AGNES TEGEN

**APPROACHES TO INTERACTIVE
ONLINE MACHINE LEARNING**

**APPROACHES TO INTERACTIVE ONLINE
MACHINE LEARNING**

Malmö University,
Studies in Computer Science No 10,
Licentiate Thesis

© Agnes Tegen, 2020

ISBN 978-91-7877-084-7 (print)

ISBN 978-91-7877-085-4 (pdf)

Holmbergs, Malmö 2020

AGNES TEGEN

**APPROACHES TO
INTERACTIVE ONLINE
MACHINE LEARNING**

Malmö University, 2020
Faculty of Technology and Society
Department of Computer Science and Media Technology

Studies in Computer Science

Faculty of Technology and Society
Malmö University

1. Jevinger, Åse. Toward intelligent goods: characteristics, architectures and applications, 2014, Doctoral dissertation.
2. Dahlskog, Steve. Patterns and procedural content generation in digital games: automatic level generation for digital games using game design patterns, 2016, Doctoral dissertation.
3. Fabijan, Aleksander. Developing the right features: the role and impact of customer and product data in software product development, 2016, Licentiate thesis.
4. Paraschakis, Dimitris. Algorithmic and ethical aspects of recommender systems in e-commerce, 2018, Licentiate thesis.
5. Hajinasab, Banafsheh. A Dynamic Approach to Multi Agent Based Simulation in Urban Transportation Planning, 2018, Doctoral dissertation.
6. Fabijan, Aleksander. Data-Driven Software Development at Large Scale, 2018, Doctoral dissertation.
7. Bugeja, Joseph. Smart Connected Homes: Concepts, Risks, and Challenges, 2018, Licentiate thesis.
8. Alkhabbas, Fahed. Towards Emergent Configurations in the Internet of Things, 2018, Licentiate thesis.
9. Paraschakis, Dimitris. Sociotechnical Aspects of Automated Recommendations: Algorithms, Ethics, and Evaluation, 2020, Doctoral dissertation.
10. Tegen, Agnes. Approaches to Interactive Online Machine Learning, 2020, Licentiate thesis.

Electronically available at:

<http://doi.org/10.24834/isbn.9789178770854>

To Jonatan

ABSTRACT

With the Internet of Things paradigm, the data generated by the rapidly increasing number of connected devices lead to new possibilities, such as using machine learning for activity recognition in smart environments. However, it also introduces several challenges. The sensors of different devices might be of different types, making the fusion of data non-trivial. Moreover, the devices are often mobile, resulting in that data from a particular sensor is not always available, i.e. there is a need to handle data from a dynamic set of sensors. From a machine learning perspective, the data from the sensors arrives in a streaming fashion, i.e., online learning, as compared to many learning problems where a static dataset is assumed. Machine learning is in many cases a good approach for classification problems, but the performance is often linked to the quality of the data. Having a good data set to train a model can be an issue in general, due to the often costly process of annotating the data. With dynamic and heterogeneous data, annotation can be even more problematic, because of the ever-changing environment. This means that there might not be any, or a very small amount of, annotated data to train the model on at the start of learning, often referred to as the cold start problem.

To be able to handle these issues, adaptive systems are needed. With adaptive we mean that the model is not static over time, but is updated if there for instance is a change in the environment. By including human-in-the-loop during the learning process, which we refer to as interactive machine learning, the input from users can be utilized to build the model. The type of input used is typically annotations of the data, i.e. user input in the form of correctly labelled data points. Generally, it is assumed that the user always provides correct labels in accordance with the chosen interactive learning strategy. In many real-world applications these assumptions are not realistic however, as users might provide incorrect labels or not provide labels at all in line with the chosen strategy.

In this thesis we explore which interactive learning strategies are possible in the given scenario and how they affect performance, as well as the effect of machine learning algorithms on performance. We also study how a user who is not always reliable, i.e. that does not always

provide a correct label when expected to, can affect performance. We propose a taxonomy of interactive online machine learning strategies and test how the different strategies affect performance through experiments on multiple datasets. The findings show that the overall best performing interactive learning strategy is one where the user provides labels when previous estimations have been incorrect, but that the best performing machine learning algorithm depends on the problem scenario. The experiments also show that a decreased reliability of the user leads to decreased performance, especially when there is a limited amount of labelled data.

Keywords: Interactive Machine Learning, Interactive Learning, Online Learning

PUBLICATIONS

Included Papers

- Paper I. Tegen, A., Mihailescu, R.C., Davidsson, P., Persson, J.A.: Collaborative Sensing using Dynamic Intelligent Virtual Sensors. In: Sensors, 2019, 19, 477, DOI: <https://doi.org/10.3390/s19030477>
- Paper II. Tegen, A., Davidsson, P., Persson, J.A.: Activity Recognition through Interactive Machine Learning in a Dynamic Sensor Setting. Accepted for publication in Personal and Ubiquitous Computing, DOI: <http://dx.doi.org/10.1007/s00779-020-01414-2> Extended version of publication b).
- Paper III. Tegen, A., Davidsson, P., Persson, J.A.: A Taxonomy of Interactive Online Machine Learning Strategies. Submitted to Conference. Extended version of publication c).
- Paper IV. Tegen, A., Davidsson, P., Persson, J.A.: The Effects of Reluctant and Fallible Users in Interactive Online Machine Learning. Submitted to Conference.

Personal Contribution

For all publications above, except Paper I, the first author was the main contributor with regard to the planning and execution of the research as well as the writing of the publications. Regarding Paper I, the first author was the main contributor of data curation, formal analysis, investigation and software development. For the writing of paper I, all four authors were equal contributors.

Subsidiary Publications

- a) Tegen, A., Davidsson, P., Persson, J.A.: An Interactive Learning Scenario for Real-time Environmental State Estimation Based on Heterogeneous and Dynamic Sensor Systems. In: Proceedings of the Workshop on Interactive Adaptive Learning co-located with European Conference on Machine Learning (ECML 2018) and Principles and Practice of Knowledge Discovery in Databases (PKDD 2018), 2018, 77-79
- b) Tegen, A., Davidsson, P., Persson, J.A.: Interactive Machine Learning for the Internet of Things: A Case Study on Activity Detection. In: Proceedings of the 9th International Conference on the Internet of Things, 2019, 10, 1-8, DOI: <https://doi.org/10.1145/3365871.3365881>.
- c) Tegen, A., Davidsson, P., Persson, J.A.: Towards a taxonomy of interactive continual and multimodal learning for the internet of things. In: Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers, 2019, 524-528, DOI: <https://doi.org/10.1145/3341162.3345603>

ACKNOWLEDGEMENTS

First of all, I would like express my sincere gratitude to my supervisors Paul Davidsson and Jan Persson. I am lucky to have you as my supervisors and I am very thankful for your guidance and support. Thank you for sharing your knowledge and expertise and for always taking the time when I ask for help.

I am grateful to the Knowledge Foundation and Malmö University in collaboration with several industrial partners for financing the “Internet of Things and People (IOTAP)” Research Profile. I also want to thank the members of the research project “Dynamic Intelligent Sensor Intensive Systems (DISS)”.

I would like to thank my examiner Thomas Pederson and follow up team Carl-Magnus Olsson and Romina Spalazzese for aiding me in improving my study plan and my research. Many thanks to my fellow PhD students at Malmö University for always helping out and for interesting discussions during well-needed coffee breaks. Furthermore, I would like to thank other colleagues at Malmö University who have helped me during my PhD studies. I would like to give an especially big thanks to the following people. Åse Jevinger, for your work as director of PhD studies and for being a person I know I can always turn to for advice. Johan Holmberg, for always helping out in the lab and with other technical matters as well as giving great podcast tips. Annabella Loconsole, for bringing positivity and empathy to people around you. Solveig-Karin Erdal, for always being helpful and for multiple times patiently explaining the workings of Retendo to me. Susanne Lundborg, for your help with the logistics of the licentiate seminar as well as other parts of my PhD studies.

Many thanks to my family and friends who are supporting me during my PhD journey. Special thanks to Hilda Lidén for always being a supportive and understanding friend, even during difficult times.

Finally, I am eternally grateful to my husband Jonatan Tegen. Your kindness, encouragement and love are unequalled. I would not have got far without you.

TABLE OF CONTENTS

PART I. COMPREHENSIVE SUMMARY	1
1. INTRODUCTION	3
2. BACKGROUND.....	5
2.1. Machine Learning.....	5
2.2. Interactive Machine Learning	7
2.3. Problem setting.....	8
3. RESEARCH QUESTIONS	11
4. METHODOLOGY.....	13
4.1. Literature review	13
4.2. Design Science Research.....	13
4.3. Simulation experiments.....	15
4.4. Evaluation methods and metrics.....	15
5. CONTRIBUTIONS	19
5.1. Research Question 1	19
5.2. Research Question 2.....	21
5.3. Research Question 3.....	22
5.4. Research Question 4.....	22
5.5. Limitations	23
6. CONCLUSIONS AND FUTURE WORK.....	27
6.1. Conclusion	27
6.2. Future Work.....	28

PART II. PAPERS 35

PAPER I: COLLABORATIVE SENSING USING DYNAMIC INTELLIGENT VIRTUAL SENSORS.....	37
1. Introduction	38
2. Materials and Methods	40
3. Results.....	45
4. Discussion	56
5. Conclusions	57
 PAPER II: ACTIVITY RECOGNITION THROUGH INTERACTIVE MACHINE LEARNING IN A DYNAMIC SENSOR SETTING	63
1. Introduction	64
2. Related work.....	65
3. Experimental setup.....	67
4. Results.....	74
5. Discussion	80
6. Conclusion and future work	83
 PAPER III: A TAXONOMY OF INTERACTIVE ONLINE MACHINE LEARNING STRATEGIES	89
1. Introduction	90
2. Related work.....	90
3. Interactive online machine learning strategies	92
4. Experimental setup.....	99
5. Results and Discussion.....	101
6. Conclusion and Future Work.....	105
 PAPER IV: THE EFFECTS OF RELUCTANT AND FALLIBLE USERS IN INTERACTIVE ONLINE MACHINE LEARNING.....	111
1. Introduction	112
2. Related work.....	113
3. Experimental setup.....	114
4. Results.....	120
5. Discussion	121
6. Conclusion and Future Work.....	127

Part I.

**COMPREHENSIVE
SUMMARY**

1. Introduction

The name Internet of things (IoT) was coined by Ashton in 1999 [1]. It is a collective term for devices with built in electronics and the possibility of transmission of information through a network. These devices are becoming increasingly more present in our surroundings, mainly because of the expansion of connectivity and that the technology is getting cheaper. IoT can now be found in a multitude of application areas, such as healthcare, transportation and energy management [2].

Activity recognition and smart buildings are other application areas where IoT has created new possibilities. In activity recognition, the aim is to correctly classify the ongoing activity in real-time. The aim of smart buildings is to regulate aspects of a particular environment, for instance heating or access. The first step of regulating these aspects is to classify the current state of the environment. The classification is often done using data from sensors that can either be worn by humans or are placed in the environment. Instead of manually programming an algorithm that do the classification based on the sensor data, which could be very difficult and time-consuming, machine learning approaches are frequently used. In contrast, machine learning algorithms can use previously obtained data to train a classifier. Activity recognition can for instance be used in elder care for fall detection [3]. Techniques developed for smart buildings can for example be used to optimize energy consumption in larger buildings such as an office building or a university [4]. These types of buildings typically have a large flow of people coming and going. By estimating the current number of people present, as well as where they are located, attributes such as lighting or heating can be adjusted to suit the setting.

These possibilities also comes with challenges however. The sensors are streaming data, which means that one data instance at a time is presented to the machine learning algorithm and the order of the data instances can not be changed. This can be compared to machine learning problems in pool-based settings, where data instances might have timestamps, but they do not restrict the way in which the data is presented to the machine learning algorithm. Furthermore, unless data has been previously collected for the particular problem setting at hand, there

will not be any data to train the model. Because the machine learning algorithm starts out without any data to learn from, the data needs to be collected gradually over time and learn to classify better and better. This way the model can be incrementally trained over time. To learn from sensor data, the model does not only need data streamed from the sensors, but also a label telling the correct class of a data instance, i.e. the current state of the environment. For example, if the goal is to classify whether a room is occupied or empty, we do not only need a data instance containing current sensor values collected from devices in the room, but also whether the data instance corresponds to the room being occupied or not. To obtain information of which class label is correct for the given data instance, input from human users can be utilized. These users need to know the current state of the environment or the current ongoing activity to be able to provide labels. Since activities are performed by humans and the state typically is with regards to humans, they can be asked to provide labels. Human users might not always provide labels when expected to however. Furthermore, the labels they do provide might not be correct. These factors of reliability will affect the performance of the classification and need to be considered when constructing the system.

Another challenge concerns the access to the streams of sensor data from the devices. The devices that are streaming data can be mobile, for instance the previously mentioned wearable sensors or smart phones that tend to move with their owner. This means that their data streams might not always be available, as they can enter and leave an environment. Even if they are not mobile they can suddenly stop streaming because of network problems, that the device breaks down or is out of battery power. Moreover, new devices that are streaming data might appear, for instance new mobile devices entering the environment or installation of a new device in the environment. Any machine learning algorithm used for this task must thus be able to handle the dynamic sensor setting.

In this thesis, the design of systems learning to classify from data generated by a dynamic set sensors is explored. By using input from users in the form of labels together with the sensor data, the system can be trained incrementally over time. We investigate which different strategies are possible for providing labels and how they can affect the performance of the classification. How the choice of machine learning algorithm, as well as how the reliability of the user providing labels affects performance are also explored.

2. Background

The following chapter presents concepts used throughout the rest of the thesis. The problem setting that is the starting point for the thesis is explained as well as particular aspects that has to be taken into consideration when developing possible solutions.

2.1. Machine Learning

Machine learning entails algorithms that can evolve through self-learning mechanisms, in contrast to algorithms that are manually crafted. Supervised machine learning algorithms, which is the type of machine learning that we focus on in this thesis, construct a classifier for a given problem by learning from labelled data. Before any data is presented, the mathematical setup of the classifier and possible parameters are defined. It learns what to classify and how to do it through the data processed, referred to as training data.

The three different machine learning algorithms employed in the work of this thesis are presented below. The aim of the research was not to develop new machine learning algorithms, therefore well-known off-the-shelf machine learning algorithms were selected. The three algorithms were chosen because of their qualities appropriate for the given problem setting. While algorithms such as Neural Networks are popular and produces good results in many scenarios, they were not included for our experiments. The reason being that Neural Networks typically need to be trained on a significant amount of data before they become high performing, which is not ideal in our problem setting.

2.1.1. Naïve Bayes classifier

The Naïve Bayes classifier is a probabilistic classifier based on Bayes' theorem and with assumptions of independence between the features. Through Bayes' theorem, the probability of a certain data instance belonging to a given class can be calculated based on prior knowledge of the data distribution av classes. In a multi-class problem, this calculation can be made for each class. The probabilities can then be compared so that the class with the highest probability can be found. This is the basis

of the Naïve Bayes classifier and can be mathematically expressed as,

$$\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

where $\hat{y}$ is the estimated class, $p(C_k)$ is the probability of class C_k and $p(x_i | C_k)$ is the probability that data instance x_i belongs to class C_k . Essentially the model classifies a data instance as the class which is most probable. Even though the classifier assumes independence among the used features, it can still perform at a high level if the assumption is not fulfilled [5].

The Naïve Bayes classifier is a suitable choice for an online learning problem. The classifier needs fewer training examples compared to many other machine learning methods, which also is useful in a cold start scenario. Furthermore, it is not computationally heavy, which makes it suitable for real-time classification [6].

2.1.2. Support Vector Machines

Support vector machines builds a classifier model based on training set of data. The model maps the samples to a space such that the distance between classes is as large as possible. The division is made by constructing hyperplanes with optimal placement for separating the classes. When a new sample is to be classified, it is mapped to the space of the SVM model and its placement with regards to the hyperplanes determines the classification [7].

Support Vector Machines are also efficient regarding memory usage, which is useful for high-dimensional datasets [8]. In a dynamic sensor setting, the number of features increase as the number of sensors streaming data increase. A machine learning algorithm that can handle a possibly large number of features is thus useful.

2.1.3. k-Nearest Neighbor

In k-nearest neighbor, the classifier uses the labels of the most similar samples to decide the class for a new data instance. The number of samples looked at is decided by the value of k , which is to be set before classification starts. The smaller the value of k , the more local is the focus of the algorithm [9].

The algorithm in itself is very simple and easily updated with new incoming labelled data which makes it a good choice for online learning [10]. The computational complexity of k-nearest neighbor is dependent on how many labelled samples are stored, since a distance has to be calculated for each when a new data instance arrives. Theoretically this could lead to problems with streaming data, because the amount of

accumulated data is constantly increasing. In reality however, an upper limit is typically set for how many labelled data instances are allowed to be stored. By choosing this limit wisely, the computational complexity can be kept low while still providing enough labelled data for the model to perform at a satisfying level. Moreover, the classifier only needs k data instances to start classifications, which is useful in a cold start scenario.

2.2. Interactive Machine Learning

Interactive Machine Learning is the term used throughout the thesis for including human-in-the-loop in the machine learning process. When the data arrives in a single-pass streaming fashion, it becomes interactive online machine learning. Most previous work in the research area falls either under active learning or machine teaching, which are both described below. The main motivation behind both is the fact that typically is costly to label data. Because of this, it is useful if only a selection of all data has to be labelled, without compromising with accuracy. The size of the selection is called *labelling budget* and is often specified in terms of a percentage of the total amount of data. Given a labelling budget, the aim is to maximise information gain for the machine learning model by choosing which data instances for labelling wisely.

2.2.1. Active Learning

In active learning, a learning algorithm queries a so called *oracle* to provide labels for the chosen data instances [11], [12]. The oracle can be another system or a human where the human can be everything between a domain expert to a regular user with a possibility to interact with the system. The active learning strategy has criteria for which data instances should be selected for queries. One of the more frequently used active learning strategies is *Uncertainty sampling*. In this strategy the criteria for query is that the machine learning model is uncertain regarding its own estimation.

The active learning process works differently depending on if it is a pool-based or stream-based setting. In the pool-based setting, the sampling strategy has access to all data at the same time and can select instances for labelling according to the criteria until the labelling budget is used up. In the stream-based setting with online learning however, the data arrives in a single-pass manner. The sampling strategy must for each new incoming instance decide if it should query for the label or not. This means the only opportunity to obtain a label for a particular data instance is when it has just arrived and is processed by the algorithm. For stream-based sampling the labelling budget must also be handled differently, since the total amount of data is not known at the start of

streaming and can, at least in theory, be infinite [13]. While there exists different approaches that tackle this, one of the more popular techniques uses a sliding window. To balance the labelling budget over time, temporary labelling expenses are calculated over a window of recent time. The labelling budget here becomes the upper limit for the window instead, since it is not reasonable to use all data in the calculation.

The general assumption in active learning of the oracle, whether another system, an expert or regular user, is that a correct label will always be provided when a query is posed. In many real-world scenarios these are not always realistic assumptions, but the issue is still rarely discussed in the literature. There does however exist previous work that explores the effect of relaxing the assumptions [14], [15].

2.2.2. Machine Teaching

In machine teaching it is the human user that takes the initiative of which data instances should be labelled, compared to active learning where an algorithm poses queries [16], [17]. Which data instances are selected to be labelled depends on the learning scenario, on the user and on the machine teaching strategy selected. If the user is an expert in the relevant domain of the labelling and given the explicit task of providing labels the pattern of provided labels might look different compared to a random user that happens to be in a situation where they have the possibility to provide labels.

For machine teaching, the same adaption of labelling budget needs to be employed when moving from pool-based sampling to stream-based sampling. Also worth noticing here is that there does not exist much previous research on machine teaching in a stream-based setting.

