mathbf{x}) - \mu_c)^2\right) \tag{4.4}$$

We learn this probabilistic framework from instances of the offline phase and we maintain it in the online phase. For clarification, from now on, we denote a model as a tuple of an autoencoder and Deep SVDD networks that represents one specific class. The next section exposes how the newcomer time series are predicted and how the model is updated to both tackle the concept drift and consider new emerging classes.

### 4.3.2 Online phase

Time series $\mathbf{x}_t$ arrive in a stream fashion, one at a time, $t = n+1, n+2, \ldots$ and not necessarily in equally-spaced time intervals. For each class, the anomaly score $s_c(\mathbf{x}_t)$ is computed. Considering the exposed probabilistic framework, an unlabeled, streaming instance will not be predicted if the instance is in set $\boldsymbol{\Omega}_c = \{\mathbf{x}_t | \mathcal{N}_c(\mathbf{x}_t) \geq r_\alpha\}$ such that $p(\boldsymbol{\Omega}_c) = 1-\alpha$, for all existing classes $c \in C_t$; i.e. if the instance has a low probability of belonging to any of the classes, it is inserted into a fixed-sized buffer. $\alpha$ is a user parameter that provides an intuitive threshold based on probability. The computation of the $\boldsymbol{\Omega}_c$ set is a difficult task. However, in the case of the Gaussian distribution, this can be computed with the Mahalanobis distance as $\boldsymbol{\Omega}_c = \{\mathbf{x}_t | d_M^2(\mathbf{x}_t, \mathcal{N}_c(\mu_c, \sigma_c)) \leq h_\alpha\}$ where the Mahalanobis distance is defined as:

$$d_M^2(\mathbf{x}_t, \mathcal{N}_c(\mu_c, \sigma_c)) = \frac{(\mathbf{x}_t - \mu_c)^2}{\sigma_c^2} \tag{4.5}$$

Considering $d_M^2$ as a function of a Gaussian random variable, then $d_M^2 \sim \chi_d^2$ where $d = 1$ in this univariate case. The computation of the $h_\alpha$ is performed as $h_\alpha = \chi^2_{d,1-\alpha}$.

Besides, when an instance is inside $\boldsymbol{\Omega}_c$, we assume that there is enough evidence to classify it as that class $c$. Therefore the prediction is made by assigning the $c = \arg\max_c\{\mathcal{N}_c(\mathbf{x}_t), \forall c/\mathbf{x}_t \notin \boldsymbol{\Omega}_c\}$. This is computed using the Mahalanobis distance:

$$c_t = \arg\min_c d_M^2(\mathbf{x}_t, \mathcal{N}_c(\mu_c, \sigma_c)) \tag{4.6}$$

When an instance is classified, the tuple $(\mathbf{x}_t, c_t)$ is used by the *update process* to update the model of that class $c$.

#### 4.3.2.1 Update process

The goal of the update process is twofold. Firstly, it updates the ensemble of models to tackle concept drift; secondly, it analyzes the buffer when is full to discover new emerging classes. Regarding the former task, the pair $(\mathbf{x}, c)$, where $c$ is the newly predicted class, is used to perform another step in the optimization of the weights of both the autoencoder and Deep SVDD networks of class $c$. According to the second task, complete-linkage hierarchical clustering is performed with DTW distance between the buffered time series.

At this point, an important decision needs to be taken. We need to determine the number of new emerging classes in the buffer (if any). Note that the initial offline ensemble is learned from a fully supervised dataset. When the buffer is full, only unsupervised data is available. The buffered data is likely to be different from the already known classes but, since concept drift might have occur, the buffer might be filled with both, novel instances from new emerging classes and incorrectly identified instances as novel. This increments the uncertainty on the process for determining the number of new emerging classes. Moreover, the number of new emerging classes is not restricted, meaning that more than one class could emerge in the same buffer release. Nevertheless, the proposed approach, as well as literature approaches [Faria et al., 2016, Carreño et al., 2022], assume a maximum number of classes that could emerge at the same time.

Incorrectly updating the current ensemble to considering a larger or lower amount than the real number of classes has a notable impact on the performance of the classifier. Furthermore, recovering from such incorrect solution is highly unlikely, [1] deriving, in the long time, in a useless classifier. Moreover, SND assumes that no true labels are gathered at any time of the stream; hence, there is no way to assess the validity of the ensemble in a real-world situation. In order to overcome this characteristic of the SND problem, what we call a *parallel universe* is proposed.

## 4.4 A Framework of Parallel Universes

To solve the problem of taking erroneous solutions we propose the use of parallel hypotheses or *parallel universes*. We maintain every new hypothesis of new emerging classes, such that an expert could evaluate in hindsight. Specifically, we dynamically maintain a tree of ensemble models; a graphical representation can be seen in Figure 4.3. At time $t$ a model that classifies among 2 classes is available and at time $t + v$ its buffer is full. The maximum number of new emerging classes to seek for is set to 1 in this example. Two possibilities are maintained: on the one hand, to keep the same model that considers $|C| = 2$ classes, while on the other, to learn a new model that considers one new emerging class ($|C| = 3$). A similar behavior can be seen at time $t + v + u$.

Clearly, the number of ensembles grows exponentially throughout the stream. In order to reduce such exponential growth, the newly created ensembles are compared with each other[2] and, if similar, duplicates are removed. This is not too different from the resampling step used to prevent particle generation in particle filters, e.g., in Martino et al. [2017]; however, in our case, ensembles are more complex than a typical 'particle', and we focus on efficiency via removing close-duplicates rather than only pruning away degenerate (low-likelihood) hypotheses.

In the proposed methodology based on neural networks, comparing two models is a non-trivial task that has been thoroughly researched in recent years [Bernstein et al., 2020, Chicco, 2021]. The proposed solution uses an ensemble of autoencoders and Deep SVDD networks; one pair for each class. In order to compare two ensembles that classify

[1] Recovering in terms of taking back the prior valid ensemble or to remove the incorrect new emerging class in an online manner

[2] Only the leaves of the tree need to be tested.

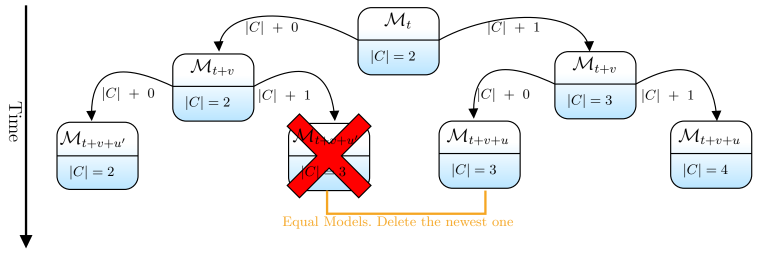


Fig. 4.3: A graphical representation of the *parallel universe*. Each model has its own buffer so it is analyzed at different times, creating multiple ensembles.

among the same number of classes, we randomly generate vectors (of the same size as the time series) and we provide them as time series to the autoencoders of the ensemble. We compute the Mean Squared Error (MSE) of the reconstruction error of the time series for each of the ensembles. If the difference in error among two ensembles is lower than a user defined threshold, we assume that these two ensembles have learned to reconstruct the same set of classes and hence, they are considered as duplicate ensembles. Therefore, we remove the newer one. Although this approach does not account for the exponential growth of the *parallel universe* framework, it notably reduces the number of ensembles kept.

The *parallel universe* framework offers two key characteristics. Firstly, an expert could evaluate the ensembles to avoid the effect of wrong *automatic* decisions, and select the most adequate one to dynamically continue learning the stream. Secondly, it offers a global explanation about the stream and evolution of the concepts within.

