University of Padua

Department of Mathematics

Master Degree in Data Science

MASTER THESIS

Adaptive Learning in Low-Power Wide-Area Networks

Supervisor: Prof. MICHELE ROSSI
Department of Information Engineering
University of Padua

Co-supervisor: Prof. XAVIER VILAJOSANA
Faculty of Computer Science
Open University of Catalonia

Student: FEDERICO MATTEO
Student N° 1206030

Abstract

Internet of Things applications are driving the need for better and advanced solutions to connect and manage sensors communicating into a smarter world. In this thesis we present preliminary results derived from tests on dual-band Low Power Wide Area Networks, trying to solve both effectiveness and efficiency challenges from a data science point of view. In the former case, by applying machine learning models with high performances while keeping the load of sensor networks as low as possible in terms of acknowledgements requested by single nodes, in the latter by storing a simple model in a small device while providing fast and accurate predictions. Statistical inference techniques and online learning algorithms are investigated under non-stationary conditions and adopted for testing multiple practical scenarios, with the aim of estimating the network status in the licensed and unlicensed bands, i.e. NB-IoT and LoRaWAN respectively.

Contents

Chapter 1

Introduction

The rise of new technological trends in the Internet of Things (IoT) industry is promising to revolutionize our user-experience by providing us with a set of incredible smart devices, which will assist us everywhere and at any time. Several research communities have actively studied how to manage and develop the IoT devices, including the field of network technologies that are having an important role in the success of IoT since they are responsible for the connection between the objects.

From a computational point of view, specific wireless sensor networks (WSNs) have to handle a large number of potential devices. In practice, multiple sensors must satisfy precise requirements, for example only small chunks of data can be intermittently transmitted, without power supply, implying to operate by adopting battery with reasonable lifetimes while often being installed in places with poor coverage.

From a data science point of view, the characteristics of ubiquity and pervasiveness of those technologies sets industry practitioners and academic researchers both effectiveness and efficiency challenges. On one hand, it is required to design machine learning models with high predictive performances (effectiveness) on a given set of tasks. On the other hand, these models are required to be stored on a device (space-efficiency) and produce fast predictions (time-efficiency). In addition, the prevalence of machine-machine communications and networks of sensors has led to an enormous and ever increasing amount of data that are now more commonly

available in a streaming fashion. Traditional machine learning and statistical tools have been developed to deal with batches of data and only in recent studies have been adapted to cope with data streams or big data problems, addressing issues in data availability and resource scarcity respectively.

The sequential data acquisition process has lead to establishing the online learning paradigm, a framework that can be naturally adapted to the IoT context, since online learning algorithms are fast and often make few statistical assumptions, making them suitable for the majority of IoT applications. Online learning is based on the acquisition of a representative training data, which is expensive and time consuming. In such settings, it is necessary to update an existing model in an incremental fashion to accommodate new data without compromising the performance on old data, i.e. learning new information without forgetting previously acquired knowledge. This idea is commonly based on the assumption that the probability function generating the data is not changing over time, i.e. the process under study is stationary.

The stationarity has lead to the construction of mathematical models that have good theoretical properties, but it is hardly verified in practical applications. Phenomena can evolve or merely change during time intervals and if machine learning models do not consider the evolving and varying data flow - also called concept drift in the literature - they can become obsolete and perform poorly after the alteration. In the IoT context, the non-stationarity can be caused, for example, by seasonality or periodicity effects, variations in the external environment, changes in the sensor locations or aging effects in the sensors performances. Machine Learning in non-stationary environments is a topic acquiring increasing attention in recent years; in this work we investigate how they practically behaves and how they are trying to encompass more effective and efficient algorithms.

The objective of this thesis is studying how to include adaptive learning and concept drift detection techniques in the Long Range Wide Area Network (LoRaWAN) technology, a data-link layer with long range, low power, and low bit rate, appeared as a promising solution for IoT in which end-devices use LoRa to communicate with gateways through a single hop. The algorithms developed in this work are designed

to be appropriate for real life applications where training data becomes gradually available over time or their size is out of system memory capability. Therefore, the tools presented require the sensors to dynamically employ LoRaWAN in order to improve the packet delivery ratio of the whole network. In the end, it is shown that the designed procedure is effective also in practice, when no information on the external environment, noise and sensor' locations is available. The structure of the rest of this thesis is as follows.
In Chapter 2 we present the technologies studied and a brief summary of the learning in non-stationary conditions. In Chapter 3 all the mathematical instruments required to tackle the problem of the thesis are presented. In Chapter 4 the problem and its solution are formally proposed. Finally, in Chapter 5 we present the results of the analysis and in Chapter 6 the conclusion.

Chapter 2

Background

This Chapter is intended to present what is the LoRaWAN through an high level description, which includes the motivation beside the rise and the diffusion of this technology, its main technical and practical characteristics as well as some use-cases examples and limitations. The background concerning learning in non-stationary conditions literature is also reported via a brief survey of actual methods.

2.1 Wireless Sensor Networks

Issues with resources like water and energy, as well increasing awareness of safety security and efficiency are driving the need for more information and better options to connect and manage advanced sensors to build a smarter world. In a few years, global, national and regional networks will connect trillions of wireless devices. Hundreds of objects will be connected and send and receive hundreds of messages every day even at home. We will be able to monitor and control all the devices connected to the Internet. The Internet of Things is going to connect the world and create lots of efficiencies and new business opportunities.

2.1.1 IoT Challenges

The rising of new scenarios for Internet of Things introduces new challenges that cannot be addressed by already available connectivity protocols, since the diffusion and development of IoT applications has increased the performance demand required by nodes in WSNs.

One of the first aspect to take into account in order to enhance network performances is finding the optimal transmission power for each network node, i.e. increasing the *lifetime* and improving energy efficiency by minimizing wasted power of single nodes [1]. Due to economical reasons, most sensors are powered by non-rechargeable batteries. However, the need for replacement and ecological implications will become a severe problem when powering billions of IoT devices employing non-rechargeable batteries as the primary energy storage. Nevertheless, keeping communication effective while reducing energy consumption has been an open research point [3] and could be the first point to extend the work of this thesis. Other aspects such as *throughput*, *range coverage* and *bandwidth* are also important features to support the massive number of expected nodes connecting to the Internet.

2.1.2 Overview of LoRaWAN

To avoid some of these problems and to solve some of them, LoRa (Long Range), a low-power wide-area network protocol developed by Semtech, identifies end-devices that have limited energy and transmit few bytes every time [2]. Both a sensor (end-device) and an external entity (gateway) can initiate a communication, i.e. a data transfer. LoRa is an excellent candidate for being used in many applications in smart environments given its long range and low power features.

LoRa exploits a physical layer based on spread spectrum modulation techniques derived from chirp spread spectrum (CSS) technology, and a MAC layer protocol yielding access to the LoRa architecture itself: the LoRaWAN. To maximize the *lifetime* of the final device batteries, the LoRaWAN server controls the Radio frequency output and an output rate through an adaptive scheme for each end-device, achieving an average battery life time longer than any other technology.

LoRa provides a *throughput*, i.e. the rate of successful message delivery over a communication channel, greater than technologies based in ALOHA, with low complexity in the Medium Access Control. Throughput is usually measured in bits per second and LoRa offers a data rate between 290 bps and 50 kbps.

New technologies have the objective to provide information access to the Internet to people/things away from the big metropolis. To achieve greater distances, the power of the radio should be increased, causing greater consumption of the battery. Hence, new protocols aim to obtain greater distances while maintaining lower energy consumption. Currently, LoRaWAN obtains multi-kilometers communication, about 2-5 km of *range coverage* in urban perimeters and about 10-15 km in rural areas. *Bandwidth* and data rate are used to determine the amount of data being transferred (bit rate) in a given time unit, normally, seconds. Bandwidth means the spectrum range in Hertz that a system can use for digital communication. Data rate depends on the bandwidth of the Internet connection. If the bandwidth is high, the bit rate tends to be high whether adequate digital communication technologies are employed. In LoRaWAN, the data rate is selected by a trade-off between the communication range and the duration of the messages, thus lower bit rate are considered the best solution for IoT applications.

2.1.3 LoRaWAN

LoRaWAN employs LoRa as physical layer. It features low-power operation, guaranteeing around 10 years of battery lifetime, low data rate, such as 27 kb/s with spreading factor 7 and long communication range, i.e. 2-5 km in urban areas and 15 km in suburban areas.

LoRaWAN networks are organized in a star-of-stars topology, in which gateway nodes relay messages between end-devices and a central network server. End devices send data to gateways over a single wireless hop, and gateways are connected to the network server through a non-LoRaWAN network (e.g., IP over cellular or Ethernet) [8]. Communication is bidirectional, although uplink communication from end-devices to the network server is strongly favored, as explained in the following [9] LoRaWAN defines three types of devices (Classes A, B, and C) with different ca-

pabilities. Class A devices use pure ALOHA access for the uplink. After sending a frame, a Class A device listens for a response during two downlink receive windows. Each receive window is defined by the duration, an offset time, and a data rate. Although offset time can be configured, the recommended value for each receive window is 1 s and 2 s, respectively. Downlink transmission is only allowed after a successful uplink transmission. The data rate used in the first downlink window is calculated as a function of the uplink data rate and the receive window offset. In the second window the data rate is fixed to the minimum, 0.3 kb/s. Therefore, downlink traffic cannot be transmitted until a successful uplink transmission is decoded by the gateway. The second receive window is disabled when downlink traffic is received by the end-device in the first window. Class A is the class of LoRaWAN devices with the lowest power consumption.

Class B devices are designed for applications with additional downlink traffic needs. These devices are synchronized using periodic beacons sent by the gateway to allow the scheduling of additional receive windows for downlink traffic without prior successful uplink transmissions. Obviously, a trade-off between downlink traffic and power consumption arises.

Finally, Class C devices are always listening to the channel except when they are transmitting. Only Class A must be implemented in all end devices, and the other classes must remain compatible with Class A. In turn, Class C devices cannot implement Class B. The three classes can coexist in the same network, and devices can switch from one class to another. However, there is not a specific message defined by LoRaWAN to inform the gateway about the class of a device; this is up to the application.

The underlying PHY of the three classes is the same. Communication between end-devices and gateways start with a Join procedure that can occur on multiple frequency channels by implementing pseudo-random channel hopping.

Each frame is transmitted with a specific spreading factor (SF), defined as SF = $\log_2(Rc/Rs)$, where Rs is the symbol rate and Rc is the chip rate. Accordingly, there is a trade-off between SF and communication range. The higher the SF (i.e., the slower the transmission), the longer the communication range. The codes used in

the different SFs are orthogonal. This means that multiple frames can be exchanged in the network at the same time, as long as each one is sent with one of the six different SFs (from SF = 7 to SF = 12). Depending on the SF in use, LoRaWAN data rate ranges from 0.3 kb/s to 27 kb/s.

The maximum duty cycle, defined as the maximum percentage of time during which an end device can occupy a channel, is a key constraint for networks operating in unlicensed bands. Therefore, the selection of the channel must implement pseudo-random channel hopping at each transmission and be compliant with the maximum duty cycle.

The LoRa physical layer uses chirp spread spectrum (CSS) modulation, a spread spectrum technique where the signal is modulated by chirp pulses (frequency varying sinusoidal pulses), hence improving resilience and robustness against interference, Doppler effect, and multipath. Packets contain a preamble (typically with 8 symbols), a header (mandatory in explicit mode), the payload (with a maximum size between 51 and 222 bytes, depending on the SF), and a cyclic redundancy check (CRC) field (with configurations that provide a coding rate from 4/5 to 4/8). Typical bandwidth (BW) values are 125, 250, and 500 kHz in the HF ISM 868 and 915 MHz band, while they are 7.8, 10.4, 15.6, 20.8, 31.2, 41.7, and 62.5 kHz in the LF 160 and 480 MHz bands. The raw data rate varies according to the SF and the BW, and ranges between 22 b/s (BW = 7.8 kHz and SF=12) to 27kb/s (BW=500kHzandSF = 7) [10]. Frequency hopping is exploited at each transmission in order to mitigate external interference [11].

2.1.4 Examples of LoRaWAN use-cases

We propose three examples of LoRaWAN use-cases. The first is *Loadsensing*, i.e. the global leader product for connecting and wirelessly monitoring infrastructures in remote locations. Construction and mining companies and operators of bridges, tunnels, dams, railways and many other inaccessible assets can exploit this technology to work with reliable data. Having access to this information and real-time insights enables operators to anticipate needs, manage their workforce, diminish risks, and

even prevent disasters. This technology is battery-powered, characterized by a long range and developed by low-power wide-area Networks devices.