2.3. Problem setting

Figure 2.1 displays an illustration of the problem setting together with the different parts of the learning system. The *IoT environment* in the figure is the room, building or defined physical space where we want to detect the current activity or state by using data streamed from devices. The streamed data consist of a sequence of individual data instances. Each instance contains the latest value of the currently streaming sensors and a timestamp. The streamed data is used by the machine learning algorithm to produce classifications in real-time.

The user in the figure has access to the current activity or state and can share this information with the machine learning algorithm by providing labels. They can either be partaking in the activity to be classified or only be observing it. When the user provides a label, it can be used together with the sensor values collected at the same time to incrementally train the machine learning algorithm. The arrows pointing towards

Figure 2.1.
An illustration of the problem setting including the interactive online machine learning system. The dashed arrows indicate optional interactions.

the user in the figure are all examples of events that could trigger the user to provide labels.

The separate components mentioned of the problem setting that has to be taken into account when developing a solution, are each described in their own subsection below.

2.3.1. Streaming Data

In learning scenarios with streaming data the data arrives sequentially in real-time. Often the data is arriving in a single-pass fashion, which means that it is only possible to process each data instance once, when it arrives. This is in contrast to pool-based learning scenarios, where all data is available from the start and even though the individual data instances can have timestamps, it does not affect how the data is processed. The sequential order and single-pass fashion of the streamed data puts extra restrictions on the machine learning algorithm. Most machine learning problems deal with a pool-based setting where the data for instance can be divided into a training set and testing set without this restriction. Streaming data also means a continuous flow of information, i.e. it is generally not known beforehand how much data will be generated in a stream-based setting. Any machine learning algorithm storing data must thus have a strategy for how to deal with a possible infinite

stream of data.

In stream-based settings, it is often needed to produce estimations in real-time. As new data arrives, it is of interest to produce an estimate without delay. Furthermore, any setting with streaming data have the possibility of *concept drift* [18]. Concept drift means that the statistical properties of the streamed data changes over time. For instance, assume the room from the example of occupancy detection is located in the north of Sweden and that there is a sensor detecting the light level. In winter, when there is less natural light, a certain light level might indicate that the room is occupied, because the lights are turned on. In summer however, the same light level could be the natural light and does not necessarily mean that the room is occupied. In this example there has been a concept drift with regards to the light level and occupancy of the room, as the seasons go from winter to summer.

2.3.2. Dynamic set of sensors

A dynamic sensor setting entails that the set of sensors streaming data is not constant over time. In sensor-intensive environments, there are several reasons why sensors might stop or start streaming data. A mobile device can enter or leave an environment. For instance, smartphones or wearable sensors typically follow the movements of a human. Sensors can also stop streaming because they for example break down or there are network problems. New devices can also be installed and will then appear suddenly as a new source of streaming data. To restart and build a unique classifier every time the setup of sensors changes is not feasible. Another option would be a learning model that can adapt to a changing sensor setting. This model would be able to continue to produce estimations, even though sensors stop streaming and be able to utilize information from new sensors appearing.

2.3.3. Cold start problem

The cold start problem mainly has been studied within the area of recommender systems but is present in many real world machine learning applications [19]. In essence, the cold start problem means that there is not enough information available to train the model before it will be used for estimations. In our problem setting, it refers to no labelled data being available when the machine learning model is put into use or, expressed differently, at the start of the streaming data session. To provide labelled data, interactive learning is used (described further in section 2.2) and the model is incrementally trained as more data gradually becomes available. Because of the cold start scenario, the learning model must be able to learn from a limited amount of labelled data and be able to be incrementally trained over time.

3. Research Questions

In the problem setting investigated in this thesis, a dynamic set of heterogeneous sensors present in a given environment is streaming data. The task is to classify the currently ongoing activity or state of the environment by using this data. As this task is highly dependent on what sensors are available, as well as on the properties of the environment and the persons present, it is very difficult to program such a classifier. Instead, we will use machine learning methods to generate the classifier. Moreover, we assume that there is no previous data collected and labelled, thus we are facing a cold-start problem. Even though there is no previously labelled data, a user has the possibility to provide labels in accordance with an interactive online learning strategy. Given the problem setting, the following research questions were explored.

RQ1. Which are the possible interactive online machine learning strategies?

The core of an interactive learning strategy is that a human user provides labels in accordance with a well-defined procedure. What differs the strategies is when labels are provided, or in other words what could trigger a user to provide a label. While there exists previous work that for instance compares different active learning strategies, our aim is to study possible events in the given scenario that could affect the user to provide or withhold labels.

RQ2. How does the choice of interactive learning strategy affect the accuracy of the classification over time?

While the aim of RQ1 is to identify possible events that could trigger a user to provide labels, it does not evaluate the performance of the strategies. The evaluation is instead the focus of this research question. Because of the cold start scenario, measuring accuracy over time becomes especially interesting. It is both of interest to study performance with a limited amount of labelled data i.e. early in the data stream, and when larger amounts of data has been gathered and used for training.

RQ3. *How does the choice of machine learning algorithm affect the accuracy of the classification over time?*

The main focus of this thesis is the interactive learning aspect, but since machine learning algorithms may have very different performance in this context, it is important to study accuracy. The aim is not to develop new algorithms, but rather to use off-the-shelf machine learning algorithms that are suitable for the given scenario. It is an important aspect to study not only because it might influence performance in itself, but also because the combination of an algorithm and different interactive learning strategies can affect the performance.

RQ4. *How does the reliability of users affect the accuracy of classification over time?*

In previous work on interactive learning it is typically assumed that a user is completely reliable. The term *oracle* used within active learning embodies this perspective. While there do exist scenarios where it is reasonable to assume complete reliability from the user, it is unrealistic in many real-world settings. Findings from experiments where these assumptions were made might not hold if the user is not behaving like an oracle. The aim here is to find out both how the reliability of users affect the accuracy by itself, but also in combination with interactive learning strategies and machine learning methods. Are certain strategies or methods more robust with regards to the reliability of the user?

4. Methodology

The following chapter explains how the research questions were addressed with regards to methodologies used, how evaluation was performed as well as limitations of the research done.

4.1. Literature review

Literature review has been used as a secondary methodology in this thesis. To motivate the research, literature review was used to generate an overview of related work but also to find gaps in the relevant research fields. In RQ1 it was used to explore interactive online machine learning strategies used in previous research. For RQ3, it was used to study what machine learning algorithms have been successful for similar problem settings. This gave a selection of suitable choices based on previous work, instead of testing all possible algorithms.

Database searches were used in combination with snowballing, to obtain a better coverage of the literature, as suggested by Wohlin [20]. Both backward snowballing, i.e. looking the references of a relevant publication to find other possible relevant sources, and forward snowballing, i.e. looking at new papers citing a relevant publication to find other possible relevant sources, were employed. Google Scholar was used because of its coverage and to avoid bias from any particular publisher.

4.2. Design Science Research

The starting point for this research was a problem setting where the goal was to find theoretical models as well as practical solutions. Design Science was considered a suitable methodology, as it is a problem-solving paradigm essentially, where the methodological process centres around the creation and application of an artifact [21]. Hevner et al. discusses four types of artifacts, constructs, models, methods and instantiations. The creation of a taxonomy, i.e. a theoretical model, falls under the model category, as it is a way to represent a real world situation including both the problem and the solution space. The implementation and testing of different interactive machine learning strategies belongs to the

instantiations category, as it provides a way to concretely assess the suitability of the strategies for their intended use [21].

Design Science Research was used to address RQ1 and partially RQ2 and RQ3 as well as constructing an environment where RQ2, RQ3 and RQ4 could be further studied through simulation experiments. In this methodology, the research process is typically divided into several steps. Below, the different steps of the Design Science Research Methodology, as proposed by Peffers et al. [22], are used to briefly describe the process in this thesis.

Activity 1: Problem identification and motivation. In this first step, the limitations and requirements of the problem setting were defined, see section 2.3 for more details. Given this setting, the aim was to produce estimations of the current activity or state of the environment through machine learning by using sensor data and labels provided by a user as input.

Activity 2: Define the objectives for a solution. The objectives were to both create a theoretical model of possible interactive online learning strategies and be able to test the performance of the strategies.

Activity 3: Design and development. This is the step where the artifact is created. In the case of the theoretical model of possible interactive online learning strategies, a taxonomy was created. For the testing of the strategies, versions of the strategies were implemented using Python^a along with other necessary scripts that could simulate the described problem setting.

Activity 4: Demonstration. Demonstration can be seen as the initial phase of evaluation where the artifact is analyzed with regards to the problem. In this stage, the taxonomy was studied in the relevant context, i.e. the problem setting, and datasets (described further in a later section) were used to test the implementations.

Activity 5: Evaluation. Here the goal is to determine how well the proposed artifact provides a solution to the problem. As previously mentioned, it is connected to the previous step, Demonstration. For the taxonomy, we have through the experiments shown that all strategies can be implemented, i.e. they are true potential strategies. We also applied the taxonomy to the problem setting and compared strategies presented in related work to the strategies of the taxonomy. To evaluate the implementation, the different strategies were compared to each other and with

---

^aPython Software Foundation. Available at <http://www.python.org>

regards to various changing factors in a simulated environment. These experiments are described further in the next section. Steps 3-5 were repeated several times to further improve the artifact.

Activity 6. Communication. Finally, the artifact and the knowledge gathered is communicated in a fashion suitable for the given disciplinary culture. The findings of our research were communicated by presentations at scientific conferences as well as workshops with practitioners, and publishing in renowned scientific journals and conference proceedings within the relevant research field.

4.3. Simulation experiments

The research questions RQ2, RQ3 and RQ4 focus on what effect a certain factor has on the accuracy of classification over time. To address these types of questions, experiments is often a suitable methodology. The basis of experiments is to examine cause and effect by studying how a dependent variable is affected by changing an independent variable. In the three research questions mentioned, the dependent variable is accuracy of classification over time. The independent variable is interactive learning strategy for RQ2, machine learning method for RQ3 and reliability of users for RQ4.

The experiments carried out were of the type simulation experiments [23]. Previously collected data or synthetically generated data were used to simulate scenarios where the experiments could be performed. There are advantages and drawbacks to using simulation experiments compared to experiments with human subjects. When the behavior of human subjects is modeled, the simulation will be a simplification of the real-world scenario. The aim is to include all relevant factors in the simulation. Still, it is important to acknowledge the limitations of a simulation experiment. On the other hand, simulation experiments increases the possibility for control with regards to the experimental setup. This means that it is easier to isolate the effect of the specific independent variable that is to be studied. The datasets and evaluation metrics used in the simulation experiments are described further in separate sections below.

4.4. Evaluation methods and metrics

This section discusses how the experiments were evaluated including the datasets used, simulations performed, and how performance was measured.

4.4.1. Datasets

Four different datasets were used in total for evaluation through experiments: the Occupancy dataset [24], the Opportunity dataset [25], the mHealth dataset [26], [27] and a synthetic dataset. All datasets were used to explore RQ2 and RQ3. In the study of RQ4, the mHealth dataset and the synthetic dataset were used for the experiments.

All of the datasets consist of recordings of human activities or the state of a specified environment using different types of sensors except for the synthetic dataset, which is synthetically constructed. The Occupancy dataset consists of three separate sensor recordings of a room in an office setting along with annotations whether the room is occupied by humans or empty. The recordings vary between 2664 instances (1.8 days) and 9752 instances (6.8 days). The sensors used measure light, temperature, humidity, CO2-level and humidity ratio. In the Opportunity dataset, human subjects that are performing different daily living activities are recorded with both wearable sensors placed on their body and sensors placed throughout the room where they perform the activities. The recordings are between 22230 instances (12.35 minutes) and 51116 instances (28.4 minutes). The data is annotated at multiple abstraction levels with regards to the activity performed. The annotations at the highest level of abstraction were used for the experiments and contained four different types of activities ("Stand", "Walk", "Sit" or "Lie") as well as a null category representing unlabelled data.

The mHealth dataset is recordings of human subjects performing a program of physical exercises. The activities are captured by wearable sensors placed on the human subjects' body. In total the program contain 12 different physical activities. Each recording contains 98304-161280 instances, including unlabelled data. The unlabelled data was excluded in the experiments which resulted in recordings with a length between 32205 and 35532 instances.

The benchmark datasets described were chosen because of their suitability for the problem setting and that they varied in characteristics between themselves. Moreover, they have all been used in multiple other studies. While datasets containing real recordings gives an indication of how a particular strategy or algorithm would perform in a real-world setting, real data does also contain noise and irregularities. To complement the benchmark datasets with a cleaner option a dataset was also synthetically created. The synthetic dataset was generated by first randomly generating five pairs of values. Each pair of values became the mean values of a class in a two dimensional space, i.e. five classes in total. For each class and mean values, 10000 new samples was generated, drawn from a Gaussian distribution. Since the synthetic dataset, unlike the other three datasets, did not have an inherent time feature, this was also randomly generated for all the samples of the dataset.

4.4.2. Simulation of dynamic sensor setting

To simulate a dynamic sensor setting, experiments were performed where the set of sensors streaming data was not constant over time. Since we did not have datasets with a dynamic sensor setting, the Occupancy dataset and the Opportunity dataset were used and the dynamic sensor setting was simulated. Before each run of the experiment, a setup for the dynamic sensor setting was randomly generated. A maximum and a minimum number of dynamic sensors was manually set for each experiment and dataset. The values ensures that the run contains at least the minimum of dynamic sensors, but at the same time always has at least one sensor actively streaming at all times. After the number of dynamic sensors has been generated, it is randomly chosen which sensors this should be and during which time periods they should stop streaming.

4.4.3. Test-then-train evaluation

Because of the streaming data and cold start scenario most evaluation methods frequently used in batch learning, for instance cross-validation, are not suitable options. Furthermore, the research questions state that it is of interest to study the accuracy over time. This means that we want a metric that continuously over time can assess the performance, from the very first estimation to the final one.

The main evaluation method chosen for the experiments was test-then-train [18]. In this method, the classifier first produces a classification for a new incoming sample, i.e. test. After this the interactive learning strategy decides whether a label should be provided for the data instance. If a label is provided, the labelled data instance is used to update the algorithm, i.e. train. This method gives an updated value of the accuracy for each new incoming sample, which is relevant since the algorithm is evolving. Over time this produces an accumulated accuracy that showcases the average performance of the model over time.

5. Contributions

The following chapter contains summaries of the contributions made in the included papers with regards to the previously presented research questions.

5.1. Research Question 1

The first research question uses the problem setting as a starting point and asks what events might trigger a user to provide labels.

RQ1. Which are the possible interactive online machine learning strategies?

In Paper III, a taxonomy of interactive online machine learning strategies is presented. The taxonomy was created with regards to the given problem setting and contains a set of different strategies, each defined by what triggers the user to provide labels. The identification of the different strategies was derived from studying interactive learning strategies described in the literature but also from studying the problem setting and connecting it to possible factors that could trigger the user to provide labels. Figure 5.1 showcases the taxonomy.

The strategies of the taxonomy are divided into either active learning (AL), where the user is queried, or machine teaching (MT), where the user provides labels from their own initiative. In total seven differ-

Figure 5.1.
Taxonomy of the interactive online machine learning strategies.

ent strategies were presented, where three belong to AL and four belong to MT. All strategies have to comply with the labelling budget, i.e. a label can only be provided if the labelling expenses allows it. In *AL triggered by uncertainty*, the user is queried if the estimation of the machine learning is considered uncertain. In this strategy, an uncertainty metric has to be defined which is dependent on the machine learning method employed. A threshold is specified using the uncertainty metric and creates the boundary for which estimations are considered uncertain. The threshold can be static or vary depending on how often a query is posed.

In the strategy *AL triggered by time* the user is queried with a given frequency, regardless of other factors. The frequency is calculated based on the labelling budget. *AL triggered at random* is similar in the sense that it is primarily calculated based on the labelling budget. Compared to the previous strategy however, the queries are not distributed evenly over time, but scattered at random. Both *AL triggered by time* and *AL triggered at random* can be seen as baseline strategies, as they use the labelling budget without taking any other factors into consideration.

In *MT triggered by error*, the user provides a label when the current estimation from the machine learning method is incorrect, given that the labelling budget allows it. For *MT triggered by state change* it is a change of the activity or the state that of the environment prompts the user to provide a label. If the labelling expenses are still lower than the allowed labelling budget after a label has been provided, we can be certain that the user will provide another label if the state changes again. This means it can be assumed that the last provided label is still accurate, as long as the labelling expenses allows for at least one more label being provided. Because of this, the algorithm can continue to collect data samples with the latest label, without involving the user.

The strategy *MT triggered by time* is similar to *AL triggered by time* in that a label is provided with a certain frequency which in turn is based on the labelling budget. However, here it is the user that initiates the label being provided and keeping track of time themselves. Finally, in *MT triggered by user factors*, the user is providing labels when they are triggered by a factor related to themselves. This could for instance be the stress level of the user, i.e. they have a higher probability of providing labels when they are less stressed.

Apart from the different categories listed in the taxonomy, hybrid versions of them can also exist, i.e. combinations of two or more strategies. In a real-world scenario with human users, hybrid versions might be the most frequently occurring, and not necessarily by design. For example, a user might employ the strategy *MT triggered by error*, but they might also be affected by their stress levels, i.e. *MT triggered by user factors*. Hybrid version can also be designed with a goal in mind. For instance,

MT triggered by state change might be used as the primary strategy, but if it does not lead to enough labelled data, it could be combined with *AL triggered by time*. The AL strategy would in this case query in enough time has passed without new labelled data being collected.

5.2. Research Question 2

The second research question addresses how the different strategies discovered through RQ1 performs compared to each other.

RQ2. *How does the choice of interactive learning strategy affect the accuracy of the classification over time?*

From the results throughout all experiments it is clear that the choice of interactive learning strategy has an effect on accuracy of the classification. This is often most apparent when only a limited amount of labelled data has been collected, i.e. early in the data streaming process. The overall best performing strategies are *MT triggered by error* (paper III and paper IV) and *MT triggered by error* in combination with *AL triggered by uncertainty* (paper I and paper II). It is also clear from the results in paper I-III that the size of the labelling budget has an affect on performance, where an increased budget leads to increased accuracy. Apart from the size of the labelling budget, it is also of significance which data instances are labelled, results from paper II showcase. Neither *MT triggered by error* nor *MT triggered by state change* uses up the allowed budget in the experiments of paper II. Still, *MT triggered by error* has the highest performance of the strategies, while *MT triggered by state change* has the lowest in this experiment.

It should be noted that there is a difference in the implementation of *MT triggered by state change* between paper II and paper III. In the latter, the implementation follows the description that can be found in the discussion of contributions on RQ1 (see section 5.1). In paper II, the implementation has the same trigger but the strategy does not assume that the label is the same for the new instances until the user labels a new instance. In this version the strategy therefore does not continue to collect labelled samples without the involvement of the user. The result is that the implementation of paper II leads to less labelled data for the model to train on. While the additional labelled data instances leads to increased performance, it is built on the assumption that the user will always deliver a correct label in accordance with the interactive learning strategy. This might not be the case in a real-world scenario however and is further discussed in section 5.4 below.

5.3. Research Question 3

The third research question focuses on the machine learning methods and if the choice will affect the performance.

RQ3. How does the choice of machine learning method affect the accuracy of the classification over time?

In paper II and paper III, the results show that the choice of machine learning method does affect performance but generally not to the same extent as the choice of interactive learning strategy. The results also show that there is not one machine learning method that always performs best, but it depends on the experiment. Between the two papers, experiments are run on four different datasets. In paper II, where the Occupancy dataset and the Opportunity dataset are used in the experiments, SVM gives a higher accuracy. In paper III on the other hand, the Naïve Bayes classifier gives the highest accuracy for the experiments on the mHealth dataset. For the experiments on the synthetic dataset, the different machine learning methods tested produces result at similar levels.