## 4.5 Experimental Results and Discussion

We evaluate the proposed method on two well known time series datasets from the Time Series Classification repository [Dau et al., 2018]. In order to realistically simulate a SND problem, we have removed some of the classes from the offline data and supplied afterwards in the online phase for class discovering. In Table 4.1, the datasets used and the number of time series from each of the classes provided at each phase is shown. Both the datasets and the code will be available upon publication in a public Github repository.

Table 4.1: Summary of the number of instances of each class provided in the offline and online phases for each of the tested time series datasets.

| **Dataset** | **Class** | **Offline phase** | **Online phase** |
|---|---|---|---|
| **CBF** | **1** | 230 | 80 |
| | **2** | 226 | 84 |
| | **3** | 0 | 310 |
| **BME** | **1** | 42 | 18 |
| | **2** | 46 | 14 |
| | **3** | 0 | 60 |

We compare our method with respect to a collection of diverse state-of-the-art approaches. Note that there are no algorithms in the literature that are specifically developed to deal with time-dependent data for the SND problem. Nevertheless, the time series are supplied as regular vectors to the competitors. We decided to compare with respect to the MINAS, SENCForest and SNDProb approaches as they are well known literature solutions for the SND problem. Accuracy is used as evaluation metric.

Evaluating SND approaches is a difficult task [Carreño et al., 2022]. The instances while are stored into the buffer have no assigned class label. However, when the buffer is released, a prediction for the buffered instances is provided. At this point, they take part in the evaluation measures. This criterion is followed by most of the literature approaches such as SENCForest and SNDProb algorithms, and we also follow it in our paper. In the case of MINAS, the buffered instances do not receive a class at any time, not when they are introduced into the buffer, neither when they are released. Hence, they do not compute in the evaluation measures. Note that this behavior implies that, at each iteration of the stream, the number of predicted instances and the number of iterations do not necessarily match. We also show the number of instances stored into the buffer at each iteration of the stream for each of the tested algorithms.

The proposed approach uses the *parallel universe* framework described in Section 4.4. Therefore, a set of accuracy lines is shown in Figures 4.4 and 4.5. Each accuracy line corresponds to one of the different ensembles/hypotheses. Note that since the creation of new ensembles is constrained to the buffer filling time, the starting point of these ensembles is different. In the plots, each ensemble is referred to as *root*, plus a set of numbers. Each position corresponds to one buffer release. Hence, *root200* corresponds to an ensemble with 3 buffer fills (3 numbers, 3 buffer releases). In the first buffer release, an ensemble with 2 new emerging classes was created. In the second buffer release, no new emerging class is considered, hence, the same ensemble is kept. This occurs again with the third buffer release.

### 4.5.1 Results on CBF dataset

Cylinder-Bell-Funnel is a simulated dataset in which time series instances follow one of 3 shapes (cylinder, bell, or funnel) plus noise which makes this classification task more challenging. Furthermore, the time series are misaligned. All series have the same length that corresponds to 128 values. The number of instances used in each of the phases of the SND problem are shown in Table 4.1. Instances of each class of the online phase arrive with equal probability.

All hyper-parameters of the compared algorithms have been set according to recommendations of the original papers, except: In MINAS we set the number of micro clusters per class to 10; and for SENCForest the number of subtrees per class to 10. The buffer size is set to 50 instances for every algorithm.

In Figure 4.4, the accuracy values and the number of buffered instances of the MINAS, SENCForest, SNDProb and the proposed algorithm are shown at each iteration of the stream. For our proposal, multiple accuracy lines can be seen as a result of the *parallel universe*. These ensembles can be dynamically evaluated by an expert to select the most appropriate hypothesis. The initial ensemble *(root000000)* fills its buffer 6 times. These correspond with the different accuracy peaks shown in Figure 4.4. The obtained accuracy tends to the 35% which corresponds to the distribution of the offline classes according to Table 4.1. In other words, it classifies properly the instances of the offline classes but it wrongly predicts the new emerging ones. The ensemble *root10* provides the best results. In this case, the ensemble fills the buffer twice. Firstly, the initial ensemble is split to consider one new emerging class. Secondly, the later fills its buffer again and this branch does not increase the number of classes. The rest of the ensembles that do not match the correct number of classes offer worse results that degrade during time.

SNDProb performs properly until the buffer is released. This suggests that almost all the instances are introduced into the buffer and when these are released, they are missclassified. Moreover, due to the meaningful drop in accuracy, it can be said that the SNDProb can not classify among any of the classes, either new or old.

The MINAS approach shows a constant increase in accuracy. However, it does not discover the new emerging class. Furthermore, note that MINAS does not predict the instances stored into the buffer. The same accuracy trend can be seen for the SENCForest approach.

### 4.5.2 Results on BME dataset

Begin-Middle-End is a dataset that consists of 3 classes. Each instance (of length 128) represents a bell, and the class indicates its position in the series (beginning, middle or end). The distribution of classes at each of the phases is shown in Table 4.1. Similar to the CBF dataset, the instances of the different classes of the online phase arrive uniformly.

The parameters of the algorithms have been set to default except of the buffer size, which has been fixed to 30 for all the algorithms.

Figure 4.5 shows that the proposed approach, offering a set of solutions to an expert, notably outperforms the other approaches when *root1* is selected in hindsight. The offline ensemble fills its buffer twice throughout the stream. This can be clearly seen in the peaks of the number of buffered instances throughout the stream.

SNDProb offers better results than MINAS and SENCForest approaches, but it only obtains 50% accuracy, suggesting that it does not discover any new emerging class, and evidenced by not a single buffer release.

The MINAS and SENCForest approaches show similar results. Particularly, it can be seen that MINAS increases, above the maximum buffer size, the number of buffered instances along the stream. This is due to the fact that the MINAS approach does not release the buffer:

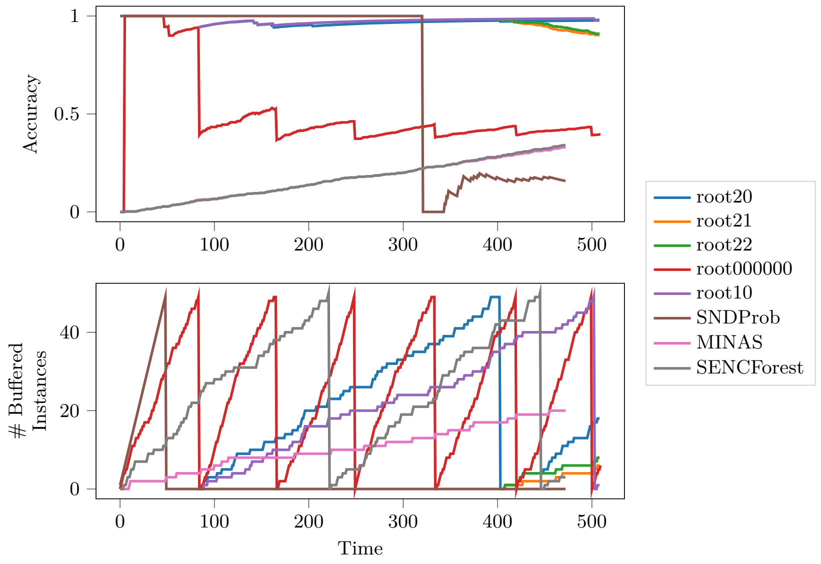


Fig. 4.4: Results (accuracy, number of buffered instances) on the CBF data. Multiple lines correspond to the proposed approach under the *parallel universe* setting. Compared SNDProb, MINAS and SENCForest algorithms are also shown.

those instances are never predicted by this algorithm but only used to update the model.