Wireless sensors play an important role in today's agriculture. There is a need to optimize production while minimizing the environmental impact. Sensors make this possible, since help monitor nutrients in the soil, humidity, temp, density of weeds and also help reduce the use of chemicals such as fertilizers and herbicides and yield safer corps. Hence, the second example is represented by *Sol-Chip*'s everlasting IoT platform, which uses compact, embedded panels that require no maintenance and allow data transmission via LoRa. It's long-range two-way wireless communications collects sensory data from the field and allows operating valves or other devices remotely. By receiving real-time data, users can make optimal irrigation and treatment decisions in their fields.

Finally, we conclude by citing *Urbiotica*, a company that has reached the leader position of smart parking solutions, strengthening its leadership position in the sector and opening up new opportunities to penetrate new markets thanks to the acquisition of *Fastprk*'s technology, based on dual detection and LoRa communication protocol.

2.1.5 Limitations and open issues

The LoRaWAN protocol has several advantages over other LPWA technologies, namely the data rate ranges from 300 bps up to 5 kbps (with 125 kHz bandwidth) and 11 Kbps (with 250 kHz bandwidth) allowing for better time-on-air and better battery life, furthermore communication is bidirectional and unlimited. Research works comparing several long-range technologies and LoRa are presented in [12]. This latter article provides an overview and functional description of LoRaWAN. Continuing this work, performance analysis studies of the protocol may consider the estimation of the collision rate, total capacity, channel load, single device maximal throughput and maximum transmission unit, scaling networks to a massive number of devices, and mobility/roaming proposing possible solutions for performance enhancement. All research topics that are currently carried out from both academic and industry researchers and experts of internet and sensor technologies.

2.2 State of the Art

2.2.1 Non-stationary environments

In data science modelling and statistical estimation procedures, either implicitly or explicitly, both researchers and practitioners assume that the process generating the distribution of interest is stationary, i.e. the data are drawn from a fixed and unknown probability distribution.

However, in many empirical application and natural phenomena this assumption is not satisfied. Hence, in order to avoid a decrease in the performances of the trained models it is convenient to employ methods able to deal with distributions that can drift over time. This condition is known as learning in non-stationary environments. Generally, it is assumed that the probabilistic properties of the data change over time or that the parameters describing the phenomenon of interest scale according to changes in the external environment. Machine learning models can effectively deal with these kind of scenarios by adopting an adaptive learning algorithm, since models trained under the false stationarity assumption are intended to become obsolete in time, as well as to present sub-optimally performances with respect to an adaptive model.

When coping with non-stationary environments, it is possible to consider two different scenarios affecting the active learning process.

Let $\mathcal{P}$ be the probability distribution generating tuples (x_t, y_t), which are sampled from an unknown probability distribution $p_t(x, y)$, and let $p_t(y|x)$ and $p_t(x)$ be the posterior and the marginal distributions, respectively, at some arbitrary time stamp t. We can distinguish between the real-drift case, in which the posterior distribution varies over time, independently from variations in the marginal one; and the virtual-drift case, where the marginal distribution of the data changes without affecting the posterior probability of classes.

The drift in the probability distribution can be further decomposed with respect to the rate at which the change is accomplished and then be defined as permanent if the effect of the variation is not limited in time, or transient, in case the change is maintained for a certain amount of time, after which the effect of the drift disappears.

Other assumptions can be made based on the nature of the data, for instance the change in the distribution could be recurrent, data on which the model is trained could be available only for a certain amount of time and, most importantly, the process could be subjected to the so called verification latency, in which the labels do not become available immediately as the next batch of data arrives - assumption at the basis of the online learning paradigm -, and in the most complex case of which true labels are never revealed for certain associated instances.

In this work, all the assumptions on the data generating process changing over time are not taken into account, since due to the loss of information on the external processes we do not know neither the rate of the change nor the properties of its nature. Thus, we simply define the concept drift as an external agent influencing the learning procedure, that can be perceived from the model under certain conditions. Rather than actual, in this scenario the drifting distribution is caused by insufficient, unknown and unobserved attributes [4].

Having the capacity of observing the hidden parameters would remove the non-stationarity. Since the hidden context can never be known, the learner must rely on the above mentioned probabilistic definition to describe non-stationary environments. Due to this lack of knowledge, in the next applications we will constantly track the external environment, trying to detect eventual changes in both the probability distribution and the classifier performance. To this aim, the adopted active approach for learning a classifier in non-stationary environments will be based on a feature extraction process extracting features from the data generating process both for change detection and classification. The change detectors inspect features extracted for change detection purposes and/or the classification error evaluated over labeled samples. Once a change has been detected, the adaptation phase is activated to update or rebuild the classifier. The majority of authors working in this field provide solutions based on replacing an already trained classifier with a new one, restarting the learning procedure as soon as a change in the environment is warned or detected. This approach, although useful and efficient in many applications, turns out to be disastrous when applied to our use-case. Indeed, since we are dealing with a problem characterized from both class imbalance and verification

latency problems, the replacement of the learner caused the loss of all the precious information slowly gathered.

For this reason, we preferred to simply update the classifier with respect to the change, and as will be further described later, modify the rate of the sampling from the unknown probability distribution generating the data, in order to accelerate the adaptation phase and to limit the availability of labels if not strictly necessary. This allows us to avoid a decrease in the final performances.

Another important aspect to take into account when dealing with adaptation algorithms for learning in the presence of concept drift is that they are primarily based on either an active or on a passive approach [5, 6]. Algorithms following the active approach try to detect concept drift using techniques suited for this task, while algorithms following the passive one continuously update the model every time new data are presented, regardless whether drift is present or not. Both active and passive approaches expect to provide an up-to-date model. However, the mechanisms used by each to succeed in this purpose are not the same.

Both active and passive approaches can be successful in practice, but the reason for choosing one approach over the other is typically specific to the application. As a matter of fact, before choosing a specific algorithm for learning in a non-stationary environment, it is important to consider the dynamics of the learning scenario, e.g. drift rates and whether the data arrive online or in batches of samples. Other aspects to consider are the computational resources available, such as limited memory sensors or high-performance machines, and any assumption that can be made about the distributions of the data. In general, passive approaches have been shown to be quite effective in prediction settings with both gradual and recurring drifts [4]. Furthermore, passive approaches generally perform better for batch learning, whereas active approaches have been shown to work well in online settings, i.e. when only one instance is presented at each time stamp t. [7].

As anticipated, in this work we consider an online learner constantly trained and updated on the available data pairs. Hence, we start from a passive approach based on the loss induced from the machine learning process, but also consider active learning procedures to detect changes in the external environment. This procedure

yields to an hybrid algorithm mixing the two paradigms. The model embedding this method is able to interact with the sampling procedure by accelerating or decreasing it according to its own requirements. We started from an online approach and exploiting the non-stationary assumptions on the external environment we end up with models employing active learning strategies.

Chapter 3

Mathematical Instruments

3.1 Learning Paradigms

In this chapter, we present a comprehensive catalog of the learning methods adopted to solve the problem of the thesis. As statistician, my exposition will naturally reflect my background, but I preferred to use the modern language of machine learning which highly influenced my thinking in the last two years.

Hence, in the following the more classically called independent variables may also be named predictors, inputs or features, as they are mostly known in the pattern recognition literature [13]. The features are usually presented by examples of a set of variables and stored in the data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. For each example, the goal is to use the inputs to predict the values of the output. This technique is called supervised learning in the statistical learning field [14]. It goes without saying that also in this case the output might be called responses or, traditionally, dependent variables.

3.1.1 Supervised Learning

The outputs, indicated by y_i, vary in nature among the examples.

In this thesis the output is qualitative and assumes values in the finite set $Y = \{\textit{packet received}, \textit{packet not received}\}$. Thus, there is no explicit ordering in

the classes, and in fact descriptive labels rather than numbers are often used to denote the classes, nevertheless we represented them numerically by using codes. We are in the easiest case, since only two categories are inferred and from now on they will be represented as -1 and 1, or else by [1] 0 and 1. For reasons that are apparent, such numeric codes are sometimes referred to as targets.

As already stated in Section 3.1, we will denote an input variable by the symbol $\mathbf{X}$, indicating a matrix with n rows (examples or empirical observations) and d columns (variables). If $\mathbf{X}$ is a vector, it is denoted as $\boldsymbol{x}$ and its components can be accessed by subscripts x_j. Qualitative outputs will be denoted by Y. We use uppercase letters such as X or Y when referring to the generic aspects of a variable, random variable or finite set. Observed values are written in lowercase. We will also use script lettering to indicate vector spaces. Matrices are represented by bold uppercase letters; it is clear that a set of n input d-vectors $\boldsymbol{x}_i$, $i = 1, ..., n$ would be represented by the $n \times d$ matrix $\mathbf{X}$.

In general, vectors will not be bold, except when they have d components; this convention distinguishes a n-vector of inputs x_j for the jth observation from the d-vector $\boldsymbol{x}_i$ consisting of one example of all variables X_j, $j = 1, ..., d$. Since all vectors are assumed to be column vectors, the ith row of $\mathbf{X}$ is $\boldsymbol{x}_i^\top$, i.e. the vector transpose of $\boldsymbol{x}_i$.

Now that the notation has carefully been explained, the supervised learning framework is formally introduced from a mathematical point of view. Consider a functional dependency that maps points from an input space $\mathrm{X} \in \mathbb{R}^{d_{in}}$ into an output space $\mathrm{Y} \in \mathbb{R}^{d_{out}}$. In a typical supervised learning task we are given a training set T of n input-target pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, i.e.

$$T = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \,:\, \boldsymbol{x}_i \in X,\, \boldsymbol{y}_i \in Y \text{ and } i = 1, ..., n\}. \tag{3.1}$$

The goal of supervised learning is to define a mapping f or hypothesis h and produce a prediction $\hat{\boldsymbol{y}} = f(\boldsymbol{x}) = h(\boldsymbol{x})$, which correctly predicts the true output $\boldsymbol{y}$. When

[1] Notice that this is just a transformation of the previous $y_{\text{old}} \in \{-1, 1\}$. Now we set $y_{\text{new}} = 1 + y_{\text{old}} \in \{0, 1\}$

dealing with classification problems, the input space X is divided into K subsets $X_1, ..., X_K \in X$ such that $X_i \cap X_j = \emptyset$ for all $i, j = 1, ..., K$ and $i \neq j$. Now the task is to assign a given input vector $\boldsymbol{x}$ to the subset it belongs to. The basic form of any classification task is the binary classification, where there are two sets $X_1, X_2 \in X$ such that $X_1 \cap X_2 = \emptyset$ and we want to determine whether the input vector $\boldsymbol{x}$ belongs to X_1 or X_2. In this case, the training set is formally defined as

$$T_{\text{binary}} = \{(\boldsymbol{x}_i, y_i) \,:\, \boldsymbol{x}_i \in X,\, y_i \in \{-1, +1\} \text{ and } i = 1, ..., n\}, \tag{3.2}$$

with the two subsets X_1 and X_2 labelled by $+1$ and -1, respectively.

We need data to construct prediction rules, often a lot of it. We thus suppose we have available the training data $(\boldsymbol{x}_i, y_i)$, $i = 1, ..., n$, with which building our prediction rule.

3.1.2 Online Learning

The notation introduced in Section 3.1.1 for a binary classification task is typically adopted when the training data, i.e. the data matrix $\mathbf{X}$, is presented as a batch set. Thus, all the pairs $(\boldsymbol{x}_i, y_i)$ for all $i = 1, ..., n$ are available from the beginning of the analysis.

This is a common setup for many practical analysis and research problems; however, it is also possible to cope with a problem whose data instances are not totally available from the beginning, but instead revealed by time. In this case, retrieving the mapping f is usually more challenging and in many situations it stands for a more realistic analysis, since the batch dataset is usually collected at different intervals of times and studied without considering this aspect. The learning algorithms studied within this framework are known as online learning algorithms because they operate on a sequence of data examples with time stamps.

At each step t, the learner h receives an incoming example $\boldsymbol{x}_t \in \mathcal{X}$ in a d-dimensional vector space, that is, $\mathcal{X} \in \mathbb{R}^d$. Then, it first attempts to predict the class label of the incoming instance, which could be done by adopting a classical machine learning model. For example, we could choose the sign function $\hat{y}_t =$

Algorithm 1 Online Learning Framework for Binary Classification

Input: an initialized vector $\boldsymbol{w}_0$
Output: a sequence of predicted labels

for each time stamp t **do**
  the learner receives an incoming instance $\boldsymbol{x}_t \in \mathcal{X}$
  the learner predicts the class label $\hat{y}_t$
  the genuine class label is revealed from the external environment $y_t \in \mathrm{Y}$
  the learner calculates the suffered loss $\mathcal{L}(h(\boldsymbol{x}_t; \boldsymbol{w}_t), y_t)$
  if $\mathcal{L}(h(\boldsymbol{x}_t; \boldsymbol{w}_t), y_t) > 0$ **then**
    the learner updates the classification model, i.e.