The conclusion that there is not one best performing machine learning method suitable for all scenarios explored is in line with the famous “No free lunch” theorem by Wolpert [28]. A consequence of the theorem is that there does not exist one machine learning algorithm that is the best choice for all problems.

5.4. Research Question 4

In the fourth research question, the reliability of the user and how it affects performance is the focus.

RQ4. How does the reliability of users affect the accuracy of classification over time?

Reliability entails several different aspects of possible human behaviour, but what all bring to light is the important acknowledgement that human users often do not behave in the same fashion as the perfect oracles often assumed in active learning research. Since we did not have a dataset where reliability aspects of the user were included as factors, the mHealth dataset and the synthetic dataset were used with the reliability factors of the users being simulated. While a potential drawback is that the behaviour is simulated and not actual recordings, it did give a higher level of control in the experiments.

The reliability aspects that were considered to be most prominent and with the highest probability of affecting the accuracy were *fallibility*

and *reluctance*. By fallibility we here mean that the labels that the user provides are not always correct. By reluctance we here mean that the user do not always provide a label in accordance with the employed interactive learning strategy, e.g., due to lack of attention. How fallibility and reluctance of the user affects performance was explored in paper IV. As can be expected, an increased reluctance of the user leads to decreased accuracy. The results also shows a decreasing accuracy with increased fallibility of the user. Where the effect of the two different factors differs is in the nature of the decrease.

The decline in accuracy when the reluctance is increased is at a steady pace and has a similar effect regardless of the interactive learning strategy and machine learning method employed. The different interactive learning strategies and machine learning methods does perform at different levels, as discussed with regards to RQ2 and RQ3 in previous sections, but none of them appears to be significantly more robust towards a reluctant user. In practice, what happens when the user is reluctant is that it will take longer until a certain amount of labelled data has been collected. This leads to the accuracy gradually becoming lower when the reluctance is increased.

When the level of fallibility of the user is varied, the choice of machine learning algorithm had a significant impact on robustness. For Support Vector Machine, there is a big drop in performance as soon as fallibility of the user is introduced, especially on the synthetic dataset. This is not a surprising result when considering how the classifier works. As soon as incorrectly labelled data points are introduced, the calibration of optimizing hyperplanes that separates all classes, i.e. how the Support Vector Machine is trained, becomes difficult. The calibration has to comprise too much for it to result in useful classification output.

The combination that appears to be most robust towards fallibility is the Naïve Bayes classifier with *MT triggered by error*. The Naïve Bayes classifier uses averaged values, i.e. even if a portion of the values are incorrect the total average is not necessarily affected to a high degree. The interactive learning strategy *MT triggered by error* is corrective in nature. If an incorrect label is provided due to fallibility this might affect the performance negatively, but this will also keep the user providing labels, which gives more opportunity to provide correct labels.

5.5. Limitations

While the research has been designed to answer the research questions as well as possible, there are limitations, both by choice and imposed from the problem scenario.

There is a limitation to the amount of interactive learning strategies and machine learning algorithms that has been implemented and tested.

Three separate machine learning algorithms were chosen because of their suitability to the given problem setting. While the selection gives an indication of the effect of machine learning algorithms on the results in the different experiments, it is of course limited to algorithms that have been included. The limitation was a conscious choice since the main focus of our research is not to compare all possible machine learning algorithms or develop new ones. There already exists previous work surveying online machine learning algorithms [29]. From related work it was also possible to find the machine learning methods that would be most suitable for our problem setting. With this said, it could be of interest to further explore options with regards to machine learning methods in the future.

In the experiments, only a selection of possible interactive learning strategies has been implemented and evaluated. In paper II for instance, one implementation of each category of interactive learning strategy is chosen to represent their category. The selection was made to get suitable representations for each category. Still, it is possible, if not likely that a different implementation or version of some of the strategies would have yielded different results. Since the interactive learning aspect is one of the main elements of the research, it will continue to be investigated in future work.

Four datasets in total were employed to perform the experiments. These were chosen to have varying characteristics such as number of classes, number of features, types of classes, etc., with the aim of exploring the research questions in different contexts. There were still simulations and modifications on the datasets that had to be performed as part of the experiments however, e.g. the dynamic sensor setting, because of the limitations of the datasets. Another limitation of most datasets that were considered to be included, was the length of the recordings. Because of the cold start scenario, it is of interest to view how the performance changes over time which in turn demands a certain length of the data stream. This was solved by simulating longer sessions if needed but ideally the datasets containing longer recordings would be used.

Another issue that is of importance to acknowledge is the difficulty in measuring performance over time, especially in a cold start scenario. Because of the cold start, the model does not have any data to train on until the first labelled data instance is provided. The datasets used here typically have the same class label over a period of time. At first the model will therefore only have labelled data from one class. Gradually, as more labelled data is gathered, the number of classes that has been introduced to the model increases. It is therefore possibly misleading to use the same performance metric when only two classes have been introduced compared to later in the stream when the model has labelled data from ten different classes.

One limitation with regard to the interactive aspect is that human subjects were simulated in the experiments. As was discussed in section 3, this is a limitation but does also provide more control over the experiments.

6. Conclusions and Future Work

6.1. Conclusions

In this thesis, the problem of detecting an activity or estimate the state of an environment in real-time was explored. The problem setting where this was studied included data streaming from a dynamic set of sensors and labels provided by a user in a cold start scenario. A novel taxonomy consisting of seven different interactive online machine learning strategies was presented. The strategies were implemented and tested on benchmark and synthetically created datasets. While the experiments show that the choice of machine learning algorithm can affect accuracy, there is not one algorithm that always performs best regardless of the problem at hand. For the interactive learning strategies, some overall trends could be seen from the results. As expected, for a given strategy, the accuracy increased when the labelling budget increased. The gain in accuracy got lower the larger the labelling budget, leading to a flattening of the curve of accuracy over labelling budget size. The choice of interactive learning strategy displayed a relative large effect on the result. Certain strategies resulted in better performance compared to others, even though less labelled data was used for training. This shows that the choice of which data instances to use for training, i.e. which interactive learning strategy is chosen, has a significant impact on the performance.

The experiments also showed that the reliability of the user did affect the result, but that the effect varied depending on the choice of interactive learning strategy and machine learning algorithm. A reluctant user, i.e. a user that did not always provide a label in accordance with the given strategy, resulted in lower accuracy regardless of interactive learning strategy or machine learning algorithm. While the different strategies and algorithms appeared equally robust towards a reluctant user, that was not the case for a fallible user. A fallible user, i.e. a user that did not always provide a correct label, overall decreased the accuracy but the loss in accuracy varied depending on the chosen interactive learning strategy and machine learning algorithm. The Naïve Bayes classifier with an interactive learning strategy where a label is provided when a classification is incorrect was most robust towards a fallible user.

6.2. Future Work

The work in this thesis has tried to answer certain questions, but has also led to new questions which could deepen knowledge by further study. A selection of possible directions are described below.

6.2.1. Machine learning algorithms

In this thesis three different machine learning algorithms were employed in the experiments. While these algorithms were chosen for their suitability in the problem setting based on previous research, a more comprehensive survey of the performance of different machine learning algorithm could increase knowledge in the area. Since the results show that the best performing algorithm depends on the problem setup, it would be of interest to study if there are correlations between characteristics in the setting and performance of the algorithms.

6.2.2. Experiments with human subjects

While there do exist several benchmark datasets on activity recognition and other applications where streaming data is of interest, there are still aspects missing. In the experiments, the datasets were adjusted so that the aspects of interest could be studied, whenever suitable datasets were not available. For instance, simulations were done to create a dynamic sensor setting and varying the level of reliability of the user. Another example is the length of individual recordings in many activity recognition and smart building datasets. To study the accuracy over time in a cold start setting, sufficiently long recordings of data streams are needed. When the recordings were not long enough, several different recordings were put one after the other to create one longer recording. While simulations can give more control over an experiment compared to if all aspects instead are part of the environment being recorded, it is still of interest to study the aspects as they appear naturally.

To validate (or challenge) the conclusions, experiments with human subjects would be of interest. If the previously mentioned aspects are studied in a real-world setting, they appear naturally instead of being simulated. This could result in more realistic representation of fatigue, reaction time of a user, etc. To record this type of data and create an open source dataset would also be of interest. It would create a possible benchmark for future research in the field.

6.2.3. Interaction design

The focus of this thesis has been on different aspects of the system for data fusion. To be able to explore these aspects others parts of the problem setting have had to be put aside. One example of this is the design of the human-computer interaction. The physical connection between

the human user and the system is needed to provide labels. While it is assumed in the work that an interface is in place, the design of it is not discussed. In a real-world scenario however, the design of the interactive component of the system might affect the performance. A well designed interface could, for instance, lead to a higher amount of labelled data from user or to a higher level of correct labels.

6.2.4. Privacy and security

Whenever the data collected contains information related to humans, privacy is a concern. Furthermore, in a setting where new sensors streaming data can appear and human users are providing labels, security is also a possible issue. While privacy and security were not the focus of this thesis, they are two important issues that needs to be addressed in the research field of interactive machine learning as a whole.

Bibliography

- [1] K. Ashton et al., “That ‘internet of things’ thing,” *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.
- [2] A. Whitmore, A. Agarwal, and L. Da Xu, “The internet of things—a survey of topics and trends,” *Information Systems Frontiers*, vol. 17, no. 2, pp. 261–274, 2015.
- [3] M. Luštrek and B. Kaluža, “Fall detection and activity recognition with machine learning,” *Informatica*, vol. 33, no. 2, 2009.
- [4] K. Akkaya, I. Guvenc, R. Aygun, N. Pala, and A. Kadri, “Iot-based occupancy monitoring techniques for energy-efficient smart buildings,” in *2015 IEEE Wireless communications and networking conference workshops (WCNCW)*, IEEE, 2015, pp. 58–63.
- [5] P. Domingos and M. Pazzani, “Beyond independence: Conditions for the optimality of the simple bayesian classifier,” in *Proc. 13th Intl. Conf. Machine Learning*, 1996, pp. 105–112.
- [6] B. Krawczyk, “Active and adaptive ensemble learning for online activity recognition from data streams,” *Knowledge-Based Systems*, vol. 138, pp. 69–78, 2017.
- [7] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [8] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, “Machine learning for internet of things data analysis: A survey,” *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

- [9] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [10] Z. A. Khan and A. Samad, "A study of machine learning in wireless sensor network," *Int. J. Comput. Netw. Appl*, vol. 4, pp. 105–112, 2017.
- [11] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.
- [12] A. Krishnakumar, "Active learning literature survey," in *Technical Report*, University of California, 2007.
- [13] I. Žliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39, 2013.
- [14] P. Donmez and J. G. Carbonell, "Proactive learning: Cost-sensitive active learning with multiple imperfect oracles," in *Proceedings of the 17th ACM conference on Information and knowledge management*, ACM, 2008, pp. 619–628.
- [15] T. Miu, P. Missier, and T. Plötz, "Bootstrapping personalised human activity recognition models using online active learning," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, IEEE, 2015, pp. 1138–1147.
- [16] X. Zhu, "Machine teaching: An inverse problem to machine learning and an approach toward optimal education," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [17] X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty, "An overview of machine teaching," *arXiv preprint arXiv:1801.05927*, 2018.
- [18] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

- [19] A. I. Schein, A. Popescu, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 2002, pp. 253–260.
- [20] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.
- [21] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, pp. 75–105, 2004.
- [22] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.
- [23] S. M. Sanchez, P. J. Sánchez, and H. Wan, "Work smarter, not harder: A tutorial on designing and conducting simulation experiments," in *2018 Winter Simulation Conference (WSC)*, IEEE, 2018, pp. 237–251.
- [24] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, pp. 28–39, 2016.
- [25] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen, "The opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, 2013.
- [26] O. Banos, R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez, and C. Villalonga, "Mhealthdroid: A novel framework for agile development of mobile health applications," in *International workshop on ambient assisted living*, Springer, 2014, pp. 91–98.

- [27] O. Banos, C. Villalonga, R. Garcia, A. Saez, M. Damas, J. A. Holgado-Terriza, S. Lee, H. Pomares, and I. Rojas, "Design, implementation and validation of a novel open framework for agile development of mobile health applications," *Bio-medical engineering online*, vol. 14, no. 2, S6, 2015.
- [28] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural computation*, vol. 8, no. 7, 1341–1390, 1996.
- [29] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *arXiv preprint arXiv:1802.02871*, 2018.

Part II.

PAPERS

PAPER I: Collaborative Sensing using Dynamic Intelligent Virtual Sensors

Agnes Tegen, Paul Davidsson, Radu-Casian Mihailescu
and Jan A. Persson

Abstract. Although the availability of sensor data is becoming prevalent across many domains, it still remains a challenge to make sense of the sensor data in an efficient and effective manner in order to provide users with relevant services. The concept of virtual sensors provides a step towards this goal, however they are often used to denote homogeneous types of data, generally retrieved from a predetermined group of sensors. The DIVS (Dynamic Intelligent Virtual Sensors) concept was introduced in previous work to extend and generalize the notion of a virtual sensor to a dynamic setting with heterogeneous sensors. This paper introduces a refined version of the DIVS concept by integrating an interactive machine learning mechanism, which enables the system to take input from both the user and the physical world. The paper empirically validates some of the properties of the DIVS concept. In particular, we are concerned with the distribution of different budget allocations for labelled data, as well as proactive labelling user strategies. We report on results suggesting that a relatively good accuracy can be achieved despite a limited budget in an environment with dynamic sensor availability, while proactive labeling ensures further improvements in performance.

In: Sensors 2019, 19, 477, DOI: <https://doi.org/10.3390/s19030477>

1. Introduction

With the evolution of sensor technology, Internet of Things (IoT), and high performance communication networks, making use of the vast amount of data being generated is critical for developing new and more powerful applications. This may concern supporting the user by distilling information from the incoming sensor data streams and presenting the user with the relevant sensory data just-in-time, while accounting for the user's interaction with the cyber-physical systems in its environment. It may also concern how the sensor data produced by a specific device can be used concurrently in multiple applications. One way to address these new challenges is to create an abstraction layer, which is overlaying the physical infrastructure. This is often referred to in the literature as a “virtual sensor”, which has the role of isolating applications from the hardware by emulating the physical sensor in software [1].

The abstraction layer provided by a virtual sensor can cater for multiple logical instances, which may support various applications with various goals, where the different virtual sensors may make use of overlapping sets of sensors [2]. Hence, it can enable a flexible way to develop services based on fusing sensor data either owned by the same provider or obtained from multiple sensor infrastructure providers. Overall, virtual sensors may facilitate concurrency by means of having different, possibly overlapping, subsets of sensors committed to different tasks. A key aspect of virtual sensors, which we are addressing in this work, has to do with measuring properties for which no corresponding physical sensor exist, e.g., the type of activity that is going on in a room. That implies that the virtual sensor needs to be capable of data fusion.

1.1. Related work

An example of deploying virtual sensors is in handling the situation when some sensors are malfunctioning. Such a solution is proposed in [3], where the authors present VirtuS, a prototype implementation of virtual sensor for TinyOS. The system uses stacks to store data temporarily, offering the possibility to perform basic arithmetic operations, e.g., computing the mean value of a number of data readings, before returning the value to the application.

Another use of virtual sensors is that of removing the occurrence of outliers in sensor readings. It is typically based on correlating data from a number of the same type of sensors, located in proximity to each other. As a result of this aggregation, not only are the readings more reliable but also, transmission is more efficient since less data needs be transmitted to the upper layers. This is particularly important in wireless sensor networks [4]. Similarly, in [5] a virtual sensor approach is employed to gather track-data from several visual sensors and to determine a performance score for each sensor based on the coherence with the rest of

the data. This information is then fed back to the sensor, allowing it to use it as external information in order to reason about the confidence of its readings. Along the same lines, active noise control is an often addressed topic in the context of virtual sensing. The challenge here is to create quiet zones around certain target points, without actually placing sensors at those desired virtual locations [6]. Among other applications, we also note the traffic domain where virtual sensors are used to provide congestion information in real-time, such as in the case of Washington state's traffic system [7].

A common characteristic for many of the existing virtual sensor solutions is that they provide a way to process homogeneous types of data, generally retrieved from a predetermined grouping of sensors. However, there are also a number of examples where virtual sensors combine heterogeneous data types for computing new values. Mobile devices, such as smartphones are a clear example of exploiting the sensors embedded in the device: accelerometers, gyroscope, GPS, microphone, camera, proximity sensors, ambient light sensors, and so forth, in order to fuse the data and obtain an overall and more complete perspective of the environment and the users' activities [8]. Note that in such an application the set of sensors are constant and are normally assumed to always be available.

1.2. Contribution

In this paper, we introduce a refined version of the Dynamic Intelligent Virtual Sensors (DIVS) concept [9] and validate some of its properties by experiments. A DIVS can be seen as a logical entity that takes the output from physical sensors and produces novel data that is needed by the applications/services. One main refinement of the original concept is to include interactive machine learning, by introducing the user-in-the-loop in order to improve accuracy and minimize the requirement for labelled data.

In cooperation with companies and facility management representatives we have identified a number of potential services for users including: understanding the usage of multipurpose office environments, guidance for users of the building and real-time monitoring of the number of people at different areas of an office complex, e.g., for evacuation. DIVS is intended to directly support such services through the possibility to define the suitable property of the environment, which it should be monitoring, as well as improving its accuracy by utilizing feedback regarding its output.

The paper is organized as follows: in Section 2 we propose a refined version of the DIVS concept. In Section 3 we demonstrate and evaluate our proposed approach in the context of a typical IoT setup. In Section 4 we discuss results and Section 5 concludes the paper.

2. Materials and Methods

A DIVS produces virtual sensor data for which there is typically no off-the-shelf sensor available (or difficult/expensive to develop). For instance, it may produce sensor data, which is a classification of the states of an environment by measuring a multitude of physical properties through physical sensors. The DIVS concept is broader than that of a virtual sensor per se, which is typically used in devices with a fixed set of sensors, such as in smartphones. In contrast, in our previous work [9] we introduced DIVS to denote an abstract measurement resulted from combining a number of heterogeneous data sources provided by a group of physical sensors and/or other DIVS, whereas the set of sensors supporting a DIVS can be adapted over time to better accommodate the current context and accuracy of the measurement. In real-world distributed sensing there are numerous instances where such adaptation needs to be performed on-the-fly in order to cope with the inherent uncertainty of the sensor infrastructure, such as sensor malfunctions, mobile sensors that are entering or leaving the system, new (type of) sensors being appended to the system, sleep cycles of sensors, battery constraints etc.

In addition to the aforementioned challenges, in this paper we extend the DIVS concept by focusing on some of the key learning modalities responsible for producing the DIVS output, in response to several issues that are characteristic to the IoT domain. In particular, we aim at addressing the inadequacy of standard machine learning approaches in computing the prediction (estimation or classification) function of the DIVS in situations when:

- there are changes to the patterns observed in the sensor data are occurring over time, cf. concept drift [10];
- sensor data streams suddenly disappear, or new sensor data streams appear;
- there are a lack of labeled data available for supervised learning, at least initially, cf. cold start problem [11] and one-shot learning [12];
- there is a need to efficiently and rapidly configure the DIVS;
- it is not possible to store all sensor data and therefore incremental learning [13] must be used;
- the ability of reusing information learned in one context to another context is desired, cf. transfer learning [14].

Our main goal is to propose a general-purpose design for virtual sensing with high versatility across different environments and applications. To this end, we start by revising the set of DIVS attributes in order to ensure the minimum viable data quality characteristics that enables us to tackle these challenges.