## 4.6 Conclusions

We propose a novel time series Streaming Novelty Detection (SND) approach based on autoencoder and Deep Support Vector Data Description (Deep SVDD) networks. The proposed solution is capable of accurately predicting newcomer time series and update itself in an unsupervised manner to consider new emerging classes throughout a stream. A novel *parallel universe* framework is designed allowing to use SND approaches in real-world scenarios that an expert could evaluate in hindsight. As a result, we offer a dynamic mechanism to recover from wrong decisions when discovering new emerging classes.

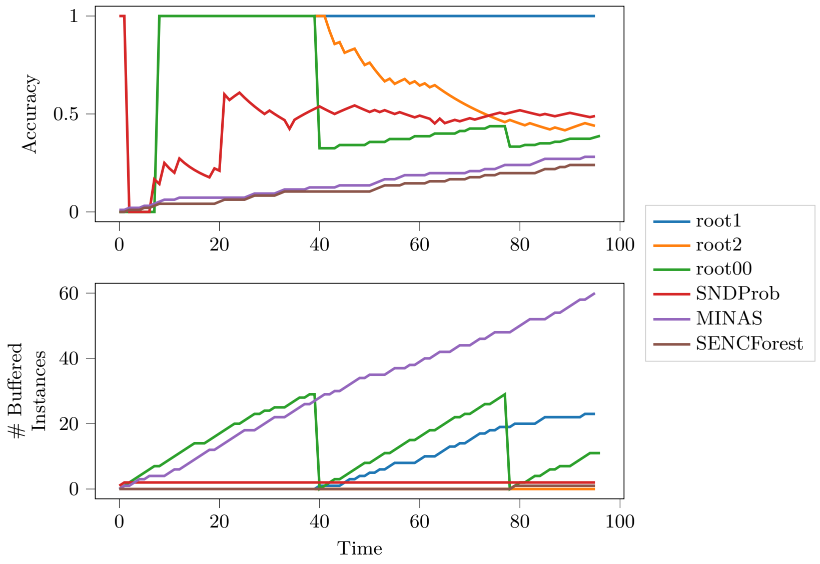


Fig. 4.5: Results (illustrated in the same way as Fig 4.4) on the BME data.

The comparative study shows that our approach outperforms state-of-the-art algorithms by leveraging from the temporal nature of time series, specially, when an ensemble of the *parallel universe* framework matches the real number of classes in the stream. Furthermore, the ensembles of the *parallel universe* framework provide an explanation about the evolution of concepts.

As a first step towards the extension of this work, could be to consider different length time series. Nevertheless, the current solution based on neural networks needs a fixed number of neurons that correspond to the timestamps of the time series. In order to approach to this problem, the time series could be adapted to a fixed length in a preprocessing step [Tan et al., 2019].

The proposed solution offers a novel methodological framework to be used by an expert in hindsight. However, we realize that the expert is not always available in certain domains. Removing the expert from this workflow is a potential, challenging, future work. Nevertheless, since in

SND there is no true labels throughout the stream, assessing the validity of an ensemble is not straightforward. As a first approach, we think that some features could be extracted from the stream, similar to the ones obtained in Chapter 3 in their metaregression approach. We believe that if a new class is discovered and instances are no longer introduced into the buffer once the model is updated whit the new emerging class, we could say that the model is getting more robust. In any case, it can be seen that without ground truth, the instances could be wrongly being classified and hence, that features would not be informative.

Finally, like virtually all methods, there are some hyper-parameters which might be fine tuned; doing so is beyond our intended scope, but would be an obvious candidate for future work.

# 5

# General Conclusions and Future Work

## 5.1 Conclusions

In this dissertation, one contribution to the area of supervised classification and two to the field of Streaming Novelty Detection (SND) have been proposed. In Chapter 2, rare event, anomaly, novelty and outlier detection terms are analyzed from the supervised classification framework and a one-to-one assignment of terms and problems is proposed. As a result, we have given a short step towards the standardization of the field. In our thorough review of the literature, we discovered that there are some different problems named with one of the aforementioned terms. Besides, the same problem is being referred to indistinctly with different terminology. In this hindering situation, we took some key papers of the literature that also tried to clarify this field in their introduction section and we proposed a one to one assignment of terms to learning scenarios. Concretely, we assign the

- rare event detection term to the (early) time series classification problem;
- anomaly detection to the (highly) unbalanced supervised classification problem;
- static novelty detection to a supervised classification problem where only one class is available for training;
- dynamic novelty detection to a supervised classification problem where the number of classes is unknown;

- and, outlier detection, that we relate to the unsupervised classification problem.

In order to validate the proposed assignment of terms and learning scenarios, we performed some experiments by retrieving papers from Google Scholar, ACM Digital Library and IEEE Xplore search engines. In the first experimental scenario, the most cited papers after the year 2000 are obtained in each of the search engines. In the second scenario, the first search-results after 2014 are considered. In both scenarios, for each paper, two terms are obtained. On the one hand, that term used by the authors to describe the problem, and on the other hand, that which would have been assigned by our taxonomy. As a result, a confusion matrix is built for every scenario and search engine. As a result of this experiment, we clearly exposed the motivating mix-up between terms and problems; we analyzed between which terms the confusion was the highest; and we validated our proposed assignment of terms to learning scenarios.

After performing a wide review of the literature, the SND problem particularly draw our attention and, in Chapter 3, our first contribution to the SND problem is described. SND is the problem where, starting from a fully labeled dataset, a model is learned. Afterwards, instances arrive in a stream fashion for classification. Predicted instances are used by the model to update itself and tackle concept drift. Once in a while, the model has not enough evidence to classify the newcomer instances among the previously learned set of classes. Hence, the model identifies these as novelties and stores them into a fixed-sized buffer for further analysis. When the buffer is full, new classes are sought among the stored instances and the model is updated accordingly. In our approach, we proposed a novel parametric framework based on a mixture of Gaussian distributions, where each mixture component models a class. For discovering new emerging classes, we encountered a challenging scenario where both probability distributions and data were present. To face this situation, we opted to weight the Gaussian distributions that model each of the classes and combine these in a modified version of the Expectation Maximization (EM) algorithm. We noticed that obtaining a proper weight set is crucial for discovering new emerging classes. Furthermore, we discovered that the weights widely vary depending on some characteristics of the stream. In particular, aspects such as the buffer filling speed, the ratio of predicted instances of each class or the separation in the feature space between the mixture components affect the adequate weight set. Moreover, we also considered that the buffered instances may belong to the already known set of classes, but due to concept drift, these were introduced into the buffer. Therefore, we proposed to learn a classifier that, based on a set of features that represent the stream, predicts an adequate weight set. Note that as many weights as mixture components need to be predicted by the classifier. We propose to use a metaregressor that follows a hierarchical classification scheme. Firstly, a random forest classifier discriminates between a concept drift situation, where the buffered instances will belong to the already learned classes; and a emerging new classes scenario, where new classes must be discovered among the buffered instances. In the second step of the metaregressor approach, a $k$-Nearest Neighbors ($k$-NN) classifier is performed that will select the adequate weight set.

We also discovered that the arrival time of the classes notably affects the performance of the SND approaches. For instance, if the instances of the different classes arrive sequentially one class after another, the classification consists on discovering that specific change. Therefore, we proposed 6 different arrival strategies that model the probability of sampling from the generative distributions of each class at every timestamp. Furthermore, for illustration purposes, we also created 6 synthetic scenarios. Our results show that when the assumption of the probabilistic framework of the SNDProb are fulfilled, our approach outperforms the selected literature approaches. Furthermore, we also confirmed that the different arrival strategies affect the performance of the classifiers.