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \Delta(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t)) \tag{3.3}$$

$\mathtt{sgn}(f(\boldsymbol{x}_t; \boldsymbol{w}_t)) = \mathtt{sgn}(\boldsymbol{w}_t \cdot \boldsymbol{x}_t) \in \mathrm{Y}$, and $\mathrm{Y} = \{-1, +1\}$. Only after making the prediction, the true label $y_t \in \mathrm{Y}$ is revealed and the learner computes the loss $\mathcal{L}(\hat{y}_t, y_t)$ based on some criterion, e.g. a function or a metric measuring the difference between the learner's prediction and the true label y_t. Finally, based on the result of the loss, the learner decides whether, when and how to update the classification model at the end of step t. We define $\Delta(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t))$ to be a general update of the model and present in the Algorithm 1 the algorithmic framework giving an overview of the majority of the online learning algorithms.

Notice that when adopting this kind of learning paradigm, the pair $(\boldsymbol{x}_t, y_t)$ is available only at time t. When considering the successive time stamp, both inputs and output are vanished and must be embedded in the model's parameters $\boldsymbol{w}_t$, which tries to adapt to the new available information while keeping track of the previous instances. It goes without saying that different online learning algorithms are distinguished in terms of various definitions and designs of both the loss function $\mathcal{L}(\cdot)$ and updating rule $\Delta(\cdot)$.

3.1.3 Function Approximations

The learning paradigm of the previous sections has been the motivation for research into the supervised learning problem in the fields of machine learning, with analo-

gies to human reasoning, and neural networks, with biological analogies to the brain. The approach developed in applied mathematics and statistics has been from the perspective of function approximation and estimation. Herein, the data pairs $(\boldsymbol{x}_t, y_t)$ are viewed as points in a $(d+1)$-dimensional Euclidean space. The function $f(\boldsymbol{x}_t)$ has domain equal to the d-dimensional input subspace, and is related to the data via a model such as $y_t = f(\boldsymbol{x}_t) + \varepsilon_t$, where ε is a noise satisfying some precise assumptions. The goal is to obtain a useful approximation of $f(\boldsymbol{x}_t)$ for all t in some region of $\mathbb{R}^d$, given the representations in $\mathcal{X}$. Although somewhat less glamorous than the learning paradigm, treating supervised learning as a problem in function approximation encourages the mathematical concepts of probabilistic inference to be applied to the problem and, as will be described in Section 4, to combine this formulation with optimization techniques based on gradient descent strategies. Investigating the problem by exploiting the geometrical properties of the Euclidean space allows us to exploit non linear function approximation before applying linear models. This is one of the main strategy adopted in this thesis, since it yields to learners that are simple, fast and hopefully more accurate than classical linear models; characteristics that are essential for IoT industry and smart sensors. As a matter of fact, notice that $f(\boldsymbol{x}_t)$ could be a function of any type when we do not treat the problem as a function approximation on. For example, we could retrieve a Multi Layer Perceptron if we consider $f(\cdot)$ to be a chain of functions, or also a more advanced non-linear learner of the inputs $\boldsymbol{x}_t$, such as a Convolutional Neural Network. These models are characterized by an high level of complexity, both in the training and prediction phases, resulting in being not optimal for IoT applications. In this thesis, we want to exploit models that are simple, since they have to be applied to practical scenarios and satisfy strict requirements of both computational complexity and storage, but that at the same time are able to capture the non linear structures of the input space, in case they are presented. For this reason, combining a non linear basis transformation with a linear model is attempted to be a good trick to maintain simplicity and gain higher performances. The approximations studied have associated a set of parameters θ that can be modified to suit the data at hand.

For example, the linear model

$$f(\boldsymbol{x}_t) = \boldsymbol{x}_t^\top \boldsymbol{w}_t \tag{3.4}$$

has $\theta = \boldsymbol{w}_t$. Furthermore, a model of this kind can easily be adapted to fit also a bias term (intercept) w_0 by considering $\boldsymbol{x}_t = [1, x_1, \cdots, x_d]^\top$ and by including the term w_0 into the vector $\boldsymbol{w}_t$ at the first entry.

Kernel Transformations

The model $f(\boldsymbol{x}_t) = \boldsymbol{x}_t^\top \boldsymbol{w}_t$ is linear since it performs a dot product between vectors. When taking into account a polynomial kernel transformation on $\boldsymbol{x}_t$, we get a model that is linear only in the coefficients, but not linear in the variables, cause the transformation is applied only to the vector of features. It is well known that linear models fitted to a basis expansion of the original inputs allow arbitrarily complex models [14].

We consider a polynomial kernel transformation of order 2. It is defined as $k(\cdot)$, and it takes in input $\boldsymbol{x}_t$ and returns as output a transformation of the original vector, whose entries include for each term j, $j = 1, ..., d$ the entry x_j itself, the single products with all the others input $x_{i \neq j}$ and the square term x_j^2. For example, after fixing t, consider the polynomial model given by

$$y = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_2 + w_4 x_2^2 + w_5 x_1 x_2 + \varepsilon, \tag{3.5}$$

i.e. a complete model of second order over the variables X_1, X_2. Notice that w_0 is multiplied by 1.

In the following a basis transformation of this type will be considered. In addition, the degree of the transformation has been fixed to 2 in order to avoid a huge number of computations that can easily heavy the hardware performance and slow down the speed of the learner's prediction, since the number of coefficients that has to be fitted in this way grows at a polynomial rate.

Finally, notice that with a change of notation, also the previous model can be restated as a linear one, since it is said to be linear in the coefficients. As a matter

of fact, by fixing $\boldsymbol{x}_t \in \mathbb{R}^d$ we can apply the polynomial basis transformation of the second order and obtain a vector $k(\boldsymbol{x}_t) \in \mathbb{R}^{2^2(d+1)}$. In practice, in every program we can reassign $\boldsymbol{x}_t = k(\boldsymbol{x}_t)$ and apply the same dot product of equation (3.4). It goes without saying that also $\boldsymbol{w}_t$ is now in the space of dimension $2^2(d+1)$.

3.2 Statistical Decision Theory

So far, we have been talking about a general learner h. In this Section we will present two families of machine learning models usually studied in the literature and describe which one has been used to tackle the thesis problem. In Chapter 4, we will analyse how they have been trained and slightly modify to operate in an online setting, characterized by non-stationary environments.

3.2.1 Discriminative vs Generative Classifiers

In the statistical learning theory, a common practice for training learning machines is to assume that there is a fixed but unknown probability distribution $\mathcal{P}$ over $\mathcal{X} \times \mathrm{Y}$ such that pairs of examples $(\boldsymbol{x}_t, y_t)$ are i.i.d. samples from it.

Then, we define

$$p(\mathrm{Y} = y_t | \mathcal{X} = \boldsymbol{x}_t) = \frac{p(\mathcal{X} = \boldsymbol{x}_t, \mathrm{Y} = y_t)}{p(\mathcal{X} = \boldsymbol{x}_t)} \tag{3.6}$$

$$= \frac{p(\mathcal{X} = \boldsymbol{x}_t | \mathrm{Y} = y_t) p(\mathrm{Y} = y_t)}{p(\mathcal{X} = \boldsymbol{x}_t)} \tag{3.7}$$

$$= \frac{p(\boldsymbol{x}_t | y_t) p(y_t)}{p(\boldsymbol{x}_t)} \tag{3.8}$$

$$\propto p(\boldsymbol{x}_t | y_t) p(y_t), \tag{3.9}$$

where $p(\mathrm{Y} = y_t | \mathcal{X} = \boldsymbol{x}_t)$, for short $p(y_t | \boldsymbol{x}_t)$, denotes the conditional probability of observing y_t given $\boldsymbol{x}_t$, i.e. the probability of classifying y_t given the observed instance $\boldsymbol{x}_t$. In equation (3.6), the first term is retrieved by exploiting the Bayes theorem and then by further decomposing the joint probability $p(\boldsymbol{x}_t, y_t) = p(\boldsymbol{x}_t | y_t) p(y_t)$. Based

on this decomposition, the generative and discrimnative learning methods are introduced. Generative classifiers learn a model of the joint probability $p(\boldsymbol{x}_y, y_t)$, their predictions are retrieved by using Bayes rules to calculate $p(y_t|\boldsymbol{x}_t)$ and selecting the most likely label y_t. Discriminative classifiers model the posterior $p(y_t|\boldsymbol{x}_t)$ directly, or learn a direct map from inputs $\boldsymbol{x}_t$ to the class labels [15]. Among the differences between these two methods, we report first of all that in general generative classifiers are found to be simpler and easier to train, in particular from a computational point of view, since they avoid the intermediate step of modelling $p(\boldsymbol{x}_t, y_t)$ [16].

Moreover, discriminative learning has lower asymptotic error while a generative classifier may also approach its higher asymptotic error much faster [15]. This is usually true when the number of training examples is increased. However, since we are operating in an online setup in which observations are presented by time stamps, i.e. the training examples are slowly observed by the learner and sometimes are never available, generative classifiers have been preferred. In addition, a discriminative classifier yields to high performances only when assumptions on the data generating process (DGP) $\mathcal{P}$ are correct. Although in many practical cases it is possible to study the phenomena under investigation, in this thesis multiple scenarios of the same problem are tested, hence different DGPs are expected to be presented yielding to an higher level of complexity when trying to identify the underlined changing DGP and adapting the model according to it.

As we will see in Section 4.2, a Logistic Regression Classifier with a stochastic gradient descent update and an online version of Support Vector Machines (SVMs) are implemented to cope with the thesis problem. In particular, the former learner is also adapted to deal with processes that are changing through time, while the latter already embeds this feature. Finally, as anticipated in Section 2, this kind of models belong to the literature of learning in non stationary environments [4], from which two methods are retrieved and formally presented in the next Section 3.2.2.

3.2.2 Learning in non Stationary Conditions

The aim of this section is to develop systematic tools to promote interplay between theory and practice. We introduce general-purpose change detection methods that

can be easily deployed across different domains, yet adapted to our problem-specific structure.

Jeffreys Confidence Intervals

As reported in [17], the erratic behavior of the coverage probability of the standard Wald confidence interval for a binomial proportion is too risky if applied to deal with a probabilistic problem whose proportion is close to 0 or to 1 or also when dealing with intervals estimation of small samples, which are both common situations in our problem setup. This leads us to take into consideration an alternative interval, i.e. the Jeffreys Confidence Intervals (JCI), which have a Bayesian derivation framework.

In statistics, two kinds of estimands can be distinguished. The first are unobserved quantities for which statistical inferences can be made, i.e. potentially observable quantities, for instance think at future observations of a probabilistic process. The second are quantities that are not directly observable, that is, parameters that govern the hypothetical process leading to the observed data (for example, regression coefficients). To the first group of estimands belong the so called posterior distributions $p(\theta|y)$, which combines the background information with information in data. To the second belong the prior distribution $p(\theta)$, which attempts to quantify the epistemic uncertainty before observing the process. The process of Bayesian inference involves passing from a prior to a posterior distribution, in general trying to exploit mathematical relations that hold between these two distributions.

Conjugacy is one of this relation, formally defined as follows. If $\mathcal{F}$ is a class of sampling distributions $p(y|\theta)$, and $\mathcal{P}$ is a class of prior distributions for θ, then the class $\mathcal{P}$ is conjugate for $\mathcal{F}$ if

$$p(\theta|y) \in \mathcal{P} \text{ for all } p(\cdot|\theta) \in \mathcal{F} \text{ and } p(\cdot) \in \mathcal{P}. \tag{3.10}$$

This definition is formally vague [18], since if we choose $\mathcal{P}$ as the class of all distributions, then $\mathcal{P}$ is always conjugate no matter what class of sampling distributions is used. We are most interested in natural conjugate prior families, which

arise by taking $\mathcal{P}$ to be the set of all densities having the same functional form as the likelihood $p(y|\theta)$. Conjugate prior distributions have the practical advantage, in addition to computational convenience, of being interpretable as additional data, but this aspect is not of interest for our approach.

Beta distributions are the standard conjugate priors for binomial distributions and it is quite common to use beta priors for inference on a proportion p [19]. Hence, we suppose that a generic random variable is Bernoulli distributed, i.e. $X \sim Bin(T, p)$ and suppose p has a prior distribution $Beta(a_1, a_2)$. Then, the posterior distribution of p is $Beta(X + a_1, T - X + a_2)$ and a $100(1-\alpha)\%$ equal-tailed Bayesian interval is straightforwardly given by

$$\left[B\left(\frac{\alpha}{2}; X + a_1, T - X + a_2\right),\ B\left(1 - \frac{\alpha}{2}; X + a_1, T - X + a_2\right)\right], \tag{3.11}$$

where $B(\alpha, m_1, m_2)$ denotes the α quantile of a $Beta(m_1, m_2)$ distribution.