2.1. Data Quality Characteristics

Data obtained from IoT sensors and devices comes in various types and formats, is often noisy, incomplete and especially inconsistent [15]. Even when standardized ways are used to connect sensors to IoT platforms, such as using the MQTT or REST protocols or the JSON format, the data “packaging” is often carried out in unconventional ways, leading to data descriptions that are only human-interpretable. Importantly, data quality metrics can assess the usefulness of the data for a data consumer. In [16] the authors provide a study of data quality, while the work of [17] discusses data quality in IoT, emphasizing its importance in order to deliver user engagement and acceptance of the IoT paradigm and services. Contrary to common belief, data accuracy is just one of the aspects that ensure high qualitative data. According to [16], data quality can be evaluated in terms of the following dimensions and metrics respectively i) intrinsic: accuracy and reputation, ii) contextual: timeliness, completeness and data volume, iii) representational: interpretability and ease of understanding, iv) accessibility: accessibility and access security. In the following, we introduce a refined version of the DIVS attributes in response to the new set of challenges presented in the beginning of Section 2, while accounting for the data quality metrics previously identified. In order to ensure a minimal representation of the DIVS output, we introduce the distinction between mandatory and optional data attributes, as well as, data items that are continuously streamed and data items that are only transmitted when needed or made available upon request. The continuously streamed data items are:

- Sensor ID (iii)
- Timestamp (ii)
- Sensor value (i)

Whereas the data items provided on request or when needed are:

- Data type (iii)
- Property type (iii)
- Range of values (iii)
- Unit of measurement (iii)
- Physical location being sensed (ii)
- Estimated accuracy of measurement (i)—optional
- Event triggered or periodical sensing (iv)—optional
- Information gain (ii)—optional

The attributes of DIVS, which represent in our view the basic metadata description of a virtual sensor, can be directly correlated to the data quality dimensions in [16]. For instance, the contextual dimension is given here through the timestamp and physical location attributes, which capture the spatio-temporal characteristics of the data. The physical location being sensed is especially important for the case of mobile sensors

and location should in such cases be transmitted when changed. In addition, the information gain holds further contextual information about sensors data consumed by the DIVS (see Section 2.2). Representational information, which ensures the interpretability of the data, is given via the sensor ID, data type, property type (what aspect of the environment is being sensed), range of values and unit of measurement. Except for providing a basic characterization of the data, this information can further provide input for the pre-processing phase, such as detecting malfunctioning sensors that are producing data values that mismatch the given data type or are out of range. Moreover, the range of values attribute is essential, whenever a normalization or scaling of the data needs to be performed. A special attribute, related to accessibility, is assigned for specifying whether the sensor data is streamed at periodical intervals of time or the transmission is event triggered. The intrinsic dimension in terms of data quality, is given here by the estimated accuracy of measurement, which is a self-assessment, confidence measure of the DIVS about its prediction, or a precision of measurement in the case of regular sensors. In Figure 6.1 we outline the system architecture of DIVS, emphasizing the functionalities of the core components. One main refinement of the original DIVS concept is to include the feedback feature, which enables DIVS to actively query the user or other DIVS regarding the accuracy of the data. Thus, we provide a real-time architecture capable to integrate feedback information on-the-fly in order to improve performance. In the following, we provide a functional description of each of the DIVS components.

2.2. Dynamic Sensor Selection

In terms of data volume, IoT sensors can be regarded as one of the main data producers. However, the proliferation of sensing devices through large-scale deployments often comes at a cost in terms of sensor reliability (i.e., uncertainty of sensor availability) and noisy data. Moreover, sensors may be mobile, e.g., embedded in a smartphone, which adds to the uncertainty of the availability. Under these circumstances, it becomes imperative to provide DIVS with built-in mechanisms to assess which sensors are most significant in relation to the predictive task at hand. This problem is exacerbated when limited labelled data is available for training the model, in which case, a reduced number of relevant sensors is preferred.

Given a set of sensors representing the available data sources for a DIVS, the DIVS keeps track of a dynamic metric, termed information gain (IG). The IG has the role of assessing the usefulness, or the importance of a particular sensor for computing the desired output of the DIVS. There are clearly various ways for specifying the IG. For instance, in information theory terms, the notion of information gain is used to denote

Figure 6.1.

The DIVS data processing pipeline. DIVS 1.0 belongs to previous work and DIVS 2.0 is the extension included in this paper.

the reduction in the uncertainty of a random variable, given the value of another variable [18]. Important to note is the fact that here we use IG not merely as static parameter computed at design-time, but one that needs to be continuously updated based on the data provided by the sensor at run-time. In addition, we may introduce a threshold value, such that a sensor is not considered by the DIVS whenever its IG drops below the threshold. Furthermore, the use of IG can be instrumental for scenarios where some form of transfer learning is required. Consider for instance the case of a DIVS monitoring the type of activity in an office space. Suppose now that we want to run the same DIVS to monitor a new area of the building based on a slightly modified mix of heterogeneous sensors. Thus, having a good understanding of the usefulness of a particular type of sensor input for computing the DIVS output, enables us to reuse information learned in one context to another context.

2.3. Online Learning

The standard scenario for machine learning is the so-called batch learning or offline learning, where a model is trained on a given set of labelled instances. We have considered this approach for DIVS in previous work [9], where heterogeneous sensor data undergoes a data fusion and classification procedure via batch learning, in order to produce the DIVS output, as depicted in Figure 6.1. In contrast, in incremental learning a model is trained on data which arrives over time in streams. Online learning is a specific type of incremental learning where, after every time

step, an updated model is computed based on the previous model and the new data point (or mini-batch of data points) [13]. Most methods used for online learning are standard Machine Learning approaches that has been adapted for an online setting. For instance, online versions of Support Vector Machines, Random Forest and Naïve Bayes classifier has been suggested [19].

In this work, we augment DIVS with online learning capabilities in order to cope with two types of situations that are prevalent in the IoT ecosystems. In the first type of scenario the aim is to improve efficiency and scalability of an already existing model that has been trained in the traditional batch learning fashion with a labelled dataset. The second type of scenario is to handle situations where data naturally arrives over time and no labelled data is available in advance. Especially in the case of virtual sensing, it is key that the system is able to display online learning capabilities. That is, on the one hand, to continuously update an initially deployed model such that it is adaptable to the possibly changing target concept [20], while on the other hand, it can deal with the cold start problem when needed. This functionality provides DIVS with the versatility to operate in settings where the system can either leverage on historical data, or kick-start its process without the assumptions of previously labelled data. Moreover, given the dynamic nature of IoT, an online learning model can quickly adapt to incorporate new types of features (i.e. new sensors), or to discard them. It is also generally the case that online learners do not require to store the incoming data, which makes them a suitable candidate for pervasive embedded devices.

2.4. Active learning with user-in-the-loop

We further extend the DIVS concept by integrating an active learning mechanism. This assumes that DIVS are able to receive a certain level of user feedback or possibly, feedback from other DIVS. Especially in the situation of a cold start, unless, the learning algorithm used is unsupervised, it is critical to have a way of interacting with the users of the system in order to obtain labelled data. However, labelling data is in often expensive and might in many cases be difficult to obtain. A solution to this problem could be to only label selected instances, instead of all data. By allowing the learning method to choose which data points to label, the number of labeled instances can decrease, while still matching, or possibly even exceeding, the performance of a learner trained on a fully annotated dataset. Letting the learner decide what data points to learn from is referred to as Active Learning. The Active Learning method does this by posing a query to an “oracle”, which then provides a label for the given data point. The “oracle” can be a human user or expert, or it can be another system. How many instances can be queried is defined by a labelling budget. The selection process of which instances to query

can be made based on different criteria depending on the problem at hand. A common selection criterion is for the learner to query for the instances where it is least certain [21].

Recall now that for our IoT setup we are dealing with streaming sensor data, which means that the query decisions must be made on-the-fly, as data arrives. This poses more demands on the learning model than if all data is readily available. The labeling budget cannot be calculated over an infinite time horizon, which means that the budget must be balanced throughout the learning process. Different approaches to tackle these problems are presented and compared in [22].

To sum up, the architecture of the DIVS data pipeline is shown in Figure 6.1. The four major processes are as follows: (1) Data pre-processing, which has the role of preparing the incoming sensor data for further processing by smoothing noisy data, identifying and removing outliers, and resolving data inconsistencies; (2) Dynamic sensor selection and featurization, which makes use of information gain in order to identify the most suitable sensors, possibly reducing the dimensionality of the data and/or introducing additional statistical features (e.g. variance, FFT, etc.), as well as allowing DIVS to operate in a dynamic environment; (3) Online Incremental Learner, which is responsible for ensuring an open-ended learning procedure, capable to incorporate new data in order to improve the DIVS performance; (4) Active Learning, which fosters the interaction with users or other DIVS, by allowing it to obtain labelled data, while at the same time minimizing the amount of queries to the user.

3. Results

To illustrate the DIVS concept and validate some of its properties, we present a set of experiments where a DIVS uses data streams from a heterogeneous set of sensors to estimate the occupancy of an office room. Sensors might physically leave or enter the space, they might stop working or a new sensor might be added. This results in the availability of the sensors being dynamic, i.e. the set of data streams delivered to the DIVS is not constant over time. Thus, the DIVS must continuously adapt to make best use of the data from the sensors that are currently available.

As is the case in many real world applications, we assume that there is no labelled data from the given environment available prior to the deployment of the DIVS, corresponding to a “cold start”. Data from the currently active sensors arrives gradually over time, but to also obtain corresponding labels, i.e. whether the office space is occupied or not, feedback from a user is needed. In some of our scenarios, it is possible for the DIVS to query users for labels, but also for the users to proactively provide labels when they consider it necessary. Different active learning

strategies, as well as strategies with a proactive user, are presented in the experiments described below.

3.1. Dataset

We use the collection of datasets from [23]. The data collection is made up of three different datasets using the same set-up in a room in an office setting, but over different periods of time. Each dataset contains a recorded sequence of one data instance per minute from five features (light, temperature, humidity, CO₂-level and humidity ratio) along with whether the room was occupied or not at that point in time. The length of the data streams is between 2664 instances (1.8 days) to 9752 instances (6.8 days). The state of the room is occupied during similar hours and in similar patterns to what a regular work day might look like. On most work days it is occupied from around 8.30 a.m. to 5.30 p.m., with a longer period of being unoccupied around lunch time and some shorter periods throughout the day. During days corresponding to the weekend however, the room was unoccupied for the entire day.

In the experiments, the dynamic availability of the sensors is simulated by restraining the access to data from some of the sensors during periods of time. We randomly generate from which sensors data is withheld, when it is withheld and for how long. This represents the mobility of the sensors, i.e. when sensors leave the environment and therefore stop streaming data or new sensors appear and start to stream data. While the sensors could stop uploading due to e.g. sensors breaking down or network failures, the longest dataset is less than a week long and contains five features. The probability in a similar real world scenario of one or several of the sensors, to stop uploading for these reasons within a week is not very high. Therefore, we did not base the simulation of a dynamic setting on Mean Time Between Failures or equivalent. Instead, we randomly generated each scenario as follows. One to three sensors, out of five, were randomly chosen for each simulation. For each of these sensors it was randomly generated whether they would be disappearing sometime during the sequence (i.e. stop uploading data) and/or appearing (i.e. going from not uploading data, to start uploading data) and at which point in time this would happen. This means that a sensor can disappear and stop streaming data at one point, but later reappear and start streaming again, during the same simulation. Thus, within each scenario at each point in time at least two sensors are uploading data, but during periods of time at least one sensor is absent, to represent the mobility of the sensors.

3.2. Learning Method

We use a weighted Naïve Bayes approach for classification and the Variable Uncertainty Strategy for active learning. Naïve Bayes classifier has several advantages given the type of setting we are looking at. It requires fewer training examples compared to many other machine learning algorithms [24], [25] and is suitable for online learning, i.e. when one instance of data is processed at a time. The classifier is not computationally complex, which is of importance when producing real-time predictions from streaming data. This makes it a popular choice for when annotated data is scarce, but also when the dataset is gradually built up [26].

The set of sensors supporting a DIVS can change over time, which means that a learning model that can handle this dynamic setting is needed. The different features used for classification in Naïve Bayes are handled separately, which makes it possible to add a new one (if a new sensor appears) or discard an old one (if the sensor stops uploading data), without retraining the entire model.

In DIVS, information gain is used to describe the explanatory power of one property towards computing the output value. In a classic Naïve Bayes classifier, each feature is equally influential on the output. With information gain however, the influence on the output of each separate feature can be adjusted. The information gain is represented by weights which are assigned to each feature of the model. The weights are calculated from the overlapping coefficient [27], [28] of each feature and its distributions for the separate class labels.

In [22], Variable Uncertainty Strategy was suggested as a well-performing active learning approach for data streams when no strong concept drift is expressed. The approach aims to label the instances which it is least certain about, within a given time interval and labelling budget. The budget is expressed as a ratio of how many of the incoming data points can be queried compared to all incoming data points. Since it is not possible to calculate a ratio for the future incoming instances of the data stream, an estimation is made based on a sliding window. The sliding window contains information regarding which of the latest incoming data points, within the specified time interval, has been queried. Furthermore, since in many real world applications it is not possible to store all values from the data streams, only a limited number of labelled instances for each class is stored to be used for the Naïve Bayes classifier. These are also selected based on a sliding window approach, i.e. if the maximum number of stored instances allowed is reached and a new labelled data point arrives, the oldest instance gets discarded, something which also may support the handling of potential concept drift.

3.3. Experimental Setup

The experimental setup was designed to illustrate some of the core properties of the DIVS concept. In all of the experiments, the DIVS starts without any labelled data, i.e. cold start, and must gradually obtain this from a user. Different learning approaches were tested in a dynamic setting regarding the set of sensors streaming data, to demonstrate how a DIVS handles a changing set of input values. To illustrate how different types of user feedback can influence the performance, we tested different strategies for the user to provide input, from taking the role of “oracle” (as in classic Active Learning) to being more proactive (cf. Machine Teaching [29]). We also show how different budget ratios for querying, i.e. how much the DIVS can ask for feedback from a user affects the accuracy of classification. To get variation in the sequence of incoming data, the data was divided by days (from midnight to midnight) and randomly shuffled for each simulation.

If nothing else is specified, the number of labelled instances stored for each class is set to 50, i.e. 50 labelled data points from when the room was occupied and 50 from when it was not. Tests were carried out where the number of stored labelled instances was increased, but this did not improve the accuracy significantly.

A limitation of the Naïve Bayes classifier is that at least two labelled data points for a class are needed for it to be able to calculate a standard deviation. However, the learner should produce predictions from the start. Thus, before the Naïve Bayes classifier can be used to make predictions, the label of the last query is used for prediction.

3.4. Budget Ratio vs Accuracy

To illustrate how changing the labelling budget affects the accuracy, different budget ratios were tested with the Variable Uncertainty Strategy. Each budget ratio was used for 50 simulations of shuffled days to produce an average of the accumulated accuracy at the latest point in time and its standard deviation. For this experiment all sensor values were included at all times.

Figure 6.2 displays the accumulated accuracy for different labelling budgets. With an increased labelling budget, the accuracy increases, as would be expected. This is most significant when the labelling budget is low, in which case increasing it boosts the performance significantly. Also, note that even with a 0.2 % labelling budget, the leftmost result in Figure 6.2, the performance is above 75 % on average, while 95 % accuracy is achieved with a 20 % labelling budget.

Figure 6.2.
The accumulated accuracy for different labelling budgets.

3.5. Learning Strategies in settings with static set of sensors

In this experiment we test how incremental learning with an active learning approach performs compared to an iterative batch learning approach. A static setting regarding the presence of sensor data was used, i.e. all sensor values were accessible at all times. The learning strategies were tested on 50 simulations of shuffled days, to produce an average accumulated accuracy over time.

Iterative Batch Learning: This learning strategy has unlimited budget and obtains labels for all incoming data points. It also has no restriction on how many labelled instances can be stored. Unlike the other approaches described below, that are all versions of incremental learning, this can be compared to doing batch learning with all thus far encountered data instances for each point in time. This strategy is used as a baseline, as there are no restrictions on labelling budget or storage of data.

Variable Uncertainty Strategy: The core of this active learning approach is described in section 3.2 Learning Method. For this experiment a labelling budget of 1 % was used, which was based on giving good enough accuracy, while not bothering a user with too many queries.

Figure 6.3 shows how the active learning approach compares to Iterative Batch Learning. Iterative Batch Learning performs well from the

Figure 6.3.
The accumulated accuracy for Iterative Batch Learning vs. Variable Uncertainty Strategy.

start, after 1 hour the accumulated accuracy is 97.80 % and after 1 hour it keeps increasing. This is expected, because of the unlimited labelling budget and unlimited memory of the learning strategy. The accumulated accuracy of the Variable Uncertainty Strategy does not perform as well as the baseline. Yet, after 144 hours (or 6 days), the accumulated accuracy reaches almost 91 %, while only using 1 % of the labelled instances that the Iterative Batch Learning uses.

Table 6.1 shows the F1 scores for the two strategies after 144 hours. Here, the Iterative Batch Learning outperforms the Variable Uncertainty Strategy, again, as expected. Still, the F1 score of the latter strategy is 0.8111, as compared to the F1 score of 0.9698 for the baseline strategy.

TABLE 6.1.

The accumulated accuracy and F1-score for learning strategies in static settings.

Learning Strategy	Iterative Batch Learning	Variable Uncertainty Learning
Accumulated accuracy, after 1 hour	97.80 %	92.39 %
Accumulated accuracy, after 6 hours	98.79 %	86.33 %
Accumulated accuracy, after 144 hours	98.71 %	90.94 %
F1 score, after 144 hours	0.9698 %	0.8111 %

3.6. Learning Strategies in settings with dynamic set of sensors

This experiment compares how different learning strategies perform in a dynamic sensor setting. In total, 100 simulations of randomized dynamic settings and shuffled days were made, to generate an average of the accumulated accuracy over time.

Variable Uncertainty Strategy: The method with the same parameter settings as described above was used.

Random Strategy: This approach has the same budget as the Variable Uncertainty Strategy, but instead of querying based on whether the certainty is low, the learner queries randomly.

Decreasing Budget Strategy: In this strategy the learner has unlimited budget for querying for a short period at the start, until it has collected 50 labelled data points for each class. After this, the labelling budget decreases to 0.5 % for the rest of the time. This could be seen as a user needing to constantly provide labels in the beginning, but after this is done, the user is questioned less. The total amount of queries over time is parallel to the other approaches.

Figure 6.4 displays the accumulated accuracy for the different learning strategies over time. The Random Strategy performs worse than the others at the beginning, but catches up to and even surpasses the Variable Uncertainty Strategy and performs on the same level thereafter. The

Figure 6.4.
The accumulated accuracy for Variable Uncertainty Strategy vs. Random Strategy vs. Decreasing Budget Strategy.

performance of a Naïve Bayes classifier is dependent on having labelled data that sufficiently represents the current distributions of the output classes. When enough instances have been collected, and given that the set is continually updated, the models for the two different strategies becomes gradually more similar. Another reason for the Random Strategy improving in relation to the Variable Uncertainty Strategy is that, especially after a while, the latter approach might not be as good at estimating its own accuracy. From the F1 scores displayed in Table 6.2, it is also clear that the Variable Uncertainty Strategy and the Random Strategy achieve a similar accuracy, while the Decreasing Budget Strategy does not perform as well.

TABLE 6.2.
The accumulated accuracy and F1-score for learning strategies in dynamic settings.

Learning Strategy	Variable Uncertainty Learning	Random Strategy	Decreasing Budget Strategy
Accumulated accuracy, after 1 hour	94.38 %	63.92 %	98.39 %
Accumulated accuracy, after 6 hours	87.01 %	80.67 %	91.90 %
Accumulated accuracy, after 144 hours	89.73 %	89.84 %	82.87 %
F1 score, after 144 hours	0.7645	0.7589	0.6190

3.7. User Strategies in settings with dynamic set of sensors

To examine how different approaches of a user providing feedback can affect the accuracy, strategies where the user was proactive to different degrees were tested in a dynamic sensor setting. As in the previously described experiment, 100 simulations were generated and an average of the accumulated accuracy over time was calculated from this. Variable Uncertainty Strategy with oracle: The method with the same parameter settings as described above was used. Whenever the learner queried the user, the user provides a label.