SNDProb accounted for some drawbacks of other literature approaches. However, a major issue was discovered while developing the SNDProb solution. Since SND problem assumes that no labels are obtained throughout the online phase, the validity of the model can not be assessed. As a result, the model could derive into a unreliable solution that is potentially unrecoverable. In Chapter 4, we overcome this limitation of the SND problem and also we give a step forward in the field by providing a solution to deal with time series data that has never been addressed before. In order to treat with time series data, we propose an ensemble of both autoencoders and Deep Support Vector Data Description (Deep SVDD) networks that model each of the classes. Deep SVDD networks are used as a one-class classification model that naturally provides an anomaly score. The prediction is made by assigning the class that minimizes the anomaly score. If for all the classes, an instance is out of their normal region, this instance is stored into a fixed-sized buffer. Similar to the SNDProb approach, both the instance and its predicted label are used to update the model. Particularly, this tuple is used to perform another step into the optimization of the networks that model the predicted class. To consider the major issue of the SND problem, we propose to maintain multiple ensembles, that model a different number of classes. Concretely, whenever the buffer is full, a fixed number of hypotheses (new classes) are sought. We proposed to maintain every hypothesis in what we call a *parallel universe* framework so an expert can evaluate them in hindsight. Thus, the models can be recoverable and the solutions can be assessed afterwards by an expert. Our results show that the proposed solution properly discovers new emerging classes and outperforms the other literature approaches by leveraging from time series data. Furthermore, we clearly show the scenarios where an incorrect number of new emerging classes is selected and the difference in performance with respect to an adequate selection in the number of new emerging classes. This third contribution of this PhD dissertation is a result of the research stay in the École Polytechnique with Prof. Jesse Read.

## 5.2 Future Work

The potential future extensions of this PhD work are briefly discussed in the following paragraphs.

In this dissertation, the Streaming Novelty Detection (SND) problem that combines supervised, unsupervised, streaming classification and novelty detection problems have been studied. Since many different learning scenarios have been reviewed in this dissertation, numerous new directions can be taken. Nevertheless, the briefly researched SND problem has, undoubtedly, substantial potential future work.

I would like to differentiate between extensions to developed approaches within the SND literature, and what I consider new research lines with relation to SND.

*A. Extensions to already developed SND solutions:*

- In SNDProb [Carreño et al., 2022], we built the foundations for parametric solutions based on mixtures of probability distributions.

Our approach was to consider the Gaussian mixture as the key probability distribution. As a first extension, different families of probability distributions could be explored. Furthermore, a mixture of different probabilistic families could be fit to the data to improve the flexibility of the model. Following the same idea of improving the flexibility of the model, each of the classes could be modeled with a mixture.

- In all the SND approaches, the number of instances of the buffer are fixed to consider it full. However, I believe that this parameter can be learned from the stream. In Carreño et al. [2022], we discussed about several features that are extracted from the stream. Such features are directly related to the behavior of the stream and the arrival of the instances. Hence, receiving a large number of instances that the model identifies as novel could derive into a shrink of the buffer to discover emerging new classes. On the opposite, having a very low buffer filling rate, could derive into an enlargement of the buffer size.

B. *Novel research lines related to SND problem:*

- Although several evaluation measures have been developed to evaluate SND approaches [Faria et al., 2016, Masud et al., 2013, Mu et al., 2017], non of them consider the reactiveness of the model to discover new emerging classes. Providing new scores could provide more revealing comparisons.
- In the work submitted to NeurIPS 2022 conference described in Chapter 4, instances are time series. This is the first work in SND literature that deals with this type of data. Following the probabilistic nature of SNDProb [Carreño et al., 2022], I think that a model can be learned with a mixture of Hidden Markov Models (HMMs), where each mixture component represents a class. This particular research line has been studied during these PhD years but without success. The main bottleneck is where clustering is performed to discover a set of classes from an unsupervised buffered set of observations. We have not been able to cluster the data although several works of the literature have deal with this task before [Ghassempour et al., 2014, Oates et al., 1999, Smyth, 1997, Yao et al., 2021, Li and Biswas, 2000].

- SND problem can be seen as an abrupt change in the joint probability distribution of the data $p(\mathbf{x}, c)$ where $p(c)$ changes due to an increase/decrease on the cardinality of $c$. Similarly, a problem where the features evolve throughout the stream has been treated [Beyazit et al., 2019]. Nevertheless, in such approaches it is assumed that after predicting the class of a newcomer instance, its label is received to update the model. By combining both scenarios, a challenging scenario comes across. Firstly, new classes would emerge throughout the stream. Secondly, the update of the model would be in an unsupervised manner; and thirdly, at some point, there would be a change in the feature space where the description of the instances would be different henceforth. This learning scenario would be of great interest in many real world situations, specially in manufacturing chains, where sensors tend to wear out and the replacements are alike but not identical. It is known that one of the interest topics is the predictive maintenance where a faulty machine is detected before it breaks. In this problem, the target would not only be the moment before it breaks but the *reason*; and clearly new reasons can emerge in an unsupervised manner.

## 5.3 Main Achievements

### *A. Journal Papers*

- **Carreño A.**, Inza, I., & Lozano, J.A. (2020). Analyzing rare event, anomaly, novelty and outlier detection terms under the supervised classification framework. Artificial Intelligence Review, 53, 3575-3594. Impact Factor[1]: 8.139. Ranking: Q1 (14/139). Cites: 38.
- **Carreño A.**, Inza, I., Lozano, J.A. (2022). SNDProb: A Probabilistic Approach for Streaming Novelty Detection. IEEE Transactions on Knowledge and Data Engineering. In press. Impact Factor[2]: 9,235. Ranking: D1 (10/164).

### *B. Conference Papers*

- **Carreño, A.**, Read, J., Inza, I., & Lozano, J. A (2022). Deep Time Series Streaming Novelty Detection with Emerging New Classes. Submitted to Neural Information Processing Systems (NeurIPS) 2022. Ranking: A++.
  - As a result of the 3 month research stay at the École Polytechnique with Prof. Jesse Read.
- **Carreño, A.**, Inza, I., & Lozano, J. A (2018). Eventos raros, anomalías y novedades vistas desde el paraguas de la clasificación supervisada. XVIII Conferencia de la Asociación Española para la Inteligencia Artificial 925-930. Granada, Spain.

### *C. Workshop posters*

- **Carreño A.**, Inza, I., & Lozano, J.A. (2018). Discover Emerging New Classes. Balance between Supervised and Non-supervised Classification paradigms. 3rd Bilbao Data Science Workshop. Bilbao, Spain.

### *D. Research Stays*

- 17 January - 22 April 2022. DaSciM Team (Laboratoiré d'Informatique, INRIA). École Polytechnique, Paris, France. Supervisor: Prof. Jesse Read.

[1] Category of Computer Science and Artificial Intelligence in 2020.
[2] Category of Computer Science and Information Systems in 2021.

## *E.* ***Dissemination***

- Intelligent Systems Group regular seminars in UPV/EHU. A talk was given yearly (4 in total) where the current developments and research lines were discussed.
- Dissemination talk within the Pint of Science festival entitled: "Inteligencia Artificial: progresos y amenazas". 22nd May 2019, Bilbao.
- Talk to the members and collaborators of the DaSciM Team entitled: "Parametric Streaming Novelty Detection: Discovering new classes while predicting the existing ones". 11th February 2022, Paris.
- Talk to the members and collaborators of the DaSciM Team entitled: "Deep Time Series Streaming Novelty Detection with Emerging New Classes". 8th July 2022, Paris.