The well-known Jeffreys prior and the uniform prior are each a beta distribution. The non informative Jeffreys prior is of particular interest to us, in fact historically, Bayes procedures under non informative priors have a track record of good frequentist properties [20].

In this problem, the Jeffreys prior is $Beta(1/2, 1/2)$. We will adopt this initialization and retrieve

$$CI_J = [L_J(x), U_J(x)], \tag{3.12}$$

where $L_J(0) = 0$, $U_J(T) = 1$ and otherwise

$$L_J(x) = B\left(\frac{\alpha}{2}; X + \frac{1}{2}, T - X + \frac{1}{2}\right), \tag{3.13}$$

$$U_J(x) = B\left(1 - \frac{\alpha}{2}; X + \frac{1}{2}, T - X + \frac{1}{2}\right). \tag{3.14}$$

In practice, the computation of the interval can be easily performed by sampling from a beta distribution, i.e. an operation fast and feasible even for an IoT device, although computationally more expensive than the classical Wald test for proportions, which must be excluded since not optimal for our problem, as already

Algorithm 2 PAGE HINKLEY TEST (PH)

Input: values $a_1, ...a_t$, magnitude threshold δ, detection threshold ϵ.
Output: a detection time stamp t_{PH}.

1: **for** each time stamp t **do**
2: Compute

$$\bar{a}_t = 1/T \sum_{t=1}^{t} a_t$$
$$U_T = \sum_{t=1}^{T} (a_t - \bar{a}_T - \delta)$$
$$m_T = \min(U_T,\, t = 1, ..., T)$$

3: **if** $\text{PH}_T = U_T - m_T > \epsilon$ **then**
4: **return** t_{PH}

stated.

Consider that sampling a value b associated to a probability distribution density $f_B(b)$ is an operation that can be implemented in $\mathcal{O}(T)$ operations. In a simple implementation, generating quantiles from a beta distribution is a trivial task in this way. But as we will see in Section 4, when the sensor is not memory demanding and the sample size T is given as input to the proposed algorithms, we can sample both lower and upper bounds by storing only the required values from the statistical table presented in [17], a cheaper operation performed in $\mathcal{O}(1)$ complexity.

When we assume that a learner h is operating in non stationary conditions, the Jeffreys confidence intervals will be exploited in order to infer changes in the DGPs.

Page Hinkley Test

The Page-Hinkley test (PH) [21,22] is a sequential analysis technique typically used for monitoring change detection. This method can be easily implemented, as resumed in Algorithm 2.

It allows efficient detection of changes in the normal behaviour of a process which is established by a model, although it was originally designed to detect a change in the average of a Gaussian signal. This test considers a cumulative variable U_T defined as the cumulated difference between the observed values and their mean till

the current moment:

$$U_T = \sum_{t=1}^{T} (a_t - \bar{a}_T - \delta), \tag{3.15}$$

where $\bar{a}_t = \frac{1}{T} \sum_{t=1}^{t} a_t$ and δ corresponds to the magnitude of changes that are allowed.

It computes the minimum value of $U_t : m_T = \min(U_t,\, t = 1, ..., T)$ and monitors the difference between U_T and m_T : PH $= U_T - m_T$. When the difference PH is greater than a given threshold ϵ a change in the distribution is assigned. The threshold ϵ depends on the admissible false alarm rate. Increasing ϵ will entail fewer false alarms, but might miss or delay some changes. Controlling this detection threshold parameter makes it possible to establish a trade-off between the false alarms and the miss detections.

The PH test will be adopted in this thesis to monitor and detect variations in the performance of a learner.

Chapter 4

Problem definition and solution

In this chapter we first describe the problem experiment, its architectures and formally introduce the objective of this thesis.

Then, the algorithms that have been implemented to tackle the problem are presented.

4.1 Experiments Setup

The goals of the experiments are multiple, first of all we are looking for the optimal selection of a communication interface according to the network status, user and traffic requirements and constraints imposed by the application. We would also enable devices with more than one LPWAN/Cellular interface and select the most appropriate to use at runtime, allowing guarantees needed by the data to be transmitted and opportunistically benefit from low-cost, i.e. license free network technologies, when possible.

To achieve these results, an outcome function that specifies the interface to be used for transmission have to be implemented. In this thesis we do not investigate whether simultaneous transmissions among more than one interface should be preferred. Finally, the benefits of this approach are tested in a real implementation concentrate on NB-IoT and LoRaWAN technologies given that commercial off-the-shelf hardware is already available in the market.

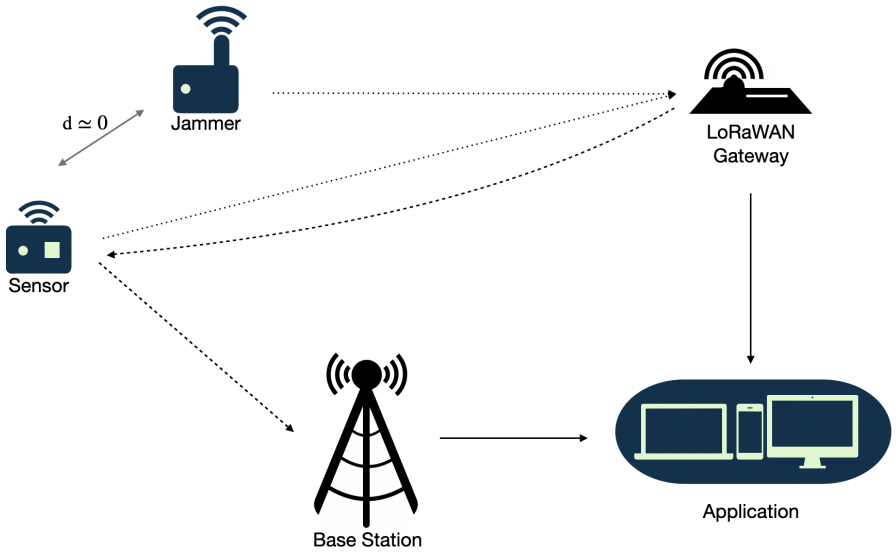

Figure 4.1: Test experiment

4.1.1 Architecture diagram

An IoT sensor, formally a Raspberry pi 3 model B with a dragino LoRa GPS Hat and an ublox NB-IoT evaluation kit, able to send packages both via NB-IoT and LoRaWAN tries to communicate with a single LoRaWAN gateway.

The experiments are conducted to replicate a realistic scenario, for this reason a jammer, i.e. a second sensor, is concurrently trying to send data to the same gateway to which the first sensor is connected. As shown in Figure 4.1, in the first experiment setup the jammer is located very near to the sensor, in particular at a distance less than one meter. Such a short space implies taking into account a distance equal to zero when dealing with network sensors configurations.

In this thesis, we will analyse this scenario and another one in which the jammer is located far away from the sensor, approximately at twenty meters of distance. Both can be considered realistic configurations. However, since the experiment tests are still in a development phase, many others configurations will be studied, such as via using more than one jammer and having each one of these located at different and changing distances. Those scenarios are not part of this thesis and will be

investigated once the data will be available. Nevertheless, as it will be shown in Section 4.2, the algorithms proposed for the former two scenarios are expected to generalize well when considering other configurations, given the high level of variance and adaptation of the machine learning techniques employed.

As already stated, both sensor and jammer are sending packets of data to the same LoRaWAN gateway, indeed notice that in Figure 4.1 the dotted lines indicating the occurrence of this communications are the thinnest, since they are not guaranteed and packets loss or collisions can actually happen in both jammer-gateway and sensor-gateway channels. On the other hand, the communications from the gateway to the sensor and from the sensor to the base station, i.e. the NB-IoT channel, is given for granted. With this approach, we ensure that whenever possible a packet containing an acknowledgment (ACK), positive or negative, can be correctly sent from the gateway to the sensor. We also assume that the communication with the base station is a service provided under payment every time it has been used. Thus, for this kind of network we would like to avoid using it and preferring the free of charge LoRaWAN gateway. Nevertheless, differently from the license free communication, it is guaranteed to work well and with a negligible error rate.

Finally, both LoRaWAN gateway and NB-IoT base station send the received information from the sensor to an application, that is considered to be a generic app installed on a mobile phone, a personal laptop or computer. This communication phase is not studied in the analysis and it is given for granted that can be performed without errors. Before proceeding, it is important to underline that the experiments described are all deployed under total control of the user.

The target variable used for tuning the decision function consists of the labels assumed by the acknowledgment send from the LoRaWAN gateway to the sensor. Notice that as stated in Algorithm 1, the learner can see the true label only after making its own prediction, i.e. after selecting the channel through which communicating. The definition of a learner (the decision function) in this case is based on the sensor' parameters, described in the next Section 4.1.2.

The objective of the thesis is to build a learner able to choose the LoRaWAN channel by favouring a trustworthy communication, so constantly exploiting as much

as possible the free-license service. We attempt to build a machine learning model having high performances also when multiple sensors are adopting our decision function, i.e. in a network scenario. Consequently, we assume that the learner cannot see all the targets, which is highly unrealistic and possible only on a controlled scenario. In this way, we also avoid to ending into a network with high load, which is inefficient and characterized from a huge number of packets collisions, i.e. a low packet delivery ratio.

As defined in the LoRaWAN protocol, each sensor selects whether requesting the ACK from the gateway. This feature will be exploited to develop active learning strategies and combined with those described in the previous Chapter 3.

4.1.2 Feature Extraction

At each time stamp t, the sensor present the input parameters vector $\boldsymbol{x}_t$. Specifically, we consider the logarithm of the absolute value of the Received Signal Strength Indicator (RSSI), x_1, the Spreading Factor (SF), x_2, the Trasmitting Power (TP), x_3, and the Signal to Noise Ratio (SNR), x_4. As explained in Section 2.1.2, the SF is a LoRaWAN based-parameter. We also measure the RSSI, i.e. a measurement of the power present in a received radio signal, in fact, the greater the RSSI value, the stronger the signal. This metric is usually used in sensor networks to determine when the amount of radio energy in the channel is below a certain threshold at which point the sensor could clearly transmit. Then we consider the TP of the sensor in decibel, when sending a new packet, and the SNR, defined as the ratio of signal power to the noise power, again in decibels. The former is a measure of the sensor power in transmitting a new packet, so the larger the higher the probability of packet delivery, the latter can be considered as an estimate of the external noise, for which again a large value is associated to an high probability of successful packet transmission.

In addition, we also apply a min-max transformation to x_1 and x_4, that is

$$\bar{x}_j = \frac{x_j - \min(x_j)}{\max(x_j) - \min(x_j)}. \tag{4.1}$$

This is possible since both $\min(x_j)$ and $\max(x_j)$ are known values for RSSI and SNR. This standardization method allows the learner to take in input entries that lies in the interval $[0, 1]$.

On the other hand, we perform a dummy transformation to x_2 and x_3. For instance, consider x_2 to be a categorical variable with g labels, a dummy transformation creates $g - 1$ indicator variables, each one notifying whether the label g is observed or not, the excluded label is automatically included in the intercept term and can be retrieved by exclusion.

Finally, a polynomial kernel of degree 2 is applied to the resulting vector. Since x_2 and x_3 have both three labels, we get a vector $\boldsymbol{x}_t \in \mathbb{R}^{d_{in}}$ with $d_{in} = 7$ and after applying the polynomial transformation we obtain $\boldsymbol{x}_t \in \mathbb{R}^{d_{out}}$ with $d_{out} = 28$, by following the bias inclusion procedure and the basis transformation described in Section 3.1.3.

4.2 Modelling Framework

In order to build a working decision function, at every time stamp t the vector of features $\boldsymbol{x}_t$ is used to predict whether we can transmit via LoRaWAN. This goal is challenging mainly due to lack of information about jammer's parameters, other external noise and how these two both change in time.

Moreover, we need to maintain the load of the network as low as possible in terms of acknowledgements requested by the sensor. Thus, the true label y_t is not always available, but can be requested when necessary: a trade-off between performances and number of ACKs have to be studied.

To carry out this job, we employ statistical inferences techniques. Statistical inference is the branch of statistics concerned with drawing conclusions and/or making decisions concerning a population based only on sample data, as shown in Figure 4.2.