Variable Uncertainty Strategy with proactive user: This approach is similar to one described above, with the difference that the user only answers a query if the latest prediction were wrong.

Proactive user with budget: With this strategy the learner never queries for labels, instead they are provided by the initiative of a user. When the prediction made by the classifier is wrong, the user provides a label given a labelling budget. The labelling budget is the same as for the other approaches.

Proactive user with unlimited budget: In this approach the labels are also provided solely by the initiative of a proactive user, but with an unlimited labelling budget. This means that a new labelled data point is provided each time the learner predicts wrong.

Figure 6.5.

The accumulated accuracy for Variable Uncertainty Strategy with and without proactive user vs. Proactive User with and without budget limit.

Figure 6.5 compares different user strategies in a dynamic setting. As expected, a user that has an unlimited budget to correct the learner when it makes a wrong prediction, performs well from the start. The Variable Uncertainty strategy, i.e. letting the learner completely decide when to query, and the strategy where a proactive user provides a label whenever there is a wrong prediction, both given a budget, performs similarly most of the time. By combining these two approaches, that is, the Variable Uncertainty Strategy with a proactive user, the accuracy improves significantly. Interestingly, this does in fact mean that the user overall will provide less labelled data points, compared to the regular Variable Uncertainty Strategy. By not responding to queries when the predictions were correct, but the learner was still uncertain enough to pose the query, the budget can instead be used for the instances where the prediction was not correct, which could explain the increase in accuracy. The F1 scores in Table 6.3 showcase that all strategies with a proactive user has better accuracy, compared to the one without. The strategy where active learning and a proactive user are combined, however, does again achieve the highest accuracy, out of the strategies with a limited labelling budget.

It is worth noting that while the Variable Uncertainty Strategy used up its labelling budget of 1 %, the other two approaches with budgets

did not. The strategy with a proactive user used up 42 % of the budget (0.42 % of all instances) and the Variable Uncertainty Strategy with a proactive user only used up 28 % of the budget (0.28 % of all instances). The strategy with a proactive user with unlimited budget on the other hand labelled 9.6 % of all instances on average. After 144 hours (or 6 days), the Variable Uncertainty Strategy with a proactive user had an accumulated accuracy of 92.69 %. This can be compared to the approach with an unlimited budget that had an accumulated accuracy of 97.29 % after the same amount of time. Figures 3-5 all contain learning curves where the accumulated accuracy is very high in the beginning, then rapidly decreases for a while, after which it increases again, but at a slow and steady pace. This is due to the fact that the predictions at the beginning are strictly based on the label from the latest query. Since the state of the room is not rapidly changing most of the time, this will give a high accuracy for a short while. When the Naïve Bayes classifier has collected at least two labelled instances, it can start to be used to make the predictions.

TABLE 6.3.
The accumulated accuracy and F1-score for user strategies in dynamic settings.

Learning Strategy	Variable Uncer- tainty Learning, with oracle	Variable Uncer- tainty Learning, with proactive user	Proactive user with budget	Proactive user with unlimited budget
Accumulated accuracy, after 1 hour	89.95 %	94.92 %	89.80 %	97.26 %
Accumulated accuracy, after 6 hours	83.99 %	95.35 %	84.18 %	97.60 %
Accumulated accuracy, after 144 hours	88.49 %	92.69 %	90.43 %	97.29 %
F1 score, after 144 hours	0.7442	0.8144	0.7783	0.9329

4. Discussion

The Naïve Bayes classifier was chosen because of its ability to perform well given few training examples (compared to many other Machine Learning methods), its adaptiveness to a changing set of input values and suitability for online learning, among other things. Despite several desirable properties of the method, it also has its limitations. The Naïve Bayes classifier is built on the assumption that all features used are independent from each other, given the output. With datasets from the real world however, this is rarely the case. The assumption of independence among all features does not hold true for the dataset we used either. Still, even though the assumption is violated, the classifier has been shown to perform well in many cases [30]. Another disadvantage with the method is concerning the probability estimations for each output class. These tend to be either very close to 0 or very close to 1. This is not necessarily a problem for the classification, as the classifier only has to choose the class with the highest probability, but it may not give a good estimation of the actual probability. Since the probability is used in the Variable Uncertainty Strategy, this may affect the performance of the active learning. In the future we plan to compare the performance of the Naïve Bayes classifier to other methods with regards to DIVS.

The dataset that all the experiments were tested on was chosen because it is a set labelled with regards to the state on an environment, and which contains a set of heterogeneous features collected as sequences of data. There are however limitations with the chosen dataset. The different states were only two in number, displaying whether the room where the data was collected was occupied or not. Furthermore, the transition from one state to the other is relatively infrequent. Most days the pattern of whether the room was occupied or not followed standard working hours in an office, but for some days the room was unoccupied during the entire day. This means that for long periods of time the state does not change. At the same time, the data is based on a real-life scenario and displays how the proposed approach would perform in a similar setting.

Since the dataset only contains five different features, it was not possible to simulate the type of sensor-intensive scenarios where the dynamic properties of a DIVS could be fully explored. Moreover, even though the features were all different, the dataset only contains features with numerical values. This simplifies the problem of heterogeneous input data, compared to if there for instance had been a mixture of numerical and categorical features for the DIVS to handle. In the future we plan to do experiments with more complex datasets, both regarding the classes and the features.

Another essential property of the DIVS that we intend to explore in future work, is information gain and how it can be used through transfer

learning. For example, if a DIVS is trained on a set of sensors in a given environment, but then moved to be utilized in a different environment, the information gain could be transferred to the set of sensors in the new environment from the old one appropriately. Another example could be if a sensor in a dynamic setting stops streaming data, but another sensor measuring the same property starts streaming at a later point in time. In this case the information gain connected to the sensor which disappeared can be transferred to the new sensor. In both cases information that has been accumulated can be reused through transferring the information gain.

In Active Learning it is generally assumed that the “oracle” will always answer a query and that this answer is correct. In reality this might not be the case. A DIVS that is operating in an office environment, for instance, will most likely direct its queries towards the people currently present in the environment. However, the users providing feedback can change over time, which in turn will affect how the data is labelled in several ways. Firstly, different people might label the same state in different ways. For instance, a state labelled by one user as ‘meeting’, can be labelled by another user as ‘presentation’. Secondly, the reliability of the answer from a user may differ.

5. Conclusions

In this paper a refined version of the DIVS (Dynamic Intelligent Virtual Sensors) concept was introduced in order to support the provision of applications based on sensor data. The ambition of the concept is to address a number of challenges identified by considering real word applications together with companies. The concept includes a minimal representation of the DIVS output to address the challenges identified in a dynamic environment with respect to sensor availability. Some properties of the concept were validated through experiments including an interactive machine learning mechanism. In particular, different strategies for making use of labelled data from potential users were studied. It was shown that the distribution of different budget allocations for labelled data and the approach of a proactive user for labelling had a rather important impact on the accuracy.

The results suggest that a relatively good accuracy can be achieved despite a limited budget. Further improvement of the performance can be achieved by proactive labelling, that is to provide correct labels when the output is wrong. Interestingly, the results indicate a significant deterioration of performance can be obtained if too much of the budget is used up early in a real-world environment where sensor may disappear. Hence, it is evident that the careful selection of how to make use of labels provided by users in an environment with dynamic sensor availab-

ility is very important. This indicates that designing applications where users provide feedback when the system output is incorrect, is more efficient than having users regularly provide correct labels. Also, by using a learning strategy for managing the budget of labels combined with a proactive user behavior, the need of user feedback can be further reduced.

Bibliography

- [1] S. Madria, V. Kumar, and R. Dalvi, "Sensor cloud: A cloud of virtual sensors," *IEEE software*, vol. 31, no. 2, pp. 70–77, 2013.
- [2] R. Boutaba and N. Chowdhury, "A survey of network virtualization," *Comput. Net.*, vol. 64, no. 5, pp. 862–876, 2010.
- [3] P. Corsini, P. Masci, and A. Vecchio, "Configuration and tuning of sensor network applications through virtual sensors," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, IEEE, 2006, 5–pp.
- [4] M. Islam, M. M. Hassan, G.-W. Lee, E.-N. Huh, et al., "A survey on virtualization of wireless sensor networks," *Sensors*, vol. 12, no. 2, pp. 2175–2207, 2012.
- [5] F. Castanedo, J. Garca, M. A. Patricio, and J. M. Molina, "A multi-agent architecture to support active fusion in a visual sensor network," in *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, IEEE, 2008, pp. 1–8.
- [6] L. Liu, S. M. Kuo, and M. Zhou, "Virtual sensing techniques and their applications," in *2009 International Conference on Networking, Sensing and Control*, IEEE, 2009, pp. 31–36.
- [7] D. J. Dailey, F. W. Cathey, and T. Trepanier, Deployment of a virtual sensor system, based on transit probes, in an operational traffic management system. Transportation Northwest, Department of Civil Engineering, University of ..., 2006.
- [8] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer, "Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach," in *2012 16th in-*

- ternational symposium on wearable computers, Ieee, 2012, pp. 17–24.
- [9] R.-C. Mihailescu, J. Persson, P. Davidsson, and U. Eklund, “Towards collaborative sensing using dynamic intelligent virtual sensors,” in *International Symposium on Intelligent and Distributed Computing*, Springer, 2016, pp. 217–226.
  - [10] 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, 2015.
  - [11] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, “A meta-learning perspective on cold-start recommendations for items,” in *Advances in neural information processing systems*, 2017, pp. 6904–6914.
  - [12] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in *ICML deep learning workshop*, Lille, vol. 2, 2015.
  - [13] A. Gepperth and B. Hammer, “Incremental learning algorithms and applications,” in *Proceedings of the European Symposium on Artificial Neural Networks*, Bruges, Belgium, 2016.
  - [14] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, “Transfer learning using computational intelligence: A survey,” *Knowledge-Based Systems*, vol. 80, pp. 14–23, 2015.
  - [15] U. Ekedahl, R.-C. Mihailescu, and Z. Ma, “Lessons learned from adapting” things” to iot platforms in research and teaching,” in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1457–1460.
  - [16] R. Y. Wang and D. M. Strong, “Beyond accuracy: What data quality means to data consumers,” *Journal of management information systems*, vol. 12, no. 4, pp. 5–33, 1996.
  - [17] A. Karkouch, H. Mousannif, H. Al Moatassime, and T. Noel, “Data quality in internet of things: A state-of-the-art survey,” *Journal of Network and Computer Applications*, vol. 73, pp. 57–81, 2016.
  - [18] C. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

- [19] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261–1274, 2018.
- [20] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *arXiv preprint arXiv:1802.02871*, 2018.
- [21] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.
- [22] I. Žliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39, 2013.
- [23] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, pp. 28–39, 2016.
- [24] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014.
- [25] Z. A. Khan and A. Samad, "A study of machine learning in wireless sensor network," *Int. J. Comput. Netw. Appl*, vol. 4, pp. 105–112, 2017.
- [26] B. Krawczyk, "Active and adaptive ensemble learning for online activity recognition from data streams," *Knowledge-Based Systems*, vol. 138, pp. 69–78, 2017.
- [27] F. Schmid and A. Schmidt, "Nonparametric estimation of the coefficient of overlapping—theory and empirical application," *Computational statistics & data analysis*, vol. 50, no. 6, pp. 1583–1596, 2006.
- [28] H. F. Inman and E. L. Bradley Jr, "The overlapping coefficient as a measure of agreement between probability distributions and point estimation of the overlap of two normal densities," *Communications in Statistics-Theory and Methods*, vol. 18, no. 10, pp. 3851–3874, 1989.

- [29] P. Y. Simard, S. Amershi, D. M. Chickering, A. E. Pelton, S. Ghorashi, C. Meek, G. Ramos, J. Suh, J. Verwey, M. Wang, et al., “Machine teaching: A new paradigm for building machine learning systems,” arXiv preprint arXiv:1707.06742, 2017.
- [30] H. Zhang, “Exploring conditions for the optimality of naive bayes,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 02, pp. 183–198, 2005.

PAPER II: Activity Recognition through Interactive Machine Learning in a Dynamic Sensor Setting

Agnes Tegen, Paul Davidsson and Jan A. Persson

Abstract. The advances in Internet of things lead to an increased number of devices generating and streaming data. These devices can be useful data sources for activity recognition by using machine learning. However, the set of available sensors may vary over time, e.g. due to mobility of the sensors and technical failures. Since the machine learning model uses the data streams from the sensors as input, it must be able to handle a varying number of input variables, i.e. that the feature space might change over time. Moreover, the labelled data necessary for the training is often costly to acquire. In active learning, the model is given a budget for requesting labels from an oracle, and aims to maximize accuracy by careful selection of what data instances to label. It is generally assumed that the role of the oracle only is to respond to queries and that it will always do so. In many real-world scenarios however, the oracle is a human user and the assumptions are simplifications that might not give a proper depiction of the setting. In this work we investigate different interactive machine learning strategies, out of which active learning is one, which explore the effects of an oracle that can be more proactive and factors that might influence a user to provide or withhold labels. We implement five interactive machine learning strategies as well as hybrid versions of them and evaluate them on two datasets. The results show that a more proactive user can improve the performance, especially when the user is influenced by the accuracy of earlier predictions. The experiments also highlight challenges related to evaluating performance when the set of classes is changing over time.

To be published in Personal and Ubiquitous Computing,
DOI: <http://dx.doi.org/10.1007/s00779-020-01414-2>

1. Introduction

Ongoing advances in Internet of things technology lead to new possibilities within the application area of smart environment and activity recognition [1]–[3]. With an increasing number of devices in our surroundings streaming data, the opportunities to collect information about those surroundings increase. In sensor rich environments, the set of sensors that is streaming data might not be constant over time. Furthermore, as labelled data typically is costly to acquire, there might not exist any to train the machine learning model with. In this case the model can be incrementally trained on labelled data that is repeatedly provided by a user. It is unrealistic in most scenarios to assume that the user can label all incoming data. In most cases only a limited ratio of the data can be labelled. Because of the restrictions on labelling data, it is of importance to choose the data instances to label such that the information for the machine learning model is maximized. The focus of our work is to classify the state of a specified environment with a dynamic set of sensors and limited labelled data provided by users.

By a dynamic set of sensors, we mean that the set of sensors streaming data is varying over time. The reasons for the dynamicity may vary, e.g. the sensors might be mobile and can enter or leave the environment at different points in time, they might stop streaming due to sensor malfunction, or there might be network problems.

The estimation of the activity or environmental state is done at each point in time by gathering and fusing data collected from the sensors currently available in the environment. By including mobile sensors, e.g. in smartphones, the number of possible data sources increases compared to a static set of sensors, but it also introduces challenges. For instance, by using data from a set of sensors that is changing over time, a machine learning model used for learning and data fusion must adapt to use the sensors that currently are streaming data in an optimal way.

When using supervised or semi-supervised machine learning techniques, not only data, but also labelled data is needed, e.g. the activity corresponding to the current sensor data. While the amount of generated data grows with an increased number of devices streaming data, the annotation of the data is still costly and often difficult to acquire. When dealing with streaming data in real world applications, there might not be any labelled dataset available to train the model with at the start. In this case, often referred to as the cold start problem, the model needs to be incrementally trained on data that is annotated gradually over time. Furthermore, in the case of streaming data and real-time estimation, the statistical properties of the data might change over time. This means that the information gathered at one point in time might not correctly represent the scenario later. To resolve this issue, referred to as concept drift, the model needs to be updated with new labelled data [4].

The aim in active learning is that only a subset of the entire dataset has to be labelled and used for training while still receiving the same performance. The size of the chosen subset is constrained by a labelling budget. Typically, there is a trade-off between performance of the machine learning model and the amount of labelled data needed. In active learning, the subset is chosen with the goal to optimise performance given the budget available. To obtain the labels, an oracle is queried, which can be a human (expert or non-expert) or another system. Generally, it is expected that the oracle will always reply with a correct label when queried. In many real-life scenarios these assumptions are unrealistic, especially if the oracle is a human expected to reply in real-time. We introduce interactive learning as an extension of active learning. To distinguish interactive learning from active learning, we adapt the term "user", instead of "oracle". The term "oracle" implies an all-knowing entity, while the term "user" is better suited for a situation where a person present in the environment provides labelled data. With interactive learning we include the possibility for the user to provide a label without being queried but instead by their own initiative, as in machine teaching [5], [6]. By relaxing the assumptions of active learning and giving the user possibility to be proactive in the learning process, our goal is to investigate how different interactive machine learning strategies affect the performance.

In this paper, we build upon previous work [7]–[9] and examine how different labelling strategies affect the performance of the estimations from the learning strategy. We use datasets simulating an environment of a user and dynamic sensors streaming data. We present a first version of a taxonomy for different strategies a user might have, or in other words, what might influence a user whether or not to provide labels.

2. Related work

Activity recognition is a popular research field, the use of which has been acknowledged within many application areas related to smart cities, such as, safety, security, medicine and smart buildings. Recently, deep learning methods applied to activity recognition problems have in many cases outperformed other methods. Among the most popular methods are Convolutional Neural Network, Recurrent Neural Network and Restricted Boltzmann Machine [10]. Ordóñez et al. introduce a framework for activity recognition based on convolutional and LSTM recurrent units and test it on two benchmark datasets [11]. Ronao et al. propose a deep Convolutional Neural Network for human activity recognition using sensors in a smart phone [12]. Hasan et al. use deep hybrid feature models that are incrementally trained through active learning [13]. All three works have divided up the data set into two, one used in a initial train phase, and the other used for testing, i.e. none addresses the

cold start problem and require very large amounts of data to be available before any activity recognition can be performed.. While there are several advantages to using deep learning for activity recognition tasks, for instance the features can be constructed by the algorithm without the need of human experience or domain knowledge, it also comes with challenges such as being used in real-time and reliance on training data [10]. Deep learning is dependant on a adequate amount of data being available to train the model on before it can be used to produce estimations. Since we are focusing on a cold start scenario within this work, training the model before prediction starts is not possible.

The challenge of lacking labelled data for activity recognition tasks has also been addressed by using unsupervised learning [14]–[18]. Kwon et al. propose unsupervised learning methods that use data from smart-phone sensors for activity recognition [16]. Ye et al. introduce USMART, a technique based on unsupervised learning that combines knowledge- and data-driven techniques [14]. Azkune et al. introduce a system that combines unsupervised learning and knowledge-based activity models [15]. They receive comparable results to supervised learning approaches in the results presented. The setup of the experiments in these works are all done in an offline manner, i.e. not in the single-pass streaming fashion that is the case in many real world activity recognition applications, and that we employ in our experiments. Using unsupervised learning has the advantage of not needing labelled training data [14], [17]. However, the models are dependent on a domain expert to provide knowledge to the model, a work effort that amounts to roughly one days work for the expert in the systems proposed by Riboni et al. and Azkune et al. [15], [17]. The presented strategies can be useful in many settings, but in the cold start setting of this work, it is assumed that there is no prior training or knowledge provided.

Stikic et al. explore different methods that could reduce the required amount of labelled data, including active learning approaches [19], [20]. Data from wearable sensors are used to aid psychiatrists in providing insight into their patients behavioral patterns in a work by Dietrich et al. [21]. The authors explore how different visualizations can be used as feedback to the psychiatrists. Compared to these, in our problem setting the feedback has to be provided in real-time, starting from a cold start scenario.