## *F.* ***Software***

- https://github.com/andercarreno/SNDProb: SNDProb: A Probabilistic Approach for Streaming Novelty Detection. Developed in R language.
- https://andercarreno.shinyapps.io/SNDProb: Interactive web for running and visualizing SNDProb: A Probabilistic Approach for Streaming Novelty Detection. Implemented with R-Shiny Apps.
- https://github.com/andercarreno/SND_TimeSeries_SVDD: Time Series Streaming Novelty Detection. Developed in Python.

## *G.* ***Other works carried out during the thesis***

- Ortigosa-Hernández, J., **Carreño, A.**, Inza, I. & Lozano, J.A. (2022). Assessing Imbalanced Classification Problems: A Study on the Performance Scores. Submitted to the special issue on imbalanced classification of the Machine Learning journal. Impact Factor[1]: 5,414. Ranking: Q2 (40/144).
- Part of the organization team, providing support to the proceedings chair, of the 2020 Genetic and Evolutionary Computation Conference (GECCO'2020) held as an online conference during 8-12 July, 2020, due to the COVID-19 crisis.

[1] Category of Computer Science and Artificial Intelligence in 2021.

# References

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Christopher M. Bishop. *Neural networks for pattern recognition.* Clarendon Press, 1995. ISBN 9780198538646. URL https://global.oup.com/academic/product/neural-networks-for-pattern-recognition-9780198538646?cc=ca&lang=en&.

Elaine R Faria, André Carlos Ponce de Leon Ferreira Carvalho, and João Gama. MINAS: multiclass learning algorithm for novelty detection in data streams. *Data Mining and Knowledge Discovery*, 30(3):640–680, may 2016. ISSN 1384-5810. doi: 10.1007/s10618-015-0433-y.

Ander Carreño, Inaki Inza, and Jose A. Lozano. SNDProb: A probabilistic approach for streaming novelty detection. *IEEE Transactions on Knowledge and Data Engineering*, 2022. doi: 10.1109/TKDE.2022.3169229.

Mohammad M. Masud, Qing Chen, Latifur Khan, Charu C. Aggarwal, Jing Gao, Jiawei Han, Ashok Srivastava, and Nikunj C. Oza. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams. *IEEE Transactions on Knowledge and Data Engineering*, 25(7):1484–1497, jul 2013. doi: 10.1109/TKDE.2012.109.

Xin Mu, Kai Ming Ting, and Zhi-Hua Zhou. Classification Under Streaming Emerging New Classes: A Solution Using Completely-

Random Trees. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1605–1618, aug 2017. ISSN 1041-4347. doi: 10.1109/TKDE.2017.2691702.

Tom M Mitchell. *Machine Learning.* McGraw-Hill, 1997. ISBN 0070428077.

Richard O. Duda, Peter E. (Peter Elliot) Hart, and David G. Stork. *Pattern classification.* Wiley, 2001. ISBN 9780471056690. URL https://www.wiley.com/en-us/Pattern+Classification%2C+2nd+Edition-p-9780471056690.

Jesse Read, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Streaming multi-label classification. In *Proceedings of the Second Workshop on Applications of Pattern Analysis*, pages 19–25. JMLR Workshop and Conference Proceedings, 2011.

Jesse Read, Antti Puurula, and Albert Bifet. Multi-label classification with meta-labels. In *2014 IEEE international conference on data mining*, pages 941–946. IEEE, 2014.

Johann Faouzi. Time Series Classification: A review of Algorithms and Implementations. *Machine Learning (Emerging Trends and Applications)*, 2022.

Akrem Sellami and Salvatore Tabbone. Deep neural networks-based relevant latent representation learning for hyperspectral image classification. *Pattern Recognition*, 121:108224, 2022.

Adriano Rivolli, Jesse Read, Carlos Soares, Bernhard Pfahringer, and André CPLF de Carvalho. An empirical analysis of binary transformation strategies and base algorithms for multi-label learning. *Machine Learning*, 109(8):1509–1563, 2020.

Jerónimo Hernández-González, Iñaki Inza, and Jose A. Lozano. Weak supervision and other non-standard classification problems: A taxonomy. *Pattern Recognition Letters*, 69:49–55, 2016. ISSN 01678655. doi: 10.1016/j.patrec.2015.10.008.

Amaia Abanda, Usue Mori, and Jose A Lozano. A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2):378–412, may 2019.

Shima Ghassempour, Federico Girosi, and Anthony Maeder. Clustering multivariate time series using Hidden Markov Models. *International journal of environmental research and public health*, 11(3): 2741–2763, mar 2014. ISSN 1660-4601. doi: 10.3390/ijerph110302741.

URL https://pubmed.ncbi.nlm.nih.gov/24662996https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3968966/.

Tim Oates, Laura Firoiu, and Paul R Cohen. Clustering time series with hidden markov models and dynamic time warping. In *International Joint Conference on Artificial Intelligence*, pages 17–21. Citeseer, 1999.

J Lu, A Liu, F Dong, F Gu, J Gama, and G Zhang. Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019. doi: 10.1109/TKDE.2018.2876857.

Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. Hierarchical Change-Detection Tests. *IEEE Transactions on Neural Networks and Learning Systems*, 28(2):246–258, 2017. doi: 10.1109/TNNLS.2015.2512714.

Y Sun, K Tang, L L Minku, S Wang, and X Yao. Online Ensemble Learning of Data Streams with Gradually Evolved Classes. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1532–1545, jun 2016. ISSN 1558-2191. doi: 10.1109/TKDE.2016.2526675.

Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495, 2017.

Albert Bifet, Ricard Gavaldà, Geoff Holmes, and Bernhard Pfahringer. *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press, 2018. https://moa.cms.waikato.ac.nz/book/.

João Gama, Indre Žliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.*, 46(4):44:1—-44:37, mar 2014. ISSN 0360-0300. doi: 10.1145/2523813.

Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we Need Hundreds of Classifiers to Solve Real World

Classification Problems? *Journal of Machine Learning Research*, 15 (90):3133–3181, 2014. URL http://jmlr.org/papers/v15/delgado14a.html.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, sep 1995. ISSN 0885-6125. doi: 10.1007/BF00994018. URL http://link.springer.com/10.1007/BF00994018.

D Tax. *One-class classification.* PhD thesis, Delft University of Technology, 2001.

Pedro Moreno, Purdy Ho, and Nuno Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. *Advances in neural information processing systems*, 16, 2003.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2017.10.013.

Y LeCun, B Boser, J S Denker, D Henderson, R E Howard, W Hubbard, and L D Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541. URL https://doi.org/10.1162/neco.1989.1.4.541.

Ander Carreño, Iñaki Inza, and Jose A. Lozano. Analyzing rare event, anomaly, novelty and outlier detection terms under the supervised classification framework. *Artificial Intelligence Review*, 53(5):3575–3594, may 2020. ISSN 15737462. doi: 10.1007/s10462-019-09771-y.

Edward W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21(3):761–777, 1965. ISSN 0006341X, 15410420. URL http://www.jstor.org/stable/2528559.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231. AAAI Press, 1996.

Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *Acm sigkdd explorations newsletter*, 6(1):90–105, 2004.

Charu C Aggarwal and Philip S Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 70–81, 2000.

Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

Guzman Santafe, Iñaki Inza, and Jose A Lozano. Dealing with the evaluation of supervised classification algorithms. *Artificial Intelligence Review*, 44(4):467–508, 2015. ISSN 1573-7462. doi: 10.1007/s10462-015-9433-y. URL https://doi.org/10.1007/s10462-015-9433-y.

M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Technometrics*, 16(1):125–127, 1974. doi: 10.2307%2F1267500.

Jonathan Ortigosa-Hernández, Iñaki Inza, and Jose A. Lozano. Towards competitive classifiers for unbalanc classification problems: a study on the performance scores. 2016.