Since we want to estimate when the sensor can transmit using LoRaWAN, we will exploit both internal information (sensor's features) and external information (targets), cause the feature vector $\boldsymbol{x}_t$ is not sufficient, in fact it does not summarize the jammer behaviour and only partially describe the external noise. When

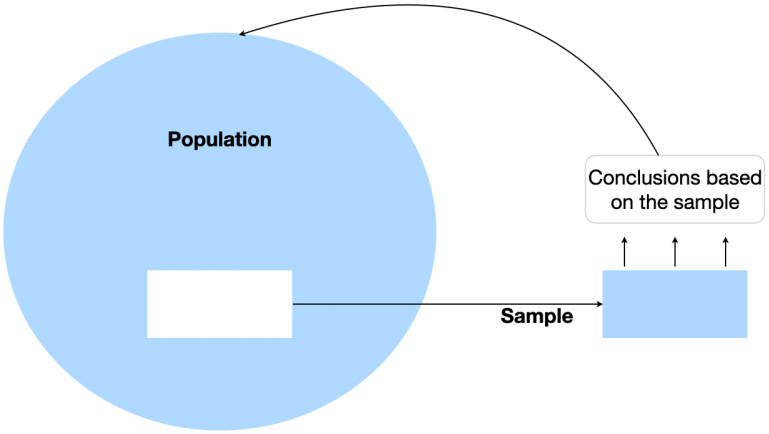

Figure 4.2: Inference at first glance.

the target variable y_t is not available it can be requested, but in cases in which it is not possible and with the aim of helping the learning activity, we apply statistical inference to estimate $p(y_t)$ and adopting some forms of smart requests for the acknowledgment. We further assume that the external environment can change during time, i.e. the DGP is not stationary. In the IoT context the non-stationarity can be caused, for example, by seasonality or periodicity effects, variations in the external environment, i.e. changes in $p(y_t)$, changes in the sensor locations or aging effects in the sensors performance. In order to minimize the load of the network, we choose to define the process to ask for the label y_t. Hence, a simple but effective sampling technique must be implemented, in otder to catch the autocorrelation of the DGP. Statistical and machine learning models can be trained using random sampling. Thus, we impose to ask a label at random with probability p_u at time t and increase p_u if necessary. This means adopting active learning strategies to get a robust and reliable estimate $\hat{p}(y_t)$.

The external environment

We assume that the sequence of target variables $y_0, y_1, ..., y_{t-1}, y_t$ are independent and identically distributed Bernoulli random variables generated with probability p_0, i.e. Y $\sim Bin(T, p_0)$. The main idea of the thesis is checking whether p_0 is constant over the time, if not we have an empirical evidence that the external environment is changing. Hence, we estimate $p(y_t)$ using p_0, so, observing non overlapping and consecutive sample windows I_k, with $k \in \mathbb{Z}$, made of T observations, and use a Jeffreys Confidence Interval to assess p_0 due to its robustness when T is small (we set $T = 10$). If two consecutive JCI are not overlapping, we evaluate p_0 as constant, otherwise a change in the environment is occurred and the learner should be adapted to the new DGP. We formalize the problem as follows.

Expected Risk Minimization

Given two spaces of objects X and Y we wish to learn the function which outputs an object $y_t \in$ Y given $\boldsymbol{x}_t \in$ X. $\boldsymbol{x}_t$ and $y_t \in \{0, 1\}$ are random input/output data sampled via $u_t \sim Be(p_u)$.

We already explained that the goal in this case is finding the prediction function $h(\boldsymbol{x}_t; \boldsymbol{w})$ (for convenience we omit t from the parameters vector), that has fixed form and is parameterized by a vector $\boldsymbol{w}$ over which our optimization will be performed. The goal in this case is finding the prediction function $h(\boldsymbol{x}_t; \boldsymbol{w})$ (i.e., the parameters w_i defining it) that minimizes the losses incurred by inaccurate predictions (also called prediction losses or prediction errors).

As already stated, a loss function indicated with $\mathcal{L}(h(\boldsymbol{x}_t; \boldsymbol{w}), y_t)$, where $h(\boldsymbol{x}_t; \boldsymbol{w})$ and y_t respectively represent predicted and true outputs, measures the difference between predicted and reals output.

Ideally, the parameters in $\boldsymbol{w}$ of our function are chosen in such a way so as to minimize the expected loss for any input-output pair. Anyway, the problem with expected risk is that we assume to know the probability distribution $\mathcal{P}$ describing the relationship between input and outputs. In practice, we never have that $\mathcal{P}$.

This is the reason why we try to just estimate the expected risk R. In supervised

learning, the problem we deal with is inferring a function from labeled data. We then want to solve the following stochastic optimization problem

$$\min_{\boldsymbol{w}} \; R(\boldsymbol{w}) = \mathbb{E}_{(\boldsymbol{x}_t, y_t)\sim p}[\mathcal{L}(h(\boldsymbol{x}_t; \boldsymbol{w}), y_t)], \tag{4.2}$$

yielding to Classifiers based on Stochastic Gradient Descent (SGDC).

4.2.1 Stochastic Gradient Descent Classifier

A gradient based approach

Besides being gradient based, stochastic gradient descent usually works under a specific type of online setting: the i.i.d. setting, where the assumption is that data are random samples from a fixed but unknown distribution. The goal is to optimize an objective function with respect to this unknown distribution. In order to gain performance guarantee, the type of objective function adopted is usually convex, leading to the so-called online convex programming setting [23].

Besides online learning, SGD can also be applied in the batch case, where instead of feeding the entire data set to the algorithm, sequentially samples of data from the batch are retrieved, by assuming this results are i.i.d.

This type of application is usually termed as the large-scale learning framework, where SGD has the advantage of a cheap computational cost. Under this framework, SGD is usually used to find the Empirical Risk Minimizer of the batch data, without having to sample all of the data set. We assume that the batch is composed of only one example and that there is a temporal dependence between observations, as well as that the sampling procedure could change through time. Online learning by itself has many different variants in terms of paradigms, such as do not assume any distributional assumption, or that data instances are generated from a drifting distribution. We are in this latter case, therefore we will adapt the SGDC to drifting distributions, that in practice are generated by changes in the external environments in which the sensor is located, via JCI estimations.

We have multiple choice for h and $\mathcal{L}$. During the project development, a variety

of options have been tested, for short herein we present the combination that better performed at the end. We select a logistic loss function with l_1 norm as regularization parameter, which is formally defined as

$$\mathcal{L}(h(\boldsymbol{x}_t; \boldsymbol{w}), y_t) = \log\left(1 + e^{-y_t h(\boldsymbol{x}_t; \boldsymbol{w})}\right) + \lambda \left\|\boldsymbol{w}\right\|_1, \tag{4.3}$$

where $h(\boldsymbol{x}_t; \boldsymbol{w}) = \boldsymbol{x}_t^\top \boldsymbol{w}$ is a simple dot product, as explained in Section 4.1.2, and λ is the regularization parameter.

Regularization terms are usually considered to improve the generalization performance, i.e. the performance on new, unseen data. Intuitively, we can think of regularization as a penalty against complexity. Increasing the regularization parameter penalizes large weight coefficients. Our goal is to prevent that our model picks up peculiarities, i.e. noise coming from the given data. Again, we don't want the learner to memorize each pair of instances, we want a model that generalizes well to new, unseen data.

In more specific terms, we can think of regularization as adding, or increasing the already present bias if our model suffers from high variance (i.e., it overfits the training data). On the other hand, too much bias will result in underfitting. An indicator of high bias, in our case, is that the model shows a low performance for both the seen and future pairs of observations. The introduction of the l_1 norm as a regularization term is of particular importance for us. In mathematics and signal processing it is usually known as a type of Tikhonov regularization, but this technique has been formally introduced and applied to statistical and data science problems by [24], with the name LASSO (Least Absolute Shrinkage and Selection Operator). The l_1 norm is a special and useful trick which induce sparsity by shrinking the vector of parameters $\boldsymbol{w}$.

A sparse statistical model is one having only a small number of nonzero weights. This is a valuable feature for our problem setup, cause a sparse model can be much easier to estimate than a dense model. Therefore, the sparsity assumption allows us to perform an automatic features selection. This could seem strange, since we initially moved to an higher dimensional space $\mathbb{R}^{d_{\text{out}}}$ with $d_{\text{out}} = 28$. However, it is

a well known practice in statistics to fit a model only with data variables that are explanatory and try to avoid useless independent variables. The initial polynomial trick allowed us to move from a linear to a non linear model, but it could have added independent variables that are not helpful for the learner. We will avoid this problem by inducing sparsity into the vector $\boldsymbol{w}$, after which only some entries w_i are nonzero, depending on the choice of the regularization parameter. A usual practice to tune the learner and find the optimal regularization parameter λ is to implement a cross validation technique. This process cannot be adopted when dealing with our online setup since we would loose the autocorrelation of the process. Hence, as it is common in the training process of the neural networks field, we proceeded by trials and errors and end up finding $\lambda = 0.001$ to be a good value, yielding to higher performances and reducing the dimensionality of $\boldsymbol{x}_t$ to $d_{\text{out}} = 18$, on average. Notice that the LASSO procedure is classically studied in a batch setup, which is not our case. We propose the following online training from an optimization perspective, which is particularly suited for IoT applications.

Online LASSO

The learner h has a fixed form, but it is updated every time a new pair $(\boldsymbol{x}_t, y_t)$ is available, using a batch of size one. A trivial update would be

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta_t \frac{\partial}{\partial \boldsymbol{w}} \Big[\mathcal{L}_t(h(\boldsymbol{x}_t; \boldsymbol{w}), y_t) - \lambda \sum_i |w_i| \Big], \tag{4.4}$$

where η_t is the step size and should be carefully chosen.

When applied feature-wise the last term on the right side is not differentiable in case of an entry equal to zero, so we use a gradient truncated method with a cumulative formulation [25] defined as

$$w^i_{t+\frac{1}{2}} = w^i_t + \eta_t \frac{\partial \mathcal{L}_t(h(\boldsymbol{x}_t; \boldsymbol{w}), y_t)}{\partial \boldsymbol{w}} \bigg|_{\boldsymbol{w}=w_t}, \tag{4.5}$$

$$w^i_{t+1} = \begin{cases} \max(0, w^i_{t+\frac{1}{2}} - (u_t + q^i_{t-1})) & \text{if} \quad w^i_{t+\frac{1}{2}} > 0 \\ \min(0, w^i_{t+\frac{1}{2}} + (u_t - q^i_{t-1})) & \text{otherwise} \end{cases}, \tag{4.6}$$

Algorithm 3 ONLINE LOGISTIC REGRESSION WITH LASSO UPDATE

Input: $u = 0$ and $\eta = 0.9$, $w_i = 0$ and $q_i = 0$ for all i.
Output: Updated values of u, w_i, q_i for all i.

for each time stamp t **do**
  $u \leftarrow u + \eta\lambda$
  for i in each feature of the current instance **do**
    Compute $$w^i_{t+\frac{1}{2}} \leftarrow w^i_t + \eta_t \frac{\partial \mathcal{L}_t(h(\boldsymbol{x}_t; \boldsymbol{w}), y_t)}{\partial \boldsymbol{w}}$$
    $z \leftarrow w_i$
    if $w_i > 0$ **then**
      $w_i \leftarrow \max(w_i - (u + q_i))$
    else if $w_i < 0$ **then**
      $w_i \leftarrow \min(w_i + (u - q_i))$
    $q_i \leftarrow q_i + (w_i - z)$

where $u_t = \lambda \sum_0^t \eta_t$ and $q^i_t = \sum_0^t (w^i_{t+1} - w^i_{t+\frac{1}{2}})$ are the total l_1 penalty that each weight could have received up to t and the total l_1 penalty w^i has actually received. We set $\eta_t = \eta$ since the model needs to constantly learn, in particular $\eta = 0.9$. Also in this case, a trial and errors procedure has been adopted to optimize this parameter.

The updating process is divided into two steps. First, the weight is updated without considering the l_1 penalty term. Then, the l_1 penalty is applied to the weight to the extent that it does not change its sign. In other words, the weight is clipped when it crosses zero. According to [25] the obvious advantage of using this method is that we can expect many of the weights of the features to become zero during training.

Another important advantage of this method, especially for our problem, is that it allows us to perform the application of l_1 penalty in a lazy fashion, so that we do not need to update the weights of the features that are not used in the current sample. This leads to a way faster training when the dimension of the feature space is large, or in our case, without a huge feature dimension, but performed by an IoT sensor with cheap computational capacity.

Finally, this update procedure can be easily implemented by following the pseu-

docode presented in Algorithm 3.