Active learning within a static setting without streaming data is a well-studied problem [4], [22]. The subset of data to be labelled by the oracle is selected at one point in time as all data is available from the start, based on the active learning strategy and the labelling budget. With streaming data however, the set of samples cannot be chosen beforehand, because they arrive one sample at a time. Instead, the choice of whether to query or not has to be made with each arriving sample, still adher-

ing to the labelling budget over time. There are some work that have studied active learning with streaming data, e.g. Lughofer presents an overview of concepts, techniques, applications, challenges and more related to online active learning [23]. Online learning is a specific case of learning from streaming data, where one instance of data is processed at a time. Cheng et al. present a systematic framework with a feedback-driven active learning approach for streaming data with multiple classes and evaluate it on various datasets [24]. The approach aims to balance exploration and exploitation during the learning process.

Regardless of whether the data is streaming or not, active learning typically assumes that if and when queried, the oracle will always provide a correct label. However, in many real world applications this assumption is not realistic. Rather, the oracle is a human that is not necessarily an expert and is asked to provide labels in real-time. Furthermore, the person expected to answer the queries might not be the same person over time, as people might come and go through the environment, and sometimes no label at all is provided.

Previous work that takes into account the human factor of the oracle in Active Learning have mostly focused on interaction design or the user experience [25], [26]. Donmez et al. present a work where the assumptions of the oracle in active learning are relaxed [27]. The oracle can be reluctant (i.e. might not reply to a query) and unreliable (i.e. might provide an incorrect label). The authors suggest that by introducing these parameters to the oracle, the model is better suited for real-world applications. Cakmak et al. perform experiments where different levels of involvement and guidance from the oracle are compared to regular active learning [28]. It is pointed out that performance can be improved for active learning if the oracle also is allowed to have a more active role.

3. Experimental setup

In this section, the different machine learning methods and the different interactive learning strategies are explained. The setup of the experiments performed, along with the datasets used, are also presented.

Several popular evaluation methods for batch learning, e.g. cross-validation, is not directly applicable in a setting with streaming data. An evaluation method used in this type of scenario needs to work for the cold start problem, as well as respect the temporal order of data. The evaluation procedure used for all experiments was test-then-train [29]. When a new data instance is received through the incoming stream of data, the system first produces an estimate of the given instance. After this, the interactive learning strategy decides whether or not the instance should be labelled. If a label is provided, the machine learning model

is incrementally trained with the new labelled data instance. The performance is then measured by calculating the accumulated accuracy over time.

3.1. Machine Learning Approaches

Three different machine learning approaches were used in the experiments: Support Vector Machine, k-Nearest Neighbor and Naïve Bayes classifier. Different parameter settings were tested for all machine learning approaches, but here we will only present results from using the ones best suited for the given problem. The classifiers also had to be adapted to be able to handle the dynamic sensor setting. The adaption is described in the section for each respective method.

Because of the regular influx of new data, all data cannot be stored indefinitely. Instead, an upper limit is set of how many labelled instances can be stored per class. If the maximum number of instances are reached and a new sample arrives, the oldest sample is discarded. After different limits of sample size were tested, the limit was set to 50 labelled instances of each class. This was big enough to represent possible variation within a given class, while still small enough to be updated if the case of concept drift.

3.1.1. *Naïve Bayes classifier*

The Naïve Bayes classifier with Gaussian distributions assumed for the variables was included in the experiments. The classifier has several advantages that makes it a suitable choice for our scenario [30]. Compared to many other machine learning approaches, it requires less training data. It is not computationally complex, which is important when estimations are produced in real-time and is also suitable for online learning. The Naïve Bayes classifier handles the different features separately, which makes it easier to adapt to the dynamic sensor setting. If a sensor stops streaming, the corresponding feature is simply discarded from the model, as the other features are not affected by it. If a new sensor starts streaming, the labelled data instances including this new feature are stored in parallel with a dataset with the old set of features. When enough data is collected of the new instance it can be added to the model used for estimation.

3.1.2. *k-Nearest Neighbor*

The k-Nearest Neighbor approach is a good choice in our scenario because of its simplicity [31]. Basically, the method classifies a new sample based on the labels of the k instances in the dataset that most resembles the sample given a distance metric. The label which is most frequent among these k neighbors is assigned to the new label. In our experiments

we had $k = 3$. The simplicity of the method makes it suitable for online learning and its computational work can be limited through the number of labels being saved. Furthermore, only k labelled data instances are needed to start classifying, which is useful in a cold start scenario.

The adaption of k-Nearest Neighbor to the dynamic sensor setting was similar to that of the Naïve Bayes classifier, but with some changes. As the features are not dealt with independently, the addition of a new sensor is less flexible for these methods. A larger number of labelled samples are needed before including a new sensor in the model. In this case, the samples that were added before the arrival of the new sensor does not contain a value for the new feature, but were filled with an average of the so far collected samples of the new sensor.

3.1.3. Support Vector Machines

Support Vector Machines with a polynomial kernel was also chosen for the experiments. Support Vector Machines are, apart from also being suitable for online classification, efficient regarding memory usage and dealing with high-dimensional datasets [32]. Efficient memory usage is important for real-time estimations. Being able to efficiently deal with high-dimensional data is useful in a dynamic sensor setting which can result in many features. The adaption of Support Vector Machines to the dynamic sensor setting was the same as for k-Nearest Neighbor.

3.2. Interactive Learning Strategies

In this work we explored what can influence a user to provide feedback in a real-time learning scenario and implemented interactive learning strategies based on this. Active learning can be seen as a special case of interactive learning, where the user is triggered to provide a label by being queried. However, other factors might prompt the user to provide, or stop the user from providing, a label. Below, the different factors detected that could influence a user are described together with the implemented interactive learning strategy used in the experiments.

- **Uncertainty:** In active learning, the user will provide a label when queried by the active learning strategy. One of the most popular Active Learning strategies is to query when the model is uncertain regarding the estimation. To decide when the model is uncertain an uncertainty measurement has to be defined. There exists several different versions of uncertainty measurement, but they all depend on a boundary, such as a threshold for the uncertainty measure, labelling budget and the implementation of the strategy. Since we have three different machine learning approaches, each one needs their own implementation of an uncertainty measure.

For the Naïve Bayes classifier, we use the Variable Uncertainty

Strategy presented by Žliobaite et al. [33]. The classifier produces probabilities for all classes and the class with the highest probability is then chosen for the classification. The probability is compared to a given threshold to decide if the estimation is uncertain or not (i.e. if the model should query). In this strategy the threshold is not static over time. If there are few queries, it might indicate that the threshold is set too high and so the threshold is gradually lowered and vice versa.

For the Support Vector Machine and k-Nearest Neighbour, we chose active learning strategies proposed by Pohl et al. [34]. For Support Vector Machine, the distance from the given data point to the hyperplane was chosen as the measurement of how certain the classifier is. Like with the Naïve Bayes classifier, this measurement is compared to a threshold that can be altered over time. In the active learning strategy for k-Nearest Neighbor, more than two-thirds of the neighbors must have the same label for the estimation to not be considered uncertain.

- **Error:** A user can be influenced by the output from the machine learning model. The model produces estimations of the current state of the environment, and in some cases also the confidence of the given prediction or, in the case of multi-class classification, the probability of other candidates. If a user has access to this type of information, it could influence them on whether to provide or withhold labels. In our experiments, the strategy has been implemented so that the user will provide a label when the previous prediction was erroneous.
- **State change:** Since the users are assumed to have knowledge of the current state of the environment, as they themselves are present, this awareness can trigger users to provide labels. For instance, assume the machine learning model is supposed to estimate the currently ongoing activity in an office setting. If the user first sits and works by themselves, but then several other people enter and a meeting starts, the status has changed from silent work to meeting, which will trigger the user to provide a label. In our implementation, the user provides labels when there is a change in the state of the environment, otherwise not.
- **Time:** The user may provide labels at certain points in time, regardless of queries, the correctness of estimations, and the state of the environment. In this strategy, labels are provided with a given frequency which is calculated based on the labelling budget. The user can either by their own initiative provide labels with a given time interval, or they can be queried by a labelling strategy systematically.
- **Random:** Labels can also be provided at random. In this case

there can either be a labelling strategy that queries the user at random or the user themselves might provide labels at random. The user might have reasons related to themselves regarding why they would provide labels or not. For instance, a user might provide fewer labels when they are more stressed. These labelling patterns might seem random, but they are not necessarily that. While they might appear random with regards to the sensor data, estimations, time etc, it might only be because the dataset does not include the relevant features such as the stress level of the user over time. The lack of appropriate data is the reason these types of strategies are not included here. However, it could be interesting to explore such strategies further in future work.

While the factors are listed separately above, they could be combined to different degrees. For example, the decision to provide a label or not could be based both on being queried by the model and what is happening around the user in the environment. To examine how such hybrid strategies performs compared to the separate strategies, a number of hybrid strategies were implemented as well. The different implementations and combinations of strategies leads to a large amount of possible hybrid strategies. In this work, we decided to focus on the combinations of the strategy Uncertainty, with the other interactive learning strategies listed in the previous section. We chose these combinations, as the decision on whether a label will be provided would not only be with the learning model nor with the user, but based on strategies of both. The different hybrid approaches are described below.

- **Uncertainty + Error:** In this strategy, the first step is the Uncertainty strategy (described further in the previous section). The model queries the user when uncertain regarding its prediction, but the user will only respond to the query with a label if the latest prediction was incorrect.
- **Uncertainty + State change:** Similarly to the previous strategy, the first step is Uncertainty, querying whenever the uncertainty of prediction is considered big enough. However, the user will only respond if the state of the environment has changed since the last label was given.
- **Uncertainty + Random:** Also this strategy consists of two steps, where the first is Uncertainty. In this case however, it is randomly generated whether the user will provide a label as a response to the query. This strategy is similar to a setting where the Uncertainty strategy is employed, but the user does not always reply to a query.
- **Uncertainty + Time:** This strategy is also Uncertainty combined with another interactive learning strategy, but differs from the other hybrid strategies. Instead of one strategy following the other in two separate steps, the two strategies are run in parallel. The

interactive learning strategy, Time, has a separate counter that keeps track of how many instances has appeared since the last sample was labelled. Given that there is enough budget, a label is provided at a certain frequency, which is calculated from the labelling budget. At each point in time, the frequency and the uncertainty of the prediction is tested in parallel and if either one decides to provide a label or query, a label for the sample will be given. However, even though the strategies are run in parallel, they share the same labelling budget.

3.3. Datasets

We used two separate benchmark datasets within Activity Recognition and smart environment to evaluate the proposed methods. Both contain collections of sequential data, i.e. the data was recorded in sessions, from multiple heterogeneous sensors. Simulations were done on the datasets to create the dynamic set of sensors that is changing over time.

3.3.1. *Opportunity dataset*

The Opportunity dataset is a collection of recordings of different daily living activities performed separately by 4 subjects in a sensor-rich environment [35]. The subjects performed a scripted scenario in five different runs, giving a total of 20 recorded data sequences. The sequences are between 12.35 minutes (22230 instances) and 28.4 minutes (51116 instances) long. In total 72 different sensors, resulting in 242 features, were used for the recordings, some sensors were worn by the subject, others placed on objects in the room of the recording. For the experiments, 19 of the separate recordings were used (one was excluded, as it contained too many missing values). All tests were run on the data sequences separately and then averaged.

The data is annotated in several abstraction levels. For the experiments in this work, the highest abstraction level was used which includes four labels and a null category (which represents unlabelled data instances). The annotations at this level describe the mode of locomotion for the subject ("Stand", "Walk", "Sit" or "Lie").

The labelling budget was set to 5 % for the experiments on the Opportunity dataset based on tests on the different machine learning approaches and the interactive learning strategies. Figure 6.1 displays that 5 % leads to a good balance between keeping the budget low, while still trying to maximise performance. The labelling budget is in reference to the number of incoming samples, not the timestamp. With this dataset a 5 % labelling budget would equal 90 labels being provided each minute on average, which might not be realistic. While an ideal scenario would be recorded over longer periods of time, we consider the results to still be representative for our problem scenario, as the order in the sequences

Figure 6.1.
The accumulated accuracy for different labelling budgets and interactive machine learning strategies for the Opportunity dataset.

are maintained and the learning approaches does not use the timestamp as a feature.

3.3.2. Occupancy dataset

The Occupancy dataset contains recordings of whether or not a room in an office setting is occupied [36]. The dataset contains 3 separate sequences of recorded data, varying in length from 1.8 days (2664 instances) to 6.8 days (9752 instances) (6.8 days) and five different features (light, temperature, humidity, CO₂-level and humidity ratio). The pattern of when the room is occupied follows regular working hours during weekdays, but the room was empty the entire day on weekends.

With the Occupancy dataset, there was not as many separate recordings as with the Opportunity dataset. Instead, the data was divided by days (from midnight to midnight) and randomly shuffled for each simulation, to obtain variation in the sequence of incoming data. In total, 100 simulations were run and then averaged to produce the presented results. The labelling budget for the experiments on the dataset was set to 1 %, based on findings in earlier work [8].

3.3.3. Simulation of dynamic sensor set

To simulate a dynamic set of sensors streaming data, the access to some sensors were restricted during periods of time. The choice of sensors to be dynamic, and during which part of the sequence their data streams would be restricted, were randomly generated for each simulation. A sensor can be present at first, but later drop off, it can be absent from the start, but start streaming data at some point in time or a combination of the two. In case of a combination, the sensor could be streaming at the start, later drop off, and even later reappear again or vice versa. If a sensor was streaming at the start, then stopped and later start streaming again, the data collected from the first period of streaming was not

Figure 6.2.
The actual amount of data labelled for different labelling budgets and interactive

stored in the model for the reappearance. In a sensor-intensive environment there might otherwise be a risk of an ever increasing amount of data from sensors that might never reappear. An alternative could have been to store the data for a limited time for the possibility of the sensor reappearing within this time period. While this was not implemented in the following experiments, it is planned to be investigated in future work.

First, experiments were done where the number of dynamic sensors was set to a fixed number. When each dynamic sensor would be streaming data and which sensors they would be, was still randomly generated. In the second part of the experiments, the number of sensors with a dynamic presence may vary between simulations, but are bounded by a minimum and maximum value. For the Opportunity dataset, 20-80% of the sensors had a dynamic presence, while it was 20-60% of the sensors for the Occupancy dataset. The reason for the different intervals is because the two datasets have different number of features. While the Opportunity dataset has 242 features, the Occupancy dataset only has five features. For each dynamicity value, the average number of streaming sensors over time has been calculated and it is this number that is displayed in the results.

4. Results

Figure 6.1 displays how different labelling budgets affects the accumulated accuracy for the three machine learning methods and the interactive learning strategies on the Opportunity dataset. The figure shows that an increased budget leads to increased performance, with decreasing relative improvements in performance for larger budgets. When choosing a labelling budget, the trade-off between a lower budget and a higher performance has to be considered. While the optimal labelling budget might differ slightly for the different strategies, due to consistency in the

Figure 6.3.

The accumulated accuracy over the average number of streaming sensors for the separate interactive learning strategies on the Opportunity dataset.

Figure 6.4.

The accumulated accuracy over the average number of streaming sensors for the separate interactive learning strategies on the Occupancy dataset.

experiments one budget was chosen for all strategies. When taking the results of all strategies into account, the labelling budget was set to 5 %.

As figure 6.2 illustrates, even though the labelling budget is increased, the actual amount of labelled samples is not necessarily increased with an equal amount. While Uncertainty and Time in most cases uses up their labelling budget, Error never use up the allowed budget, but still manages to perform better then the other two strategies.

In figure 6.3 and figure 6.4 the results from the experiments when varying the dynamicity of the streaming sensors are showcased. The figures show the final value of accumulated accuracy given an average number of streaming sensors. For the Opportunity dataset, the results are the average of simulations done on 19 separate recordings. For the Occupancy dataset, the results are the average of 100 simulations.

The results from the experiments on the separate interactive learning strategies and machine learning approaches can be found in figure 6.5 and figure 6.6, for the Opportunity dataset and the Occupancy dataset, respectively. Both figures display the accumulated accuracy over time,

Figure 6.5.
The accumulated accuracy over time for the separate interactive learning strategies on the Opportunity dataset.

Figure 6.6.
The accumulated accuracy over time for the separate interactive learning strategies on the Occupancy dataset.

starting from the first prediction. The results in figure 6.5 is the average of 19 simulations, each from a separate recording and a varying dynamic set of sensors. Figure 6.6 displays the average of 100 simulations, also with a separate and varying dynamic sensor setting.

Figure 6.7 and figure 6.8 contain the results from the hybrid versions of the interactive learning strategies evaluated on the Opportunity dataset and the Occupancy dataset, respectively. The setup is the same as for the tests on the separate interactive learning strategies, regarding number of simulations, the dynamic sensor setting, etc.

The final figure, 6.9, contains several plots, all displaying the accumulated accuracy for the separate interactive learning approaches over time. These tests were done on one of the recordings of the Opportunity dataset to give an example of how the learning curves differ when the sequence of data from the recording is preserved compared to when all data points are shuffled. The upper row contains the experiments where the sequence is kept, while the lower row contains the same data points, but in a randomly shuffled order.

Table 6.1 and table 6.2 contains the percentage of all data points

Figure 6.7.
The accumulated accuracy over time for the hybrid interactive learning strategies on the Opportunity dataset.

Figure 6.8.
The accumulated accuracy over time for the hybrid interactive learning strategies on the Occupancy dataset.

that were labeled, i.e. the actual labelling expenses used as opposed to the allowed labelling budget for the experiments with the separate interactive learning strategies. In table 6.1 the actual amount of labelling used for the Opportunity dataset is displayed (where the labelling budget was 5 %) and in table 6.2 the same can be found for the Occupancy dataset (where the labelling budget was 1 %). While some strategies used up the allowed labelling budget, all did not. For the experiments on the separate interactive learning strategies, the Error and State change strategies did not use up the allowed budget in any of the cases. Error annotated around 2.26 % for the Opportunity dataset and around 0.35 % of all incoming samples for the Occupancy dataset. For State change the numbers are 1.37 % for the Opportunity dataset and 0.30 % for the Occupancy dataset. Uncertainty even ended up slightly over the allowed budget for the Occupancy dataset, with an actual budget around 1.33 %. These numbers were consistent with all of the machine learning approaches used.

Table 6.3 and table 6.4 similarly displays the percentage of all data points that were labeled for the hybrid interactive learning strategies for

Figure 6.9.

The accumulated accuracy over time for the separate interactive learning strategies. Top row displays the results when the order of the sequential data is kept, bottom row shows the results when the order is shuffled randomly.

TABLE 6.1.

The labelling expenses used over time on the Opportunity dataset for the separate interactive machine learning strategies and the three machine learning (ML) methods Naïve Bayes (NB), Support Vector Machine (SVM) and k-Nearest Neighbor (k-NN) when the labelling budget was set to 5%.

ML	Uncertainty	Error	State change	Time	Random
NB	5.53 %	2.39 %	1.37 %	5.01 %	5.28 %
SVM	5.55 %	2.09 %	1.37 %	5.01 %	5.29 %
k-NN	5.59 %	2.30 %	1.37 %	5.01 %	5.27 %

TABLE 6.2.

The labelling expenses used over time on the Occupancy dataset for the separate interactive machine learning strategies and the three machine learning (ML) methods Naïve Bayes (NB), Support Vector Machine (SVM) and k-Nearest Neighbor (k-NN) when the labelling budget was set to 1%.

ML	Uncertainty	Error	State change	Time	Random
NB	1.34 %	0.33 %	0.30 %	1.00 %	1.02 %
SVM	1.30 %	0.32 %	0.30 %	1.00 %	1.00 %
k-NN	1.34 %	0.39 %	0.30 %	1.00 %	1.02 %

TABLE 6.3.

The labelling expenses used over time on the Opportunity dataset for the hybrid interactive machine learning strategies and the three machine learning (ML) methods Naïve Bayes (NB), Support Vector Machine (SVM) and k-Nearest Neighbor (k-NN) when the labelling budget was set to 5%.