Rita P. Ribeiro, Pedro Pereira, and João Gama. Sequential anomalies: a study in the Railway Industry. *Machine Learning*, 105(1):127–153, oct 2016. doi: 10.1007/s10994-016-5584-6.

Marco A F Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99: 215–249, 2014. doi: 10.1016/j.sigpro.2013.12.026.

Stijn Luca, David A Clifton, and Bart Vanrumste. One-class classification of point patterns of extremes. *Journal of Machine Learning Research*, 17:1–21, 2016.

Clifton Phua, Vincent Lee, Kate Smith, and Ross Gayler. A Comprehensive Survey of Data Mining-based Fraud Detection Research. *Monash University*, 2010. doi: 10.1016/j.chb.2012.01.002.

Dit-Yan Yeung and Yuxin Ding. Host-Based Intrusion Detection Using Dynamic and Static Behavioral Models. *Pattern Recognition*, 36(1):229–243, 2001. doi: https://doi.org/10.1016/S0031-3203(02)00026-2. URL http://repository.ust.hk/ir/bitstream/1783.1-2495/1/yeung.pr2003.pdf.

Athanasios Theofilatos, George Yannis, Pantelis Kopelias, and Fanis Papadimitriou. Predicting Road Accidents: A Rare-events Modeling Approach. *Transportation Research Procedia*, 14:3399–3405, 2016. doi: 10.1016/j.trpro.2016.05.293.

Yvonne Dzierma and Heidi Wehrmann. Eruption time series statistically examined: Probabilities of future eruptions at Villarrica and Llaima Volcanoes, Southern Volcanic Zone, Chile. *Journal of Volcanology and Geothermal Research*, 193(1-2):82–92, 2010. doi: 10.1016/j.jvolgeores.2010.03.009.

Hildur Einarsdóttir, Monica Jane Emerson, Line Harder Clemmensen, Kai Scherer, Konstantin Willer, Martin Bech, Rasmus Larsen, Bjarne Kjær Ersbøll, and Franz Pfeiffer. Novelty detection of foreign objects in food using multi-modal X-ray imaging. *Food Control*, 67:39–47, sep 2016. doi: 10.1016/J.FOODCONT.2016.02.023.

Shuoshuo Fan, Guohua Liu, and Zhao Chen. Anomaly detection methods for bankruptcy prediction. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 1456–1460, 2017. ISBN 978-1-5386-1107-4. doi: 10.1109/ICSAI.2017.8248515.

Alex Kafkas and Daniela Montaldi. How do memory systems detect and respond to novelty? *Neuroscience Letters*, feb 2018. doi: 10.1016/J.NEULET.2018.01.053.

M. Van Den Eeckhaut, T. Vanwalleghem, J. Poesen, G. Govers, G. Verstraeten, and L. Vandekerckhove. Prediction of landslide susceptibility using rare events logistic regression: A case-study in the Flemish Ardennes (Belgium). *Geomorphology*, 76(3-4):392–410, 2006. ISSN 0169555X. doi: 10.1016/j.geomorph.2005.12.003.

F. Dufrenois and J. C. Noyer. One class proximal support vector machines. *Pattern Recognition*, 52:96–112, 2016. ISSN 00313203. doi: 10.1016/j.patcog.2015.09.036. URL http://dx.doi.org/10.1016/j.patcog.2015.09.036.

Gary M Weiss and Haym Hirsh. Learning to Predict Rare Events in Event Sequences. *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 359–363, 1998. doi: 10.1.1.30.8264.

Guilherme O. Campos, Arthur Zimek, Jörg Sander, Ricardo J.G.B. Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E. Houle. On the Evaluation of Outlier Detection and One-Class Classification Methods. *Data Science and Advanced Analytics*

*(DSAA), 2016 IEEE International Conference on*, pages 1–10, 2016. ISSN 1573756X. doi: 10.1109/DSAA.2016.8.

Nicholas A. Heard, David J. Weston, Kiriaki Platanioti, and David J. Hand. Bayesian anomaly detection methods for social networks. *Annals of Applied Statistics*, 4(2):645–662, 2010. ISSN 19326157. doi: 10.1214/10-AOAS329.

James D. (James Douglas) (James Douglas) Hamilton. *Time series analysis*. Princeton University Press, 1994. ISBN 9780691042893.

Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys*, 45(1):1–34, nov 2012. doi: 10.1145/2379776.2379788.

Joseph F Murray, Gordon F Hughes, and Kenneth Kreutz-Delgado. Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application. *Journal of Machine Learning Research*, 6:783–816, 2005. doi: 10.1.1.84.9557.

Shengdong Zhang, Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott, and Mohak Shah. Deep learning on symbolic representations for large-scale heterogeneous time-series event prediction. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5970–5974. IEEE, mar 2017. ISBN 978-1-5090-4117-6. doi: 10.1109/ICASSP.2017.7953302.

Usue Mori. *Contributions to time series data mining departing from the problem of road travel time modeling*. PhD thesis, University of the Basque Country, 2015.

Alberto Ogbechie, Javier Díaz-Rozo, Pedro Larrañaga, and Concha Bielza. Dynamic Bayesian Network-Based Anomaly Detection for In-Process Visual Inspection of Laser Surface Heat Treatment. *Machine Learning for Cyber Physical Systems*, pages 17–24, 2017. doi: 10.1007/978-3-662-53806-7_3.

Suzan Köknar-Tezel and Longin Jan Latecki. Improving SVM classification on imbalanced time series data sets with ghost points. *Knowledge and Information Systems*, 28(1):1–23, jul 2011. doi: 10.1007/s10115-010-0310-3.

Hong Cao, Xiao-Li Li, Yew-Kwong Woon, and See-Kiong Ng. SPO: Structure Preserving Oversampling for Imbalanced Time Series Classification. In *2011 IEEE 11th International Conference on Data Mining*, pages 1008–1013. IEEE, dec 2011. ISBN 978-1-4577-2075-8. doi: 10.1109/ICDM.2011.137.

U Mori, A Mendiburu, S Dasgupta, and J A Lozano. Early Classification of Time Series by Simultaneously Optimizing the Accuracy and Earliness. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4569–4578, oct 2018. ISSN 2162-237X. doi: 10.1109/TNNLS.2017.2764939.

Jingxin Xu, Simon Denman, Clinton Fookes, and Sridha Sridharan. Detecting rare events using Kullback-Leibler divergence: A weakly supervised approach. *Expert Systems with Applications*, 54:13–28, 2016. doi: 10.1016/j.eswa.2016.01.035.

Yilong Ren, Yunpeng Wang, Xinkai Wu, Guizhen Yu, and Chuan Ding. Influential factors of red-light running at signalized intersection and prediction using a rare events logistic regression model. *Accident Analysis and Prevention*, 95:266–273, 2016. doi: 10.1016/j.aap.2016.07.017.

Gary King, GKingHarvardEdu Langche Zeng, James Fowler, Ethan Katz, Mike For research assistance, Jim Alt, John Freeman, Kristian Gleditsch, Guido Imbens, Chuck Manski, Peter McCullagh, Walter Mebane, Jonathan Nagler, Bruce Russett, Ken Scheve, Phil Schrodt, Martin Tanner, Richard For helpful suggestions, Scott Bennett, Paul Huth, Richard Tucker, Mike Tomz for research assistance, Jim Alt, John Freeman, Kristian Gleditsch, Guido Imbens, Chuck Manski, Peter McCullagh, Walter Mebane, Jonathan Nagler, Bruce Russett, Ken Scheve, Phil Schrodt, Martin Tanner, Richard Tucker for helpful suggestions, Scott Bennett, Paul Huth, and Richard Tucker. Logistic Regression in Rare Events Data. *Political Analysis*, 9(2):137–163, 2001.