4.2.2 Passive Aggressive Classifier

Instead of a gradient based approach, in this Section we present a model retrieved by exploiting the margin separation method, a technique introduced by [16] which eventually revolutionized the Statistics literature. The proposed method is known as Passive Aggressive Classifier [26] and belong to the class of online learning algorithms that do not make any assumption on the DGP governing the experiment. As it will be soon evident, a model belonging to this paradigm is characterized by high performances while usually being much simpler, in particular from the computational point of view. In this thesis, we choose to present one algorithm for each paradigm, leading to an interesting comparison that will be presented in Section 5. We evaluate the instantaneous Hinge loss function

$$\mathcal{L}(h(\boldsymbol{x}_t; \boldsymbol{w}), y_t)) = \begin{cases} 0 & \text{if } y_t(\boldsymbol{x}_t^\top \boldsymbol{w}) \geq 1 \\ 1 - y_t(\boldsymbol{x}_t^\top \boldsymbol{w}) & \text{otherwise} \end{cases}, \tag{4.7}$$

notice that $y_t \in \{-1, 1\}$ now. We proceed by solving the constrained optimization problem

$$\begin{aligned} \boldsymbol{w}_{t+1} = \underset{\boldsymbol{w} \in \mathbb{R}^p}{\operatorname{argmin}} \ & \frac{1}{2} \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + C\xi^2, \\ \text{s.t. } & \mathcal{L}(h(\boldsymbol{x}_t; \boldsymbol{w}), y_t)) \leq \xi. \end{aligned} \tag{4.8}$$

Equation (4.8) can be solved using the Lagrangian

$$L(\boldsymbol{w}, \xi, \tau) = \frac{1}{2} \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + C\xi^2 + \tau(1 - \xi - y_t(\boldsymbol{x}_t^\top \boldsymbol{w})), \tag{4.9}$$

where $\tau \geq 0$ is a Lagrange multiplier. Differentiating this equation with respect to each component w^i and later to ξ it can be proven [26] that

$$\tau_t = \frac{1 - y_t(\boldsymbol{x}_t^\top \boldsymbol{w})}{\|\boldsymbol{x}_t\|^2 + \frac{1}{2C}}, \tag{4.10}$$

Algorithm 4 PASSIVE AGGRESSIVE CLASSIFIER

Input: an initialized vector $\boldsymbol{w}_0$
Output: a sequence of predicted labels

1: **for** each $\boldsymbol{x}_t$ - incoming instance **do**
2: predict the class label $\hat{y}_t$ and wait for the true target y_t
3: set
$$\mathcal{L}(h(\boldsymbol{x}_t; \boldsymbol{w}), y_t)) \leftarrow \max(0, 1 - y_t(\boldsymbol{x}_t^\top \boldsymbol{w}))$$
4: set
$$\tau_t \leftarrow \frac{1 - y_t(\boldsymbol{x}_t^\top \boldsymbol{w}_t)}{\|\boldsymbol{x}_t\|^2 + \frac{1}{2C}}$$
5: set $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \tau_t y_t \boldsymbol{x}_t$

so that the update is simply

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \tau_t y_t \boldsymbol{x}_t. \tag{4.11}$$

In Equation (4.10) the parameter C implies a more aggressive update step and we therefore refer to C as the aggressiveness parameter. We set $C = 10^4$, favouring a model with a fast adaptation behaviour. The described method can be implemented by following the Algorithm 4.

4.3 Active Learning

We have to assume that the true target is not always available and can be requested when necessary. In order to limit the number of AKCs in a network, we generate a random variable u_t from a Bernoulli distribution with probability p_u. This process leads to a Poisson sampling and in this way we try to mitigate the probability that more than a sensor is requesting the true target at the same time.

In order to improve the performance of the classifiers, we will increase the sampling rate p_u of requesting the target variable, for the window I_{k+1}, when a decrease in the performance or a change in the external environment is detected. At the beginning of every window I_k, whose length T must be given as input, both JCI and

Algorithm 5 Active Learning Strategy

Input probability p_u; classifier h; length T of I_k; conf. level α; thresholds θ, ε
Output a sequence of predicted labels

for each $\boldsymbol{x}_t$ - incoming instance **do**
  generate $u_t \sim Be(p_u)$
  if $u_t = 1$ **then**
    request the true label y_t of instance $\boldsymbol{x}_t$
    if new window warning is signaled **then**
      start to update PH_{I_k} and JCI_{I_k}
    if current window I_k is completed **then**
      compute JCI_{I_k} on p_0, $T = |I_k|$, α and PH_{I_k} on ε
      if $JCI_{I_{k-1}} \cap JCI_{I_k} \neq 0$ **or** PH_{I_k} is not activated **then**
        maintain p_u as in input
      else ▷ concept drift/change in the performance detected
        increase p_u
    add y_t to I_k and train h on $(\boldsymbol{x}_t, y_t)$ to update $\boldsymbol{w}_t$
  if $h < \theta$ **then** ▷ h is not confident on its prediction
    increase p_u
  apply h on $\boldsymbol{x}_t$ to predict y_t
  return $\hat{y}_t$

PH methods are initialized.

Then, an higher number of targets will be requested when consecutive Jeffrey Confidence Intervals, based on a certain confidence α and the estimation of the proportion p_0 are not overlapping, and when $1 - \theta < p(\hat{y}_t = 1|\boldsymbol{x}_t) < \theta$ for the logistic model and $|\boldsymbol{x}_t^\top \boldsymbol{w}_t| < \theta$ for the passive aggressive one, with $\theta \in (0, 1]$. We choose $\theta = 0.9$ and $\theta = 0.2$, respectively. This reasoning comes from the fact that for the logistic regression, we would like to keep the learner as much as possible confident on its prediction, i.e. assigning probability of classifying a label higher than a certain threshold value θ, which should be large. A similar procedure is attributed to the PAC [26], since $|\boldsymbol{x}_t^\top \boldsymbol{w}_t|$ can be used to quantify the uncertainty of the model when it comes to classify a new instance. In our experiments, the Page-Hinkley test is applied to the estimation of the accuracy based on windows composed of one-hundred estimated labels $\hat{y}_t$ with an overlap of ninety, so that the

performance of the classifier is assessed every new $T = 10$ observations of the target variable, when considering an accuracy estimated on one-hundred labels.

The proposed active learning strategy is finally summarized in Algorithm 5. Notice that the active learning phase can correctly start after $2T$ instances and ninety targets should be available for the Page Hinkley test, the former is mandatory, for this we suggest to pre-train the model on an already available dataset, also from simulated scenarios, or simply to wait for $2T$ ACKs, which should be an easy task since T is set to be small. On the other hand, PH can be activated once the observations are available without losses in the performance. We choose to wait for one hundred true labels in order to stat the Page Hinkley estimation once the learner has presumably reached a stable accuracy.

Chapter 5

Results

In this Section we present the results of the proposed algorithms, discussing both pros and cons and eventually we give an interpretation of the outcome achieved in this work.

5.1 Active Learning in practice

In the previous Section 4.3, we presented the algorithm 5, in which all the mathematical instruments and learning models presented in the work are used to tackle the thesis problem. We would like to know whether at time t the sensor can transmit via LoRaWAN, in conditions that are not stationary and in a network scenario in which the load of packets transmitted from the gateway to end-devices should be as low as possible. We choose to exploit inference techniques and change detection methods, as well as to constantly assess the awareness of the learner in order to reduce the number of ACKs requested, that are now sent on a smart way instead of a random fashion, but increasing the rate of request if necessary.

This method necessary reduces the performance, since the sensor is never able to fully monitor the external environment. However, as we will shortly describe, the loss in the performance is negligible when compared to the reduction in ACKs requested for certain tests. We specify that we are looking for the optimal solution on a huge parameter space. We have already stated how the majority of values have

been set exploiting testing performance. However, we noticed when changing the parameter p_u could lead to a drastic change in the final behaviour of the algorithm. Hence, this chapter turns out to be particular important to investigate what would happen when the initializations of the sampling parameter are different and the thresholds modified. First of all, we would like to assess whether active learning is effectively working. In the following, we refer to a model embedding active learning when its behaviour is affected by JCI, PH and from the threshold on the confidence θ. If the model is learning without employing these methods, it is not working with any active learning strategy, i.e. it has been tested as a classical online learning algorithm, as described in Algorithm 1.

To investigate whether active learning strategies are truly working, we choose a simple metric, i.e. the accuracy. Let $\mathbb{V} = \{y_1, ..., y_V\}$ be the generic set containing the V acknowledgments associated with the true event (notice that we know V only because we are operating in a controlled scenario) and let $\mathbb{P} = \{\hat{y}_1, ..., \hat{y}_P\}$ be the set containing the P predicted labels, found using the learner chosen before running the test analysis. Each element $\hat{y}_p \in \mathbb{P}$ is individually analyzed with respect to the true target and then marked as either a True Positive (TP), a False Positive (FP), a True Negative (TN) or a False Negative (FN). Once all the TP, FP TN and FN cases have been found the accuracy can be defined as

$$\texttt{Accuracy} = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{FP} + N_{TN} + N_{FN}}, \tag{5.1}$$

where N_{TP}, N_{FP}, N_{TN} and N_{FN} are the total number of TP, FP TN and FN, respectively. In the following, we will consider the cumulative accuracy computed up to the time stamp t. Before starting any test, one should declare the initial value of p_u, i.e. the initial probability of requesting the true target at each time stamp t. This the most important parameter whose value can basically change the entire scenario. In Figure 5.1 we show the performance of a SGDC when $p_u = 0.15$ and compared with a SGDC not adopting active learning strategies, while in Figure 5.2 we show the performance of a PAC when $p_u = 0.15$ and compared with a PAC not adopting active learning strategies. Due to the randomness of the procedure, we

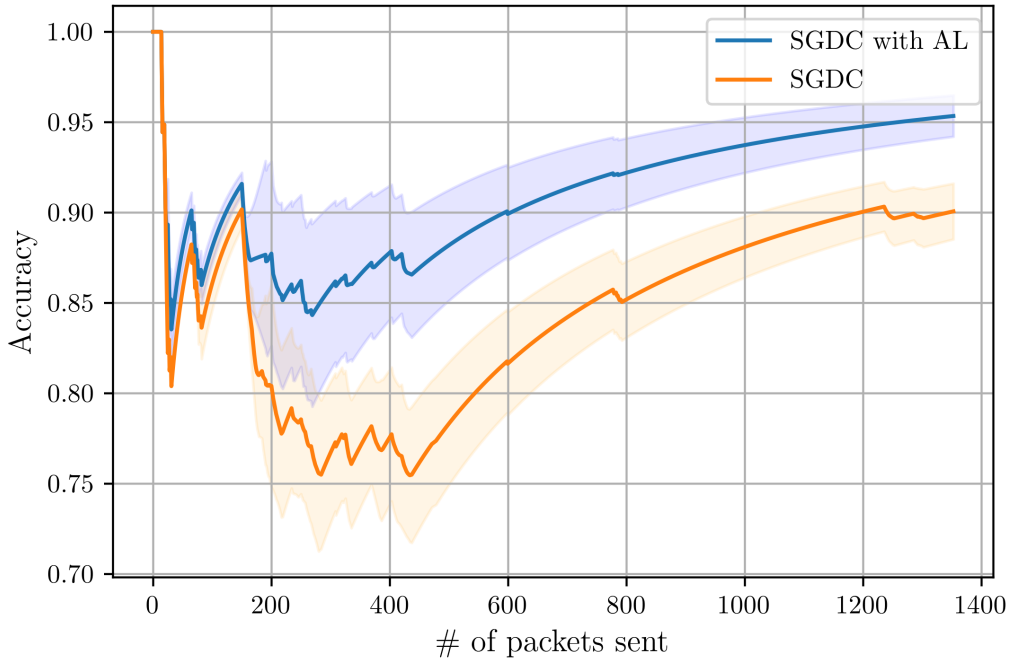

Figure 5.1: Comparison between models with Active Learning and models without it: SGDC with $p_u = 0.15$.

performed ten experiments for each approach and retrieve the average performance and its standard deviations, as shown in the confidence areas of the same plots.

It is possible to see that in particular between the two hundredth and the four hundredth packets sent, adopting the suggested active learning strategies lead to an increment in the performance, that on average is higher than 85% for SGDC. It is also evident that the SGDC is the model which benefits most from the active learning approach, while the PAC is increasing its performance but not at the same rate.