ML	Uncertainty +Error	Uncertainty +State change	Uncertainty +Time	Uncertainty +Random
NB	2.29 %	0.49 %	5.58 %	4.10 %
SVM	1.90 %	0.45 %	5.55 %	2.67 %
k-NN	2.33 %	0.49 %	5.59 %	4.17 %

TABLE 6.4.

The labelling expenses used over time on the Occupancy dataset for the hybrid interactive machine learning strategies and the three machine learning (ML) methods Naïve Bayes (NB), Support Vector Machine (SVM) and k-Nearest Neighbor (k-NN) when the labelling budget was set to 1%.

ML	Uncertainty +Error	Uncertainty +State change	Uncertainty +Time	Uncertainty +Random
NB	0.33 %	0.35 %	1.34 %	0.69 %
SVM	0.31 %	0.34 %	1.31 %	0.48 %
k-NN	0.39 %	0.35 %	1.34 %	0.71 %

the Opportunity dataset and the Occupancy dataset respectively. In the experiments with the hybrid strategies, the number of labelled samples acquired was in general lower, but in some cases on par with the number from the experiments on the separate strategies. The only strategy that used up the budget regardless of dataset and machine learning approach, was Uncertainty + Time. This approach actually used slightly more than the labelling budget when evaluated on the Occupancy dataset, as the total ended up around 1.33 % for all machine learning approaches. The hybrid strategy Uncertainty + Error, had a slightly lower but similar number of labelled data points compared to the separate strategy for both datasets. Both the hybrid strategies Uncertainty + State change and Uncertainty + Random acquired a similar amount or fewer samples.

5. Discussion

The results make it clear that the choice of interactive learning strategy has a significant effect on the performance when tested on recordings of streaming data. The State change strategy performs worst in almost all instances, regardless of machine learning approach or the dataset tested on. One reason might be that the strategy does not use up the allowed labelling budget, but it is not the only explanation, as the Error strategy does not use it up either (see table 6.1 and table 6.2). In fact, Error often collects a similar amount of labelled samples as the State change strategy for the Occupancy dataset. The Error strategy is the best, or among the best, performing strategies for all experiments displayed in figures 6.5 and 6.6. The reason for this could be that the strategy corrects the model as soon as it has made an incorrect classification, leading to a decreased risk of repeating the mistake. The results indicates that choosing an appropriate interactive learning strategy can have a significant impact on performance, as a higher accuracy can be achieved with fewer labels.

For active learning in a non streaming setting, a selection from a set of unlabelled data is chosen according to a specified strategy. By having all data available at once, the full labelling budget can be utilized, without going over the allowed budget either. Handling a labelling budget with streaming data however, means that estimations has to be made, as it only is possible to obtain a label for the current data point and the size of the total amount of data is unknown (or possibly infinite). Instead of using the entire dataset (which would include future unknown data points), the labelling budget is calculated over a window of time. The total labelling budget can therefore only be estimated at run-time. This leads to the approximate labelling budgets that can be found in tables 6.1 , 6.2 , 6.3 and 6.4 and also explains why the actual amount of labels in some cases do not always exactly match the labelling budget.

Another interesting observation from the experiments on the separate interactive learning strategies is that the active learning strategy, Uncertainty, has among the worst results in many cases, implying that other information than the uncertainty estimation of the model can be useful when deciding which samples to label. However, this work only contains one implementation per machine learning approach. While the choice of Uncertainty was made based on what would suit the problem at hand, it is not possible without further experiments to conclude if the lower performance is due to the strategy being used in general or if it is due to the specific implementations.

When comparing the hybrid approaches, displayed in figures 6.7 and 6.8, to their respective separate approaches, it varies whether the combination improved, worsened or did not affect the performance. Maybe unsurprisingly, the separate strategies that performed worse had most to gain from a combination with another strategy. The most drastic improvement can be seen for the State change strategy, when evaluated with the Naïve Bayes classifier on the Occupancy dataset. In this case the performance is much increased when compared to both Uncertainty and State change separately. These results again show that the Uncertainty strategies tested here are not always good at assessing their own performance.

Figure 6.3 and 6.4 displays the effect of varying the number of streaming sensors has on performance. For the Opportunity dataset, figure 6.3, the accuracy is almost unchanged for all the interactive learning strategies, even though the number of streaming sensors varies. This is reasonable however, as the dataset contains a large amount of features. As can be seen from the results, even if all sensors are dynamic, i.e. have an interval where they are not streaming data, there are still on average 153 sensors streaming at each point in time.

For the Occupancy dataset, figure 6.4, the performance does change when the average number of streaming sensors is altered. Interestingly, the performance does not always increase with an increased number of sensors. For most of the strategies, the performance increases to around an average of 4.19 streaming sensors, after this, it either stagnates or even decreases, before it increases yet again. A possible reason for this result can be found with the features in the Occupancy dataset. When studying how well the different features correlates to the status of occupancy in the room, it becomes clear that some of the features have a low correlation. For instance one feature measuring the humidity in the room, has a correlation coefficient close to zero. In the experiments presented in here there was no feature selection, but these results indicate that the model might benefit from adding a preprocessing step of feature selection however. From the results it can also be seen that Error seems to be the most stable interactive learning strategy when the number of

streaming sensors is changing.

It is worth noting regarding the labelling budget that the case is quite different with streaming data compared to a static dataset. In a setting with streaming data, it is not possible to store all incoming samples for an infinitely long time or to know what the future samples in the stream are. Instead of calculating the budget on the entire dataset, which would include future unknown samples, it is calculated on a set of the most recent data points. Furthermore, the labelling budget is the maximum percentage of incoming data points that the model can query for labels, but there is no minimum value. This means that a restrictive approach might not use up the allowed labelling budget when calculated over a period of time.

The chosen performance measurement for these experiments was accumulated accuracy over time, starting from the very first prediction, when the model in most cases only has one labelled sample. The accuracy over time is interesting to analyse, because of the data being provided sequentially and the fact that the model begins training without any stored data. The performance measurement falls short of providing a complete picture for several reasons however, with the changing number of classes over time possibly being the most prominent one. At the very beginning of each experiment, the learning model has only encountered one class, making the estimation trivial. Gradually, as more classes are introduced, the estimation becomes increasingly more complex. Other common performance metrics for machine learning problems suffer from the same issue, however. To the best of our knowledge there is not an established measurement that takes this into consideration.

The gradual introduction of new classes is one factor in the decreasing accuracy seen in many of the experiments. The accuracy decreased even after all classes had been incorporated into the model however, implying that it was not the only reason for the downward trend. To see whether the specific sequence of data from the recordings could influence the performance curves, tests were carried out where a sequence of data was shuffled in a random order. As figure 6.9 shows, this does affect the shape of the learning curve significantly. Here, all classes are introduced early on, which results in a poor performance at the start. As more labelled samples are collected, the performance increases rapidly at first, but stabilizes after a while. The reason for the smooth look of the curve with the shuffled dataset, is probably because the entire spectrum of possible feature values for each class is introduced from the start. While this might be an overwhelming task at first, the model can learn more effectively. When the order of the sequence is kept the model might first give an impression of being able to estimate well, but as time moves forward the full complexity of the problem unfolds. In real-world scenarios however, streaming data does arrive with a higher possibility of resembling

other data points close in time and not as a shuffled sample set for the model to begin its training on.

6. Conclusion and future work

In this work we explored the effect of interactive machine learning strategies on performance in a real-time learning scenario. The strategies were implemented and evaluated on two datasets related to activity recognition with a simulated dynamic sensor setting, both as separate strategies and as hybrid versions. The experiments show that by giving the user a more proactive role, the performance can be increased. In the overall best performing strategy, the decision of whether to provide a label was based on the accuracy of the previous prediction, i.e. a user would provide feedback in the form of a correct label when the previous estimation was incorrect. Future work includes further developing the taxonomy and broaden the different types of implementations and strategies used in experiments. This can be done by, for instance, testing multiple different active learning strategies for comparison, and in combination, with other interactive learning strategies. Future work also include testing more machine learning methods.

The experiments highlight the difficulty in comparing performance over time when the complexity of the problem is not constant, e.g. by a varying number of classes. In future work we intend to define and explore possible measurements suitable for comparing problems with different complexities or in a streaming data setting where the complexity of the problem is changing over time.

Even though the datasets used here were adequate for evaluating the proposed strategies, there is a need of good datasets in the area of Activity Recognition or smart environment which includes longer recordings of streaming data, heterogeneous types of sensors and multiple classes. The aim for future work is to create an open source dataset with multiple heterogeneous sensors recording the state of office environments.

This research was partially funded by the Knowledge Foundation through the Internet of Things and People grant number 20140035. On behalf of all authors, the corresponding author states that there is no conflict of interest.

Bibliography

- [1] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context aware computing for the internet of things: A survey,” *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 414–454, 2013.
- [2] F. Alam, R. Mehmood, I. Katib, N. N. Albogami, and A. Al-beshri, “Data fusion and iot for smart ubiquitous environments: A survey,” *IEEE Access*, vol. 5, pp. 9533–9554, 2017.
- [3] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, “Context-aware computing, learning, and big data in internet of things: A survey,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 1–27, 2017.
- [4] A. Krishnakumar, “Active learning literature survey,” in *Technical Report*, University of California, 2007.
- [5] X. Zhu, “Machine teaching: An inverse problem to machine learning and an approach toward optimal education,” in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [6] X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty, “An overview of machine teaching,” *arXiv preprint arXiv:1801.05927*, 2018.
- [7] A. Tegen, P. Davidsson, and J. A. Persson, “Interactive machine learning for the internet of things: A case study on activity detection,” in *Proceedings of IoT’19: International conference on the Internet of Things (IoT’19)*, ACM, 2019.
- [8] A. Tegen, P. Davidsson, R.-C. Mihailescu, and J. A. Persson, “Collaborative sensing with interactive learning using dynamic intelligent virtual sensors,” *Sensors*, vol. 19, no. 3, p. 477, 2019.

- [9] A. Tegen, P. Davidsson, and J. A. Persson, "Towards a taxonomy of interactive continual and multimodal learning for the internet of things," in *Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, ACM, 2019, pp. 524–528.
- [10] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.
- [11] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.
- [12] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert systems with applications*, vol. 59, pp. 235–244, 2016.
- [13] M. Hasan and A. K. Roy-Chowdhury, "A continuous learning framework for activity recognition using deep hybrid feature models," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1909–1922, 2015.
- [14] J. Ye, G. Stevenson, and S. Dobson, "Usmart: An unsupervised semantic mining activity recognition technique," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 4, no. 4, pp. 1–27, 2014.
- [15] G. Azkune and A. Almeida, "A scalable hybrid activity recognition approach for intelligent environments," *IEEE Access*, vol. 6, pp. 41 745–41 759, 2018.
- [16] Y. Kwon, K. Kang, and C. Bae, "Unsupervised learning for human activity recognition using smartphone sensors," *Expert Systems with Applications*, vol. 41, no. 14, pp. 6067–6074, 2014.
- [17] D. Riboni, T. Sztyley, G. Civitarese, and H. Stuckenschmidt, "Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 1–12.

- [18] T. Dimitrov, J. Pauli, and E. Naroska, “Unsupervised recognition of adls,” in *Hellenic Conference on Artificial Intelligence*, Springer, 2010, pp. 71–80.
- [19] M. Stikic, K. Van Laerhoven, and B. Schiele, “Exploring semi-supervised and active learning for activity recognition,” in *2008 12th IEEE International Symposium on Wearable Computers*, IEEE, 2008, pp. 81–88.
- [20] M. Stikic and B. Schiele, “Activity recognition from sparsely labeled data using multi-instance learning,” in *International Symposium on Location-and Context-Awareness*, Springer, 2009, pp. 156–173.
- [21] M. Dietrich, E. Berlin, and K. Van Laerhoven, “Assessing activity recognition feedback in long-term psychology trials,” in *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, ACM, 2015, pp. 121–130.
- [22] B. Settles, “Active learning literature survey,” University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.
- [23] E. Lughofer, “On-line active learning: A new paradigm to improve practical useability of data stream modeling methods,” *Information Sciences*, vol. 415, pp. 356–376, 2017.
- [24] Y. Cheng, Z. Chen, L. Liu, J. Wang, A. Agrawal, and A. Choudhary, “Feedback-driven multiclass active learning for data streams,” in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, ACM, 2013, pp. 1311–1320.
- [25] E. Johns, O. Mac Aodha, and G. J. Brostow, “Becoming the expert-interactive multi-class machine teaching,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2616–2624.
- [26] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, “Power to the people: The role of humans in interactive machine learning,” *AI Magazine*, vol. 35, no. 4, pp. 105–120, 2014.

- [27] P. Donmez and J. G. Carbonell, “Proactive learning: Cost-sensitive active learning with multiple imperfect oracles,” in *Proceedings of the 17th ACM conference on Information and knowledge management*, ACM, 2008, pp. 619–628.
- [28] M. Cakmak and A. L. Thomaz, “Mixed-initiative active learning,” 2011.
- [29] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.
- [30] B. Krawczyk, “Active and adaptive ensemble learning for online activity recognition from data streams,” *Knowledge-Based Systems*, vol. 138, pp. 69–78, 2017.
- [31] Z. A. Khan and A. Samad, “A study of machine learning in wireless sensor network,” *Int. J. Comput. Netw. Appl*, vol. 4, pp. 105–112, 2017.
- [32] M. S. Mahdavejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, “Machine learning for internet of things data analysis: A survey,” *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.
- [33] I. Žliobaite, A. Bifet, B. Pfahringer, and G. Holmes, “Active learning with drifting streaming data,” *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39, 2013.
- [34] D. Pohl, A. Bouchachia, and H. Hellwagner, “Batch-based active learning: Application to social media data for crisis management,” *Expert Systems with Applications*, vol. 93, pp. 232–244, 2018.
- [35] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen, “The opportunity challenge: A benchmark database for on-body sensor-based activity recognition,” *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, 2013.
- [36] L. M. Candanedo and V. Feldheim, “Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models,” *Energy and Buildings*, vol. 112, pp. 28–39, 2016.

PAPER III: A Taxonomy of Interactive Online Machine Learning Strategies

Agnes Tegen, Paul Davidsson and Jan A. Persson

Abstract. In interactive machine learning, human users and learning algorithms work together in order to solve challenging learning problems, e.g. with limited or no annotated data or trust issues. As annotating data can be costly, it is important to minimize the amount of annotated data needed for training while still getting a high classification accuracy. This is done by attempting to select the most informative data instances for training, where the amount of instances is limited by a labelling budget. In an online learning setting, the decision of whether or not to select an instance for labelling has to be done on-the-fly, as the data arrives in a sequential order and is only valid for a limited time period. We present a taxonomy of interactive online machine learning strategies. An interactive learning strategy determines which instances to label in an unlabelled dataset. In the taxonomy we differentiate between interactive learning strategies when the computer controls the learning process (active learning) and those when human users control the learning process (machine teaching). We then make a distinction between what triggers the learning: active learning could be triggered by uncertainty, time, or randomly, whereas machine teaching could be triggered by errors, state changes, time, or factors related to the user. We also illustrate the taxonomy by implementing versions of the different strategies and performing experiments on a benchmark dataset as well as on a synthetically generated dataset. The results show that the choice of interactive learning strategy affects performance, especially in the beginning of the online learning process, when there is a limited amount of labelled data.

Submitted to conference.

1. Introduction

The performance of a machine learning method is dependent on the labelled data it is trained on. While the amount of data in the world is increasing at an accelerating rate, the act of labelling that data is often costly. In active learning, a limited number of data instances are chosen for labelling, as compared to all of them [1]. The aim is to achieve an equivalent performance to when all instances are labelled, by choosing the limited number of instances wisely. The active learning strategy selects which instances should be included and an oracle is then queried to provide correct labels for those instances. In machine teaching, the goal is also to achieve a high performance with a limited number of labelled samples, but with the teacher selecting the labelled instances to train the learner on [2], [3]. Often the oracle, in the case of active learning, or the teacher, in the case of machine teaching, is a human interacting with the system and we will henceforth adopt the term user to describe both.

In settings where data arrive as streams, the selection has to be done differently compared to a batch setting. In an online learning scenario, the data arrives in a specific sequential order and the current data instance is the only one that can be labelled at a given point in time. Furthermore, if the machine learning method is supposed to produce estimations in real-time, the computations might have to be done at the edge where algorithms often have restrictions on computational complexity.

In this work we present a taxonomy of interactive online machine learning strategies. With this we include active learning strategies, where the learner queries the user, and machine teaching strategies, where the user provides labels by their own initiative according to a specific strategy, as well as hybrid versions. We implement versions of the presented strategies and compare them through experiments on a benchmark dataset and a synthetically created dataset.

2. Related work

Fu et al. surveyed and organized existing work into two main categories based on whether the active learning strategy solely is based on the uncertainty of independent and identically distributed instances or if it also takes into account instance correlations [4]. Experiments comparing time complexity of the strategies discussed was carried out. While active learning for streaming data was mentioned as an emerging application and the related challenges discussed, the strategies are presented in a batch setting, where all unlabelled data is available at the same time. The strategies categorized as only being based on uncertainty can apply to an online learning setting, but the strategies presented where the instance correlations are included tend to be less suitable. First, they gen-

erally have a higher time complexity, which often is not suitable when estimations have to be produced in real-time. Second, they take into account a data instance's correlation to other instances, which means that a batch of data has to be collected before the strategy can be properly employed.

Active learning within a streaming data setting was explored by Miu et al. [5]. The authors present an online active learning framework, where user-provided annotations can be collected in real-time and used for Human Activity Recognition tasks. The framework is compared to more naïve annotation methods and evaluated on benchmark datasets. User studies were also carried out, where the framework was implemented in a mobile app through which the participants could provide labels. The user cannot provide labels by their own initiative however, as in machine teaching, but only when queried by the active learning strategy which is based on uncertainty of the learner.

Lughofer investigated active learning for data streams using evolving fuzzy classifiers [6]. Two different strategies of active learning are presented and tested, based on conflict and ignorance. Conflict means that the new data sample is on the border of previously defined classes, while ignorance means that the new data sample is far outside the previously defined borders. Both conflict and ignorance based learning can be considered strategies that are based on the uncertainty of the learning model. The paper does not explore a user providing labels by their own initiative.

A user with proactive capacities was studied by Chen et al., as they explored the role of adaptivity in algorithmic machine teaching [7]. In the work, the authors studied the teaching of version space learners in an interactive setting through experiments. An adaptive teacher, where the learner's hypothesis at every time step can be observed, was compared to a non-adaptive teacher, where only the initial hypothesis of the learner is known i.e. no feedback is received during teaching. In the first one, the teacher can proactively adapt the information provided to the learner on-the-fly, based on the performance. This can be seen as an interactive learning strategy where the user provides a label when the estimation of the learner is incorrect. The latter one can be seen as a batch setting where all examples are constructed beforehand. In the given setup, the learner did not have the possibility to query the teacher for labels however, and can only train on the instances provided by the teacher.

Most work where interactive machine learning strategies are employed focus on active learning, especially strategies based on the uncertainty of the learner. Several works compare different strategies, but do often not include strategies from different types of interactive machine learning categories. Particularly machine teaching in a setting with streaming data, is an area which needs further exploration.

3. Interactive online machine learning strategies

The different categories of sampling strategies presented below are all interactive online learning strategies. Interactive learning includes both active learning, where the learner queries the user, who in turn responds with a label and machine teaching, where the user proactively provides labels based on a selection strategy. Online learning means that the data stream is received in a single-pass manner, i.e. the strategy is processing one data instance at a time. Special cases of sampling strategies that are meant to be combined with specific machine learning methods (e.g. evolving models [8]) are not included. In Fig. 6.1, a visualization of the taxonomy can be seen.