J.-M. Bourinet. Rare-event probability estimation with adaptive support vector regression surrogates. *Reliability Engineering and System Safety*, 150:210–221, 2016. doi: 10.1016/j.ress.2016.01.023.

Seong-Pyo Cheon, Sungshin Kim, So-Young Lee, and Chong-Bum Lee. Bayesian networks based rare event prediction with sensor data. *Knowledge-Based Systems*, 22(5):336–343, 2009. doi: 10.1016/j.knosys.2009.02.004.

Wael Khreich, Babak Khosravifar, Abdelwahab Hamou-Lhadj, and Chamseddine Talhi. An anomaly detection system based on variable N-gram features and one-class SVM. *Information and Software Technology*, 91:186–197, 2017. doi: 10.1016/j.infsof.2017.07.009.

Jianxin Wu, James M Rehg, and Matthew D Mullin. Learning a Rare Event Detection Cascade by Direct Feature Selection. *Neural Information Processing Systems (NIPS)*, 16:1–17, 2003.

Francesco Cadini, Gian Luca Agliardi, and Enrico Zio. Estimation of rare event probabilities in power transmission networks subject to cascading failures. *Reliability Engineering and System Safety*, 2017. ISSN 09518320. doi: 10.1016/j.ress.2016.09.009.

Suraje Dessai and Mike Hulme. Does climate adaptation policy need probabilities? *Climate Policy*, 4(2):107–128, jan 2004. doi: 10.1080/14693062.2004.9685515.

Leonardo Dueñas-Osorio and Srivishnu Mohan Vemuru. Cascading failures in complex infrastructure systems. *Structural Safety*, 31(2):157–167, mar 2009. doi: 10.1016/J.STRUSAFE.2008.06.007.

T. Bedford and Roger M. Cooke. *Probabilistic risk analysis : foundations and methods*. Cambridge University Press, 2001. ISBN 0521773202.

Mathieu Balesdent, Jérôme Morio, and Loïc Brevault. Rare Event Probability Estimation in the Presence of Epistemic Uncertainty on Input Probability Distribution Parameters. *Methodology and Computing in Applied Probability*, 18(1):197–216, 2016. doi: 10.1007/s11009-014-9411-x.

Yves Auffray, Pierre Barbillon, and Jean Michel Marin. Bounding rare event probabilities in computer experiments. *Computational Statistics and Data Analysis*, 80:153–166, 2014. doi: 10.1016/j.csda.2014.06.023.

Daniel Straub, Iason Papaioannou, and Wolfgang Betz. Bayesian analysis of rare events. *Journal of Computational Physics*, 314:538–556, 2016. doi: 10.1016/j.jcp.2016.03.018.

S. Miri Rostami and M. Ahmadzadeh. Extracting Predictor Variables to Construct Breast Cancer Survivability Model with Class Imbalance Problem. *Shahrood University of Technology*, 6(2):263–276, jul 2018. doi: 10.22044/JADM.2017.5061.1609.

Ugo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, 2017. ISSN 00200255. doi: 10.1016/j.ins.2017.12.030.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection. *ACM Computing Surveys*, 41(3):1–58, jul 2009. ISSN 03600300. doi: 10.1145/1541880.1541882. URL http://portal.acm.org/citation.cfm?doid=1541880.1541882.

Ying Zhou, Wanjun Su, Lieyun Ding, Hanbin Luo, and P.E.D. Love. Predicting Safety Risks in Deep Foundation Pits in Subway Infrastructure Projects: Support Vector Machine Approach. *Journal of Computing in Civil Engineering*, 31(5):04017052, 2017. doi: 10.1061/(ASCE)CP.1943-5487.0000700.

Keith Noto, Carla Brodley, and Donna Slonim. FRaC: a feature-modeling approach for semi-supervised and unsupervised anomaly detection. *Data mining and knowledge discovery*, 25(1):109–133, 2012. doi: 10.1007/s10618-011-0234-x.

Douglas Reynolds. Gaussian Mixture Models. In *Encyclopedia of Biometrics*, pages 827–832. Springer US, Boston, MA, 2015. doi: 10.1007/978-1-4899-7488-4_196.

Matheus Araujo, Rahul Bhojwani, Jaideep Srivastava, Louis Kazaglis, and Conrad Iber. ML Approach for Early Detection of Sleep Apnea Treatment Abandonment. *Proceedings of the 2018 International Conference on Digital Health - DH '18*, pages 75–79, 2018. doi: 10.1145/3194658.3194681. URL http://dl.acm.org/citation.cfm?doid=3194658.3194681.

Mayank Swarnkar and Neminath Hubballi. OCPAD: One class Naive Bayes classifier for payload based anomaly detection. *Expert Systems with Applications*, 64:330–339, 2016. doi: 10.1016/j.eswa.2016.07.036.

Xin Mu, Feida Zhu, Yue Liu, Ee-Peng Lim, and Zhi-Hua Zhou. Social Stream Classification with Emerging New Labels. In *PAKDD*, pages 16–28. Springer, jun 2018. doi: 10.1007/978-3-319-93034-3_2.

Eduardo J. Spinosa, André Ponce De Leon F. De Carvalho, and João Gama. OLINDDA: a cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 448 – 452, 2007. ISBN 1-59593-480-4. doi: 10.1145/1244002.1244107.

Yue Zhu, Kai Ming Ting, and Zhi Hua Zhou. Multi-label learning with emerging new labels. *IEEE Transactions on Knowledge and Data Engineering*, 2018. ISSN 1041-4347. doi: 10.1109/TKDE.2018.2810872.

Shixi Chen, Haixun Wang, Shuigeng Zhou, and Philip S. Yu. Stop chasing trends: Discovering high order models in evolving data. In *Proceedings - International Conference on Data Engineering*, pages 923–932, 2008. ISBN 9781424418374. doi: 10.1109/ICDE.2008.4497501.

Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, 58:121–134, 2016. doi: 10.1016/j.patcog.2016.03.028.

Vasileios Mygdalis, Alexandros Iosifidis, Anastasios Tefas, and Ioannis Pitas. Graph Embedded One-Class Classifiers for media data classification. *Pattern Recognition*, 60:585–595, dec 2016. doi: 10.1016/J.PATCOG.2016.05.033.

Chesner Désir, Simon Bernard, Caroline Petitjean, and Laurent Heutte. One class random forests. *Pattern Recognition*, 46(12):3490–3506, dec 2013. doi: 10.1016/J.PATCOG.2013.05.022.

Daqing Zhang, Nan Li, Zhi-Hua Zhou, Chao Chen, Lin Sun, and Shijian Li. iBAT: Detecting anomalous taxi trajectories from GPS traces. *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp '11)*, pages 99–108, 2011. doi: 10.1145/2030112.2030127.

V Hodge and J Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004. URL https://link.springer.com/article/10.1023/B:AIRE.0000045502.10941.a9.

J Zhang and M Zulkernine. Anomaly based network intrusion detection with unsupervised outlier detection. In *IEEE International Conference on Communications*, pages 2388–2393. ieeexplore.ieee.org, 2006. doi: 10.1109/ICC.2006.255127. URL https://ieeexplore.ieee.org/abstract/document/4024522/.

N Billor, A S Hadi, and P F Velleman. BACON: blocked adaptive computationally efficient outlier nominators. *Computational Statistics & Data Analysis*, 34(3):279–298, 2000. URL https://www.sciencedirect.com/science/article/pii/S0167947399001012.