Finally, notice that SGDC with AL is achieving an higher average accuracy with respect to PAC with AL. On the contrary of the previous case, in Figure 5.3 we do not show the average results of multiple experiments, but instead one single test reporting the distribution of ACKs that have been additionally requested from the

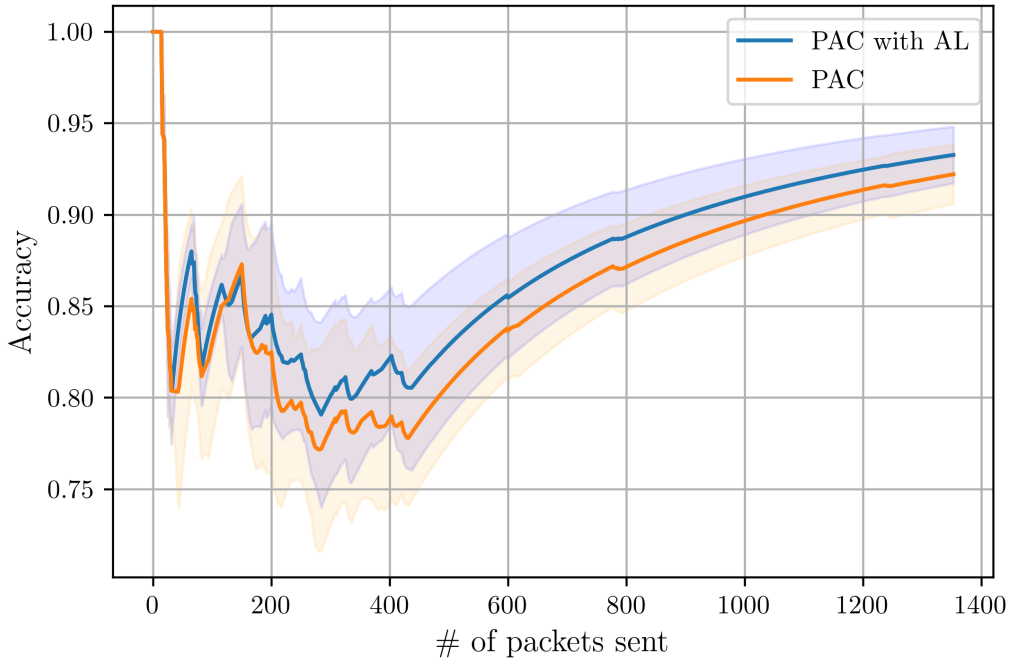

Figure 5.2: Comparison between models with Active Learning and models without it: PAC with $p_u = 0.15$.

stochastic gradient descent classifier.

It is important to highlight that in some occasions even if the model is trained on more targets values, there is an inevitable loss in the performance. However, there are some critical conditions, such as the first one hundred packets, where the additional requests completely change the final performance. Finally, notice that when adopting the active learning strategy we train the model on only thirty-six additional ACKs, in this way we guarantee the accuracy to be never lass than 80% and an evident improvement in the final performance. From the graphical analysis reported herein, but also from the same tests ran for different values of p_u, we concluded that active learning strategies is increasing the predictive accuracy of the classifiers, in particular of SGDC. In Figure 5.4 we present the performance of SGDC at different level of p_u. We state that the performance of the classifier when

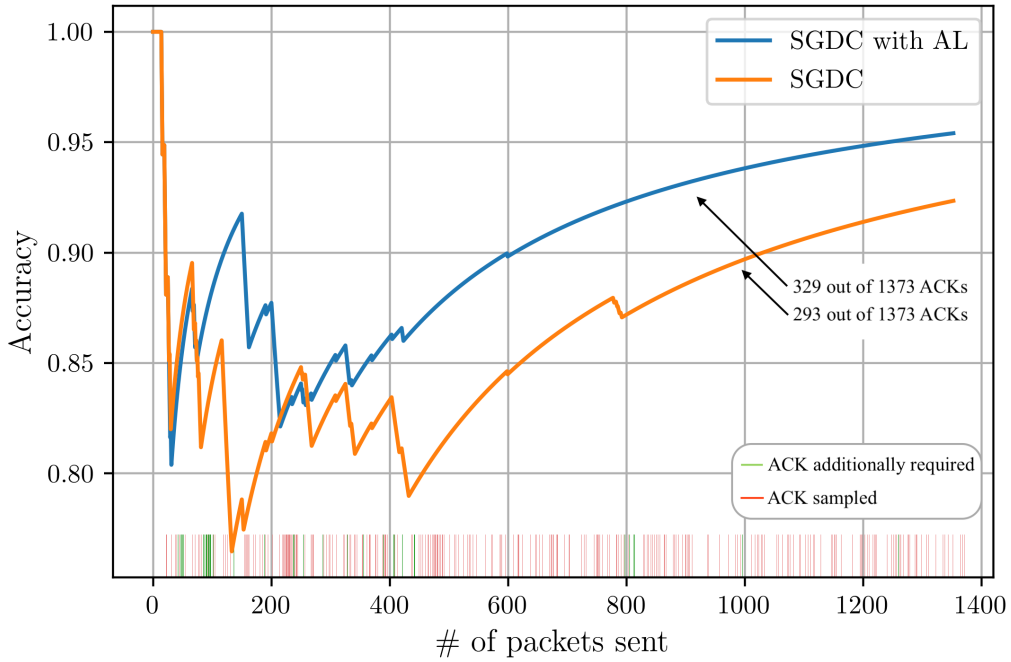

Figure 5.3: SGDC with $p_u = 0.2$ and with distribution of ACKs sampled and requested.

trained on half of the target values is basically the same of the full acknowledgment scenario. Decreasing p_u yields to an evident loss in the final accuracy, hence we further investigate the behaviour of this parameter.

5.2 Initialization of the sampling parameter

As shown in Figure 5.4, the accuracy at the end of the test is very similar among different p_u initializations. Such behavior is due to the fact that the tests investigated are characterized from a class imbalance in the target variable. When comparing SGDC and PAC among multiple p_u levels, in order to get a clearer interpretation of the output, we prefer to use the Matthews correlation coefficient (MCC), known as Pearson ϕ coefficient in statistics. This coefficient is a measure which can be

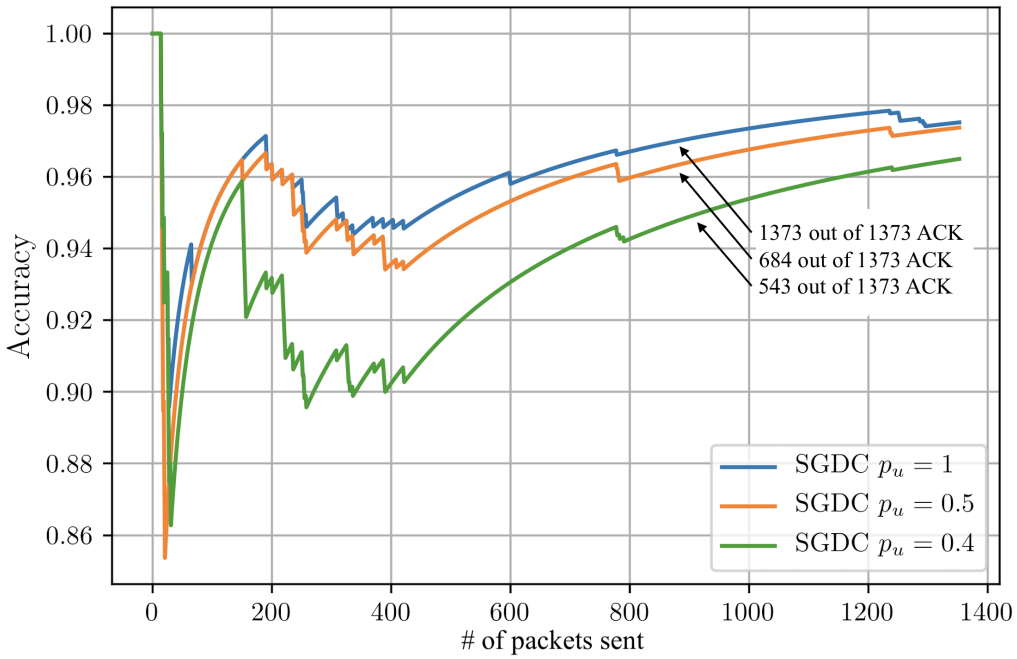

Figure 5.4: Logistic Classifier at different p_u.

used even if the classes are of very different size and is able to guarantee a fair and balance metric for the performances. It if formally defined as

$$\text{MCC} = \frac{N_{TP}N_{TN} - N_{FP}N_{FN}}{\sqrt{(N_{TP} + N_{FP})(N_{TP} + N_{FN})(N_{TN} + N_{FP})(N_{TN} + N_{FN})}}, \tag{5.2}$$

with N_{TP}, N_{FP}, N_{TN} and N_{FN} defined as in Equation (5.1). It takes into account the balance ratios of the four confusion matrix categories (true positives, true negatives, false positives, false negatives), this makes it more informative than F_1 score and accuracy in evaluating binary classification problems [27]. We stated that an active learning approach yields to an improvement in the accuracy. Now, we would like to choose which one between SGDC and PAC should be preferred according to different values of p_u. In Figure 5.5 we show the comparison between SGDC and PAC for different levels of p_u based on the MCC metric to evaluate the predictive

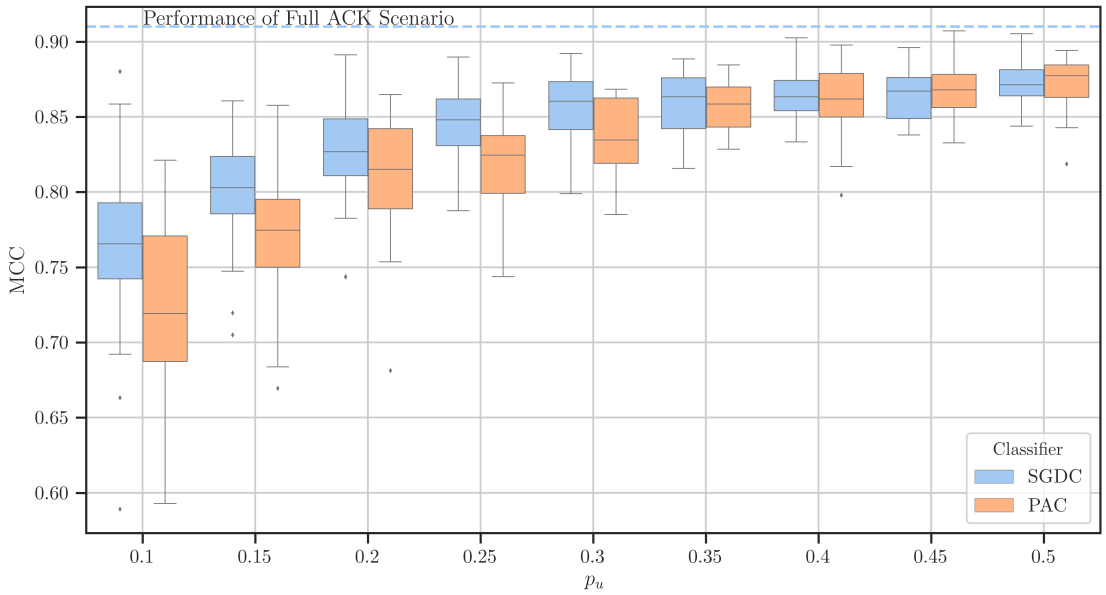

Figure 5.5: Comparison of SGDC vs PAC performance distributions at different levels of p_u

performance.

The performance of the SGDC when accessing all the targets is considered the benchmark for this analysis and is represented by the blue dashed line. In this case, every model has been tested on fifty different samples from the same test. We choose a box-plot to properly summarize the distributions of the achieved performances. In this way, we have a more informative representation, which includes both the median MCC, the interquantile range and eventually the presence of possible outliers. As anticipated, the smaller p_u the worst the MCC; but the lower the load of the network in managing the acknowledgments. We clearly see the presence of some anomalies when $p_u \in \{0.1, 0.15, 0.2\}$ and in particular a very high variance in the scored MCC, that for $p_u = 0.1$ ranges from 0.60 to 0.88. It is important to state that SGDC is performing better than PAC only for smaller values of p_u, as confirmed from a T-test on the two samples, assuming unequal variances and the MCCs values to be normally distributed for each model and p_u. This latter hypothesis is also confirmed from a Shapiro-Wilk test. For both tests the α level has been set to 0.05. Starting

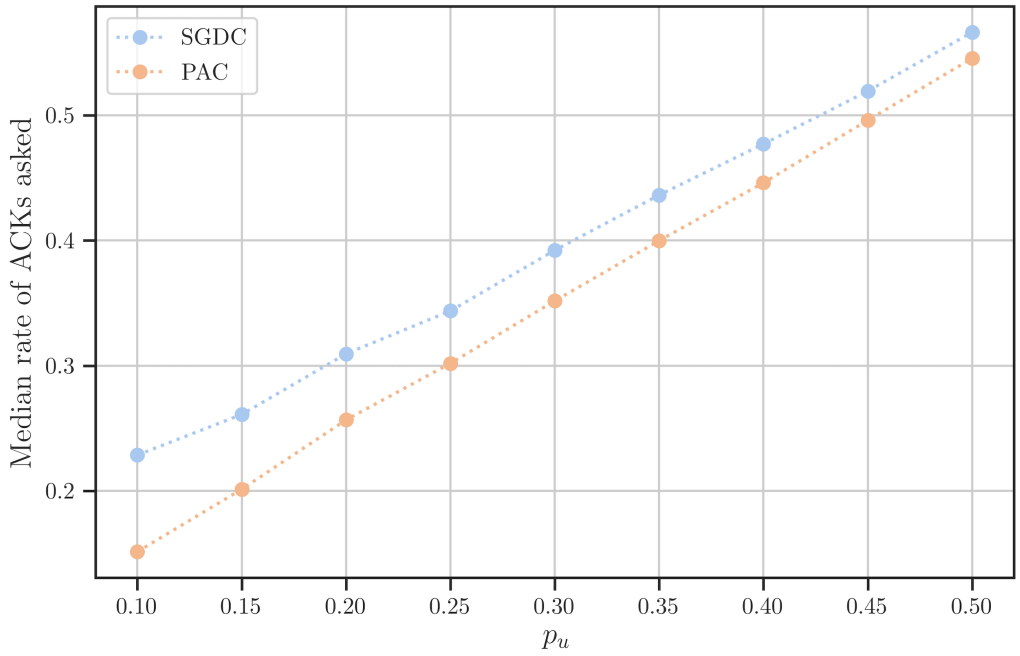

Figure 5.6: SGDC vs PAC when considering the median rate of ACKs sampled and requested for different levels of p_u.

from $p_u = 0.35$ there is no difference on the performance between the two classifier, in fact the T-test is no more significant.