Charu C. Aggarwal. *Outlier Analysis*. Springer Cham, 2017. ISBN 978-3-319-47577-6. doi: 10.1007/978-3-319-47578-3.

H.S. Teng, K. Chen, and S.C. Lu. Adaptive real-time anomaly detection using inductively generated sequential patterns. In *Proceedings. 1990 IEEE Computer Society Symposium on Research in Security* *and Privacy*, pages 278–284. IEEE, 1990. ISBN 0-8186-2060-9. doi: 10.1109/RISP.1990.63857.

Peter J. Rousseeuw, Annick M. Leroy, John Wiley & Sons., Wiley InterScience (Online service), and Mia Hubert. Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):73–79, jan 2011. doi: 10.1002/widm.2.

Anwesha Barai and Dey Lopamudra. Outlier Detection and Removal Algorithm in K-Means and Hierarchical Clustering. *World Journal of Computer Application and Technology*, 5(2):24–29, 2017. doi: 10.13189/WJCAT.2017.050202.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Reverse Nearest Neighbors in Unsupervised Distance-Based Outlier Detection. *IEEE Transactions on Knowledge and Data Engineering*, 4347(OCTOBER):1–14, 2014. ISSN 10414347. doi: 10.1109/TKDE.2014.2365790.

Xuan Hong Dang, Ira Assent, Raymond T. Ng, Arthur Zimek, Erich Schubert, Xuan Hong Dang, Ira Assent, Raymond T. Ng, Arthur Zimek, and Erich Schubert. Discriminative features for identifying and interpreting outliers. *Proceedings - International Conference on Data Engineering*, pages 88–99, mar 2014. ISSN 10844627. doi: 10.1109/ICDE.2014.6816642. URL http://ieeexplore.ieee.org/document/6816642/.

Manish Gupta, Jing Gao, and Charu C Aggarwal. Outlier Detection for Temporal Data : A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):1–20, 2013. ISSN 1041-4347. doi: http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.184.

Mohammad M Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. Integrating Novel Class Detection with Classification for Concept-Drifting Data Streams. In Wray Buntine, Marko Grobelnik, Dunja Mladenić, and John Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 79–94, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04174-7.

Ahsanul Haque, Latifur Khan, and Michael Baron. SAND: Semi-Supervised Adaptive Novel Class Detection and Classification over Data Stream. In *AAAI Conference on Artificial Intelligence*, pages 1652 – 1658, 2016a.

Ahsanul Haque, Latifur Khan, Michael Baron, Bhavani Thuraisingham, and Charu Aggarwal. Efficient handling of concept drift and concept evolution over Stream Data. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 481–492, 2016b. doi: 10.1109/ICDE.2016.7498264.

Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and Joao Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, 21(2):6–22, 2019.

Shehroz S Khan and Michael G Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014. doi: 10.1017/S026988891300043X.

Zahraa S. Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. AnyNovel: detection of novel concepts in evolving data streams. *Evolving Systems*, 7(2):73–93, jun 2016. ISSN 1868-6478. doi: 10.1007/s12530-016-9147-7. URL http://link.springer.com/10.1007/s12530-016-9147-7.

Tiago Pinho da Silva, Leonardo Schick, Priscilla de Abreu Lopes, and Heloisa de Arruda Camargo. A Fuzzy Multiclass Novelty Detector for Data Streams. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, 2018. doi: 10.1109/FUZZ-IEEE.2018.8491545.

Tiago Pinho da Silva and Heloisa de Arruda Camargo. Possibilistic Approach For Novelty Detection In Data Streams. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, 2020. doi: 10.1109/FUZZ48607.2020.9177582.

Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. Fast Anomaly Detection for Streaming Data. In *International Joint Conference on Artificial Intelligence*, pages 1511–1516, 2011.

Yuh-Jye Lee, Yi-Ren Yeh, and Yu-Chiang Frank Wang. Anomaly Detection via Online Oversampling Principal Component Analysis. *IEEE Transactions on Knowledge and Data Engineering*, 25(7):1460–1470, 2013. doi: 10.1109/TKDE.2012.99.

Jeremiah D Deng. Online Outlier Detection of Energy Data Streams Using Incremental and Kernel PCA Algorithms. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 390–397, 2016. doi: 10.1109/ICDMW.2016.0062.

Kemilly Dearo Garcia, Elaine Ribeiro de Faria, Cláudio Rebelo de Sá, João Mendes-Moreira, Charu C Aggarwal, André C P L F de Carvalho, and Joost N Kok. Ensemble Clustering for Novelty Detection in Data Streams. In Petra Kralj Novak, Tomislav Šmuc, and Sašo Džeroski, editors, *Discovery Science*, pages 460–470, Cham, 2019. Springer International Publishing. ISBN 978-3-030-33778-0.

Don Dennis, Durmus Alp Emre Acar, Vikram Mandikal, Vinu Sankar Sadasivan, Venkatesh Saligrama, Harsha Vardhan Simhadri, and Prateek Jain. Shallow RNN: Accurate Time-series Classification on Resource Constrained Devices. In H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Yue Bai, Lichen Wang, Zhiqiang Tao, Sheng Li, and Yun Fu. Correlative Channel-Aware Fusion for Multi-View Time Series Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6714–6722, 2021.

Zipeng Chen, Qianli Ma, and Zhenxi Lin. Time-Aware Multi-Scale RNNs for Time Series Modeling. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2285–2291, 2021.

Ashok Cutkosky. Parameter-free, Dynamic, and Strongly-Adaptive Online Learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2250–2259. PMLR, 2020.

Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-Lin Liu. Class-Incremental Learning via Dual Augmentation. In *Advances in Neural Information Processing Systems*, 2021.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

Maciej Grzenda, Heitor Murilo Gomes, and Albert Bifet. Delayed labelling evaluation for data streams. *Data Mining and Knowledge Discovery*, 34(5):1237–1266, 2020. ISSN 1573-756X. doi: 10.1007/s10618-019-00654-y.

Guozhong Li, Byron Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace L H Wong. Shapenet: A shapelet-neural network approach for multivariate time series classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8375–8383, 2021.

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep One-Class Classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 2018.

Luca Martino, Jesse Read, Víctor Elvira, and Francisco Louzada. Cooperative parallel particle filters for online model selection and applications to urban mobility. *Digital Signal Processing*, 60:172–185, 2017. ISSN 1051-2004. doi: https://doi.org/10.1016/j.dsp.2016.09.011.

Jeremy Bernstein, Arash Vahdat, Yisong Yue, and Ming Yu Liu. On the distance between two neural networks and the stability of learning. In *Advances in Neural Information Processing Systems*, 2020.

Davide Chicco. *Siamese Neural Networks: An Overview*, pages 73–94. Springer US, 2021. ISBN 978-1-0716-0826-5. doi: 10.1007/978-1-0716-0826-5_3.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The UCR Time Series Archive, 2018.

Chang Wei Tan, Francois Petitjean, Eamonn Keogh, and Geoffrey I Webb. Time series classification for varying length series. *arXiv preprint arXiv:1910.04341*, 2019.

Padhraic Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems*, 1997.

Ying Yao, Xiaohua Zhao, Yiping Wu, Yunlong Zhang, and Jian Rong. Clustering driver behavior using dynamic time warping and hidden Markov model. *Journal of Intelligent Transportation Systems*, 25 (3):249–262, 2021. doi: 10.1080/15472450.2019.1646132. URL https://doi.org/10.1080/15472450.2019.1646132.

C. Li and G. Biswas. Improving clustering with hidden markov models using bayesian model selection. In *2000 IEEE International Conference on Systems, Man and Cybernetics.*, pages 194–199, 2000. doi: 10.1109/ICSMC.2000.884988.