Finally, we suggest to set $p_u = 0.25$ or $p_u = 0.30$ and select the logistic classifier in order to guarantee a small variance and avoid anomalies in the expected performance, but in particular to get a median MCC higher than 0.85. Smaller values of p_u should not be considered, since characterized from an high variability. When p_u is larger that 0.35, we suggest to pick the PAC classifier. The reason of this choice is motivated from both Figure 5.6 and Figure 5.7.

In the former, we plot the median rate of ACKs asked versus different p_u, in the latter the median loss in the final performance with respect to the benchmark at different values of p_u, when considering the two classifiers.

From Figure 5.6, we conclude that SGDC is requesting more ACKs if compared to PAC. In addition, by fitting a simple linear regression on the two lines, we retrieved the incremental rate of p_u when adopting an active learning strat-

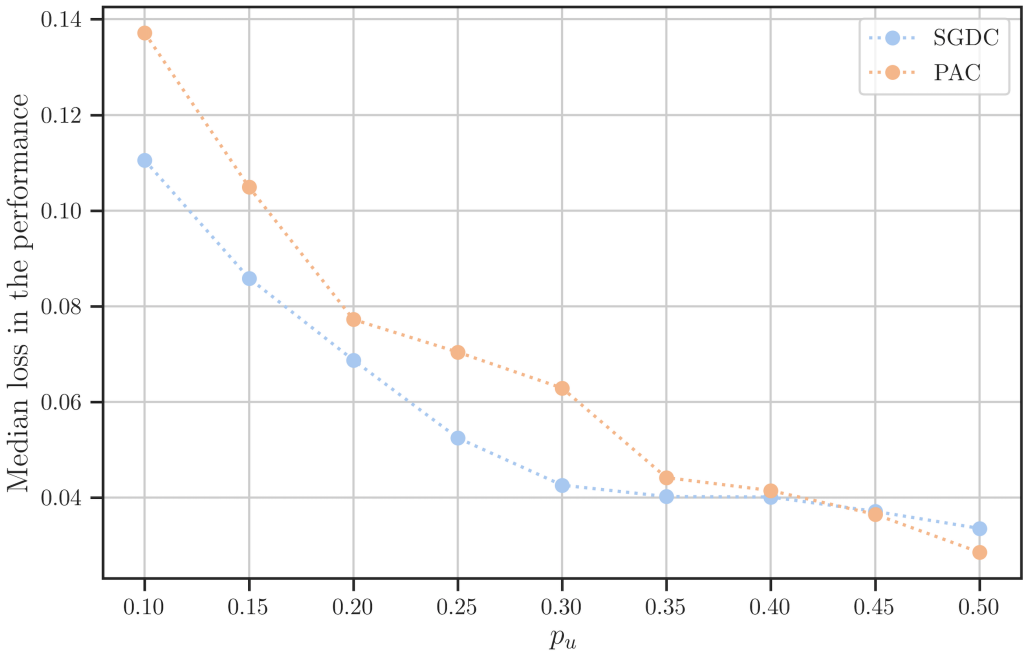

Figure 5.7: SGDC vs PAC when considering the median loss for different levels of p_u.

egy, i.e. what is the final median percentage of ACKs that are requested from each classifier for each p_u investigated. We have $p_{u,t=T} = 0.1309 + 0.8783 p_{u,t=0}$ and $p_{u,t=T} = 0.0419 + 1.0268 p_{u,t=0}$ for SGDC and PAC, respectively. Hence, for example if we initialize $p_u = 0.35$, we can expect that SGDC requires 43.83% of the total ACKs, while PAC the 40.13%, at the end of the test.

As anticipated from the analysis of the box-plots, in Figure 5.7 we see that PAC has an higher median loss in the performance, due to the fact that it is requesting a lower ratio of ACKs. Indeed, we highlight that for $p_u \in \{0.45, 0.50\}$ PAC has a lower loss, hence this classifier should be preferred with respect to SGDC, in particular if considering that it is able to achieve this performance when requesting a smaller number of ACKs. In conclusion, we report some confusion matrices to state that a MCC higher than 0.8 is associated to a classifier performing very well, e.g.

$$\Lambda_1 = \begin{bmatrix} 1185 & 12 \\ 14 & 162 \end{bmatrix}_{p_u=1.0,\,\mathrm{MCC}=0.91}, \qquad \Lambda_2 = \begin{bmatrix} 1183 & 14 \\ 23 & 153 \end{bmatrix}_{p_u=0.5,\,\mathrm{MCC}=0.88}, \tag{5.3}$$

$$\Lambda_3 = \begin{bmatrix} 1175 & 22 \\ 19 & 157 \end{bmatrix}_{p_u=0.35,\,\mathrm{MCC}=0.86}, \qquad \Lambda_4 = \begin{bmatrix} 1168 & 29 \\ 26 & 150 \end{bmatrix}_{p_u=0.2,\,\mathrm{MCC}=0.82}. \tag{5.4}$$

Notice that the accuracy associated to Λ_1, Λ_2, Λ_3, Λ_4 are respectively 0.9810, 0.9731, 0.9701 and 0.9599. Hence, a decrease of 2% in the accuracy is associated to a decrease of 9% in the MCC of our analysis, confirming that a fair metric is necessary when facing class imbalance in a binary classification problem, but in particular that a decrease of 9% in the performance is associated to a reduction of approximately 70% of the ACKs requested for a test.

Chapter 6

Conclusion

In this thesis we have shown how dual-bands sensors can exploit a particular category of online learning algorithms, i.e. those employing active learning strategies, to get an high reduction in the number of feedback packet requests and, consequently, reducing the traffic when considering multiple end-devices into a low-power wide-area network.

This task was achieved using a Bernoulli distribution and Bayesian confidence intervals to determine what is the best moment to request an acknowledgement message. In addition, also the Page-Hinkley statistical test derived from drift-detection methods has been considered in order to monitor the predictive performance. Then, a passive aggressive algorithm and a logistic regression model having a stochastic gradient descent derivation are built to decide whether the channel status is good enough to use it. To further increase the final performance of the algorithm, a simple threshold on the performance of the model has been adopted with the aim of testing its awareness in the prediction of new instances.

All these methods considered together yield to an algorithm characterized by simplicity, robustness and high predictive performances. We have coped with a challenge based on the trade-off between effectiveness and efficiency. The former induced by the necessity of keeping high performances while maintaining as small as possible rates of acknowledgments requests. The latter imposed from the sensor capacity to store a simple model in a small device, while providing fast predictions.

The past decades have been characterized by communications of two kinds, either human-human or human-device, but the new IoT technologies promise a great future for a machine-machine communication. With this work, we hope to contribute to the new revolution of the Internet, providing a new example of more and more intelligent devices.

We conclude by stating that this work is based on data derived from experiments that are still in progress, the investigation of the proposed methods is limited to the available datasets and is not completed. The algorithms must be validated on more datasets and multiple scenarios, studying whether the performances reported in this work are confirmed when employed into new tests.

Bibliography

[1] R. Asorey-Cacheda, A. J. Garcia-Sanchez, F. Garcia-Sanchez, J. Garcia Haro, and F. J. Gonzalez-Castano, "On maximizing the lifetime of wireless sensor networks by optimally assigning energy supplies", Sensors Journal, Basel, Switzerland, vol. 13, pp. 10219–10244, 2013.

[2] LoRa Alliance, "White Paper: A Technical Overview of Lora and Lorawan", The LoRa Alliance, San Ramon, USA, 2015.

[3] Wei Ye, J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, NY, USA, 2002, pp. 1567-1576 vol.3.

[4] G. Ditzler, M. Roveri, C. Alippi and R. Polikar, "Learning in Nonstationary Environments: A Survey," IEEE Computational Intelligence Magazine, vol. 10, no. 4, pp. 12-25, Nov. 2015, doi: 10.1109/MCI.2015.2471196.

[5] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," IEEE Trans. Neural Netw., vol. 22, no. 10, pp. 1517–1531, Oct. 2011.

[6] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers-part II: Designing the classifier," IEEE Trans. Neural Netw., vol. 19, no. 12, pp. 2053–2064, Dec. 2008.

[7] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. Advances Artificial Intelligence–SBIA*, 2004, pp. 286–295.

[8] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui and T. Watteyne, "Understanding the Limits of LoRaWAN," in *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34-40, Sept. 2017, doi: 10.1109/MCOM.2017.1600613.

[9] N. Sornin et al., "LoRa Specification 1.0," LoRa Alliance Std Spec., Jan. 2015; www.lora-alliance.org, accessed Dec. 19, 2016.

[10] C. Goursaud and J. M. Gorce, "Dedicated Networks for IoT: PHY/MAC State of the Art and Challenges," *EAI Endorsed Trans. Internet of Things*, vol. 15, no. 1, Oct. 2015.

[11] T. Watteyne, A. Mehta, and K. Pister, "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense," ACM Symp. *Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, Oct. 2009, pp. 116–23.

[12] M. Centenaro, L. Vangelista, A. Zanella, M. Zorzi, "Long-Range Com- munications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios," in *IEEE Wireless Communications*, vol. 23, no. 5, October 2016, pp. 60 - 67.

[13] Christopher M. Bishop. 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.

[14] Hastie, Trevor, Tibshirani, Robert and Friedman, Jerome. The Elements of Statistical Learning. New York, NY, USA: Springer New York Inc., 2001.

[15] Andrew Y. Ng and Michael I. Jordan, "On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes", *NIPS'01: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 2001.

[16] Vladimir N. Vapnik. The nature of statistical learning theory. Springer-Verlag, Berlin, Heidelberg. 1995.

[17] Brown, Lawrence D.; Cai, T. Tony; DasGupta, Anirban. "Interval Estimation for a Binomial Proportion". Statist. Sci. 16 (2001), no. 2, 101–133. doi:10.1214/ss/1009213286.

[18] Gelman, A., Carlin, J. B., Stern, H. S., Rubin, D. B. (2004). Bayesian Data Analysis. Chapman and Hall/CRC.

[19] Berger, James O.; Strawderman, William; Tang, Dejun. "Posterior propriety and admissibility of hyperpriors in normal hierarchical models". Ann. Statist. 33 (2005), no. 2.

[20] Larry Wasserman. 2006. All of Nonparametric Statistics (Springer Texts in Statistics). Springer-Verlag, Berlin, Heidelberg.

[21] D.V. Hinkley (1970). "Inference about the change point in a sequence of random variables". Biometrika, vol.57, no 1, pp.1-17.

[22] M. Basseville and Igor V. Nikiforov. 1993. Detection of abrupt changes: theory and application. Prentice-Hall, Inc., USA.

[23] M. Zinkevich, "Online Convex Programming and Generalized Infinitesimal Gradient Ascent", ICML'03, 2003.

[24] Tibshirani, Robert. "Regression Shrinkage and Selection via the Lasso." Journal of the Royal Statistical Society. Series B (Methodological) 58, no. 1 (1996): 267-88. Accessed September 21, 2020.

[25] Y. Tsuruoka and S. Ananiadou, "Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty", Association for Computational Linguistics, 2009, 477–485.

[26] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. "Online Passive-Aggressive Algorithms". J. Mach. Learn. Res. 7 (2006), 551–585.

[27] Boughorbel S, Jarray F, El-Anbari M (2017), "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric". PLOS ONE 12(6): e0177678.