In interactive online machine learning, several issues have to be taken into consideration that might not be as relevant for interactive machine learning in a batch setting. The most prominent issue is the one-pass manner in which the data instances arrive. Within an online setting a decision must be made for each new instance as it appears, whether or not to query for (in the case of active learning) or provide (in the case of machine teaching) the label. Streaming data also needs another approach for dealing with labelling expenses compared to a batch setting, since the incoming data instances in theory could be infinite. In the experiments the labelling expenses, unless otherwise stated, are calculated based on the the labelling status of recent instances. Other suggestions to handle the calculation over of budget with streaming data are presented for instance by Kottke et al. [9] and Žliobaite et al. [10].

For each type of strategy, an algorithm illustrates the structure in pseudo code. It is assumed in the following descriptions that the user never is fallible or reluctant, i.e. that the user always will provide a correct label when queried by an active learning strategy or in accordance with the machine teaching strategy in question. Each algorithm has a data stream $X = \{x_0, x_1, \dots\}$, a labelling budget B and classifier Ψ as input. In the case of additional input, it is further described for that particular strategy. The data stream contains a possibly infinite sequence of data instances, where x_0 is the first instance received by the learning algorithm. The labelling budget B determines the maximum ratio of incoming samples that can be labelled, with the restriction $0 < B \leq 1$, where $B = 1$ means that all incoming instances can be labelled. The classifier Ψ specifies which machine learning method is used along with any parameter settings. The classifier in its initial state can be pre-trained, if applicable data exists, or it might not have been previously trained, in the case of a cold-start scenario. The aim for the classifier is to accurately classify a state y . The classifier produces an estimate of the state $\hat{y}_i$ for each data instance x_i . The user provides a label y_i in accordance with the interactive learning strategy.

Figure 6.1.
Taxonomy of the interactive online machine learning strategies.

3.1. Active learning strategies

We consider three categories of active learning (AL) which can be triggered by uncertainty of the estimation, AL triggered by uncertainty, at a given time interval, AL triggered by time, and at random, AL triggered at random.

3.1.1. AL triggered by uncertainty

AL triggered by uncertainty is the most commonly used interactive learning strategy. The learner queries the user for labels for the data instances where it is least certain how to label [1]. Different implementations of the strategy exist and a lot of work in the area focus on comparing and developing strategies where the querying is triggered by uncertainty. As algorithm 1 shows, the decision whether to query or not is based on the output from an uncertainty measurement function $U(\hat{Y}_i, x_i)$, where $\hat{Y}_i$ is the possible classes and x_i is the current data sample, which is compared to a threshold value θ . Depending on which strategy is chosen for the function U and which classifier Ψ is used, $\hat{Y}_i$ might contain everything from only the most probable class $\hat{y}_i$ to all the possible classes ranked by probability. The implementation is especially straight-forward for probabilistic classifiers, but can be used with any classifier where an appropriate uncertainty measurement can be defined. Different versions of AL triggered by uncertainty include for example least confident, margin sampling, entropy and others where a region of uncertainty is explored [11]. In least confident, the learner queries when the probability of the prediction is low, margin sampling when the margin between the most probable and the second most probable class is low and in entropy when the combined probability of all possible classes is low.

Algorithm 1 AL triggered by uncertainty

Input: data stream $X = \{x_0, x_1, \dots\}$, labelling budget B , classifier Ψ , threshold θ , step size s
 $i \leftarrow 0$
repeat
  receive next data sample x_i
   $\hat{y}_i \leftarrow \Psi(x_i)$
  if $U(\hat{Y}_i, x_i) < \theta$ and $\hat{b} < B$ then
    ask user about label y_i for x_i
     $\Psi \leftarrow$ incremental learning procedure((x_i, y_i))
     $\theta \leftarrow \theta - s$
  else
     $\theta \leftarrow \theta + s$
  end if
  update labelling expenses $\hat{b}$
   $i \leftarrow i + 1$
until end of data stream

In the experiments described in section 3 we include least confident, where the uncertainty function becomes $U(\hat{Y}_i, x_i) = 1 - P(\hat{y}_i | x_i)$. The inverted probability is compared to a threshold θ which determines whether the classifier is uncertain regarding its prediction. The value of the threshold is set initially, but can be time-variable [10], meaning that it can be increased or decreased with the step size s depending on whether or not a query was posed. Note that if $s = 0$, the threshold is constant.

A sliding window is used to calculate the current labelling expenses $\hat{b}$. The window contains the labelling status of the most recent data instances. Each element of the window is either a 0, if there was no label provided by the user, or 1, if the user did provide a label. The labelling expenses are calculated by computing the ratio of how many instances in the window has had an accompanying label provided by the user. This value is compared to the labelling budget B to decide whether the strategy can query the user. The labelling expenses are updated by shifting all elements of the window in accordance with a queue system, i.e. when a new element is entered the oldest one is discarded.

3.1.2. AL triggered by time

In AL triggered by time, the algorithm queries at a given time interval. The sampling rate is calculated based on the labelling budget B , as can be seen in algorithm 2. A counter, c , is updated for every new data instance

that arrives to keep track of when it is time to query. This strategy can be useful for instance as an option if another strategy is not using up the allowed budget or in a cold start scenario. In the latter case, the classifier has no access to any labelled data in the beginning, which means that AL triggered by uncertainty might not work, depending on how the uncertainty measurement is defined. Another scenario where this strategy would be useful is if concept drift is present [12]. Concept drift means that the underlying statistical properties of the data is changing over time. AL triggered by uncertainty might be inadequate in this case, since the measurement of uncertainty is based on older data.

Algorithm 2 AL triggered by time

Input: data stream $X = \{x_0, x_1, \dots\}$, labelling budget B , classifier Ψ
 $c \leftarrow 1$
 $i \leftarrow 0$
repeat
    receive next data sample x_i
     $\hat{y}_i \leftarrow \Psi(x_i)$
    if $\frac{1}{B} \leq c$ then
        ask user about label y_i for x_i
         $\Psi \leftarrow$ incremental learning procedure((x_i, y_i))
         $c \leftarrow c - \frac{1}{B}$
    end if
     $c \leftarrow c + 1$
     $i \leftarrow i + 1$
until end of datastream

3.1.3. AL triggered at random

Similar to AL triggered by time, AL triggered at random does not take the data instance itself or the classifier into account when deciding whether or not to query. Unlike AL triggered by time however, the queries are not made with a set time interval, instead the time for querying is randomly chosen, as algorithm 3 shows. This is done by first randomly generating a variable ζ from the uniform distribution $[0, 1]$. The value of ζ is then compared to the value of the labelling budget B , to decide whether to query or not.

Algorithm 3 AL triggered at random

Input: data stream $X = \{x_0, x_1, \dots\}$, labelling budget B , classifier Ψ
 $i \leftarrow 0$
repeat
  receive next data sample x_i
   $\hat{y}_i \leftarrow \Psi(x_i)$
  generate a random variable $\zeta \in [0, 1]$
  if $\zeta \leq B$ then
    ask user about label y_i for x_i
     $\Psi \leftarrow$ incremental learning procedure((x_i, y_i))
  end if
   $i \leftarrow i + 1$
until end of datastream

3.2. Machine teaching strategies

The four categories of machine teaching (MT) presented can be triggered by incorrect estimations from the classifier, MT triggered by error, by a change of the state, MT triggered by state change, at a given time interval, MT triggered by time, and by factors related to the user, MT triggered by user factors.

3.2.1. MT triggered by error

In MT triggered by error, the decision of whether or not to provide a label is based on the estimation from the classifier. The user has access to the classifier's estimation of the current state. Whenever the estimation made by the classifier is incorrect, the user provides the correct label, given that the labelling expenses $\hat{b}$ does not exceed the labelling budget B . The labelling expenses are calculated and updated the same way as in AL triggered by uncertainty. Algorithm 4 showcases the process.

Algorithm 4 MT triggered by error

Input: data stream $X = \{x_0, x_1, \dots\}$, labelling budget B , classifier Ψ
 $\hat{b} \leftarrow 0$
 $i \leftarrow 0$
repeat
    receive next data sample x_i
     $\hat{y}_i \leftarrow \Psi(x_i)$
    if $\hat{y}_i \neq y_i$ and $\hat{b} < B$ then
        label y_i for x_i is provided by user
         $\Psi \leftarrow$ incremental learning procedure((x_i, y_i))
    end if
    update labelling expenses $\hat{b}$
     $i \leftarrow i + 1$
until end of datastream

3.2.2. MT triggered by state change

In MT triggered by state change, the user will provide a label when the state y changes, given that the labelling expenses $\hat{b}$ does not exceed the labelling budget B . The labelling expenses are calculated and updated the same way as in AL triggered by uncertainty. This strategy can be useful if there is a concern that all possible classes might not be properly represented in the labelled data used for training, e.g. if the dataset is unbalanced. Unbalanced datasets can result in a high performance overall, even though the performance related to the less frequent classes is poor. For instance, if AL triggered by time is used, the learner will have many labelled instances from the frequently occurring classes, while much less from the rare classes. A user that is aware of the current status, i.e. what label would be correct for the current data instance, could counteract this by providing labels based on the changing status of the state. As illustrated in algorithm 5, in MT triggered by state change, the user provides a label whenever there is a change in state, in other words, when the class of the current label changes. Depending on the scenario however, the state might not change often, resulting in a low number of labelled instances in total. On the other hand, as long as $\hat{b} < B$ holds, the user will provide a new label if the state changes. This means that while $\hat{b} < B$ and no new label is provided from the user, the learner can assume that the incoming instances have the same label as the last one provided by the user y_l and also add these to the collection of labelled data.

Algorithm 5 MT triggered by state change

Input: data stream $X = \{x_0, x_1, \dots\}$, labelling budget B , classifier Ψ
 $\hat{b} \leftarrow 0$
 $i \leftarrow 0$
repeat
    receive next data sample x_i
     $\hat{y}_i \leftarrow \Psi(x_i)$
    if $\hat{b} < B$ then
        if $y_i \neq y_{i-1}$ or $i = 0$ then
            label y_i for x_i is provided by user
             $y_l \leftarrow y_i$
             $\Psi \leftarrow \text{incremental learning procedure}((x_i, y_i))$
        else
             $\Psi \leftarrow \text{incremental learning procedure}((x_i, y_l))$
        end if
    end if
    update labelling expenses $\hat{b}$
     $i \leftarrow i + 1$
until end of data stream

3.2.3. MT triggered by time

The user can by their own initiative also provide labels with a given frequency. The algorithm for this strategy will almost be identical to the one for AL triggered by time, presented in algorithm 2. The only difference is that the user is not queried by an active learning strategy and instead keeps track of time themselves. Depending on the scenario, the user might be less exact compared to an active learning strategy that keeps track of time. In our setting however we assume that the user does always provide a label in accordance with the strategy, in which case it does not affect the implementation or the performance of the strategy where the initiative to provide a label is coming from. Therefore, the results from AL triggered by time are representing both AL triggered by time and MT triggered by time.

3.2.4. MT triggered by user factors

The user can be triggered by factors related to themselves. These factors include for instance the internal state of the user, e.g. their current stress level. A user might provide fewer labels if they are stressed compared to when they are not. Another factor can be the characteristics of the user, e.g. the knowledge level on what they are supposed to provide labels for.

For example, if a user only wants to provide labels if they are certain of the label, then a knowledgeable user could provide labels for a bigger portion of all possible data instances compared to a less knowledgeable user. The decision function of when a user will provide a label will vary depending on which user related factor is triggering the user. Because of the broad spectrum of possible strategies and the additional information about the user needed, this strategy is not included in the experiments.

3.3. Hybrid strategies

The strategies presented above do not necessarily have to be employed separately, but can be merged in different types of combinations to create hybrid strategies. There can be several reasons for creating a hybrid strategy. For instance, if an MT triggered by error does not use up its budget, it can be combined with AL triggered by time which will guarantee an influx of new labelled instances. Another reason can be to benefit from the strength of two separate strategies. For example, Žliobaite et al. present the Split strategy as a combination of AL triggered by uncertainty and AL triggered at random which can be useful especially if concept drift is occurring [10].

4. Experimental setup

To illustrate how the different types of interactive machine learning strategies compare, we evaluated implementations of them in experiments on a benchmark dataset and a synthetically generated dataset with the user interaction simulated. In all of the following experiments the learner started with no labelled data and had to incrementally learn as the labelled data was gradually accumulated over time. The evaluation was done in a test-then-train fashion, meaning that each data instance is first used for testing, after this the instance might be added to the training data together with a label to train the machine learning model. Which labels are chosen to be used for training is decided by the interactive learning strategy. The results show an average of 20 separate runs.

4.1. Datasets

4.1.1. Dataset 1: *mHealth*

The mHealth dataset contains recordings of activities within a health application [13], [14]. The recordings are done with wearable sensors, accelerometer, gyroscope, magnetometer and electrocardiogram sensor. Ten different subjects each perform a sequence of 12 different physical exercises in one recording instance. The exercises are performed one at a time, which means that when the interval of one type of exercise is over, it is not repeated in the same recording. In the experiments, the record-

Figure 6.2.

A visualisation of the distributions of the classes of the synthetic dataset.

ings were concatenated to create one longer sequence and the different physical exercises are repeated 10 times. The order of the recordings is randomly generated for each run.

4.1.2. Dataset 2: Synthetic dataset

The synthetic dataset contains 5 classes where the mean values for each class are randomly generated in a 2D space. Ten thousand samples are then drawn for each class from a normal distribution with the given mean value and standard deviation. The total of 50000 generated samples constitutes the synthetic dataset. In all experiments, the values are the same but the order of the values in the data stream is shuffled for each run. How long the interval is for each class is also generated by sampling from a normal distribution. Fig. 6.1 displays a visualisation of the distributions of the classes of the dataset.

4.2. Machine learning methods

To compare the different interactive learning strategies they were tested in combination with three different machine learning methods, Support Vector Machine, k-Nearest Neighbor and Naïve Bayes classifier. The machine learning methods included were chosen because they are suitable for online learning, real-time classification and frequently employed in similar settings [15]–[17].

5. Results and Discussion

Figs. 6.3a, 6.3c and 6.3e display the accumulated accuracy given the labelling budget for the mHealth dataset for the Naïve Bayes classifier, Support Vector Machine and k-Nearest Neighbor, respectively. In the Figs. 6.3b, 6.3d and 6.3f the accumulated accuracy over the amount of samples estimated, with a labelling budget of 3%. Figs. 6.4a, 6.4c and 6.4e show the accumulated accuracy given the labelling budget for the synthetic dataset for the Naïve Bayes classifier, Support Vector Machine and k-Nearest Neighbor respectively. Similarly as for the mHealth dataset, Figs. 6.4b, 6.4d and 6.4f display the accumulated accuracy over the amount of estimated samples, with a labelling budget of 0.5%.

Fig. 6.3 displays the results from the experiments on the mHealth dataset. The left column of Fig. 6.3 (with panels 6.3a, 6.3c and 6.3e) shows the final accumulated accuracy given the labelling budget. The figures show that when starting from a very low labelling budget, the performance substantially improves by only increasing the budget slightly. After a while however, there is only a small increase in performance, if any, even though the budget continues to increase. In the right column of the figure (panels 6.3b, 6.3d and 6.3f) the accumulated accuracy over number of samples can be found for a labelling budget of 3%. The best performing strategies, especially at the start, are MT triggered by error and MT triggered by state change.

In Fig. 6.4, the results from the experiments on the synthetic dataset are presented. The left column of Fig. 6.4 (panels 6.4a, 6.4c and 6.4e) displays the accumulated accuracy given the labelling budget. Compared to Fig. 6.3, it does not display the final accumulated accuracy, but the accumulated accuracy after 5000 samples, or a tenth of the total amount of samples, has been processed. Since this dataset does not contain any concept drift, the performance of the different strategies will approach each other as the number of samples is increasing. After a while, the amount of labelled data is enough regardless of strategy to perform at a similar level. Consequently, it is interesting to analyze when the amount of samples is still relatively low. As the experiments were done using a cold start setup, the performance at the start of the data stream gives an indication of how the strategies perform with a limited amount of

(a) NB, accuracy vs labelling budget.

(b) NB, accuracy vs samples.

(c) SVM, accuracy vs labelling budget.

(d) SVM, accuracy vs samples.

(e) kNN, accuracy vs labelling budget.

(f) kNN, accuracy vs samples.

Figure 6.3.

The results from the experiments on the *mHealth* dataset. The left column (a, c and e) displays the final accumulated accuracy over labelling budget for Naïve Bayes (NB), Support Vector Machine (SVM) and *k*-Nearest Neighbor (kNN) respectively. The right column (b, d and f) displays accumulated accuracy over number of samples for a labelling budget of 3% for Naïve Bayes (NB), Support Vector Machine (SVM) and *k*-Nearest Neighbor (kNN) respectively.

(a) NB, accuracy vs labelling budget.

(b) NB, accuracy vs samples.

(c) SVM, accuracy vs labelling budget.

(d) SVM, accuracy vs samples.

(e) kNN, accuracy vs labelling budget.

(f) kNN, accuracy vs samples.

Figure 6.4.

The results from the experiments on the synthetic dataset. The left column (a, c and e) displays accumulated accuracy after 5000 samples over labelling budget for Naïve Bayes (NB), Support Vector Machine (SVM) and k-Nearest Neighbor (kNN) respectively. The right column (b, d and f) displays accumulated accuracy over number of samples for a labelling budget of 0.5% for Naïve Bayes (NB), Support Vector Machine (SVM) and k-Nearest Neighbor (kNN) respectively.

labelled data. Also here, the overall best performing strategies are MT triggered by error and MT triggered by state change.

It is the two machine teaching strategies MT triggered by error and MT triggered by state change that generally perform best in the experiments. The results give an indication that letting the user be more proactive can be beneficial on performance. This is noteworthy since most interactive online machine learning strategies employed in the literature fall under the active learning subcategory. Online machine teaching is an area that currently deserves further exploration.

The reason for the high performance at the start in the right-hand side column of Figs. 6.3 and 6.4, contrasting the accuracy over samples, is the cold start scenario. The figures are displaying the accumulated accuracy for each new incoming sample. For the very first estimation, there will only be one labelled data instance collected. As the learner only has been introduced to one label at this point in time, it will continue to estimate this label until another option is presented. In many types of data streams it is probable that data instances with the same label follow consecutively for a period of time (e.g. data streams collected in an activity recognition setting). At the very beginning, for a short period of time, this leads to an accumulated accuracy of 100%. When other labels are introduced to the learner, the performance goes down. As more labelled data instances are gathered for each class however, the performance reaches a level that is more representative of the learner in the long run.

We chose to include experiments on both a dataset consisting of actual recordings, the mHealth dataset, and a synthetically generated dataset. The mHealth dataset gives an example of how the strategies perform in a real-life setting, where for instance data can be noisy and there might be concept drift. The synthetic dataset on the other hand, is not meant to represent a realistic scenario for data collection, but it provides a framework for comparing the different strategies. While we have included experiments on two different datasets, further experiments on a variety of datasets would be useful to get an exhaustive comparison of the strategies.

The high performance of MT triggered by state change is because it can collect a larger amount of labelled data to use as training data, while not having to involve the user all the time. The strategy expects the user to provide a label when the state is changing and if the labelling expenses still are below the allowed labelling budget, the learner assumes that the state is the same until the user provides a new label. The learner can thus continue to collect labelled data instances, without any effort from the user, as long as the labelling expenses are below the labelling budget. The current labelling expenses are calculated based on how many times the user has provided labelled data. The labelling expenses can thus be

kept low while the learner continues to gather labelled data

While the overall performance is high for MT triggered by state change, it relies on the assumptions that the user will always provide a correct label in accordance with the strategy as long as there is labelling budget. These assumptions are found in most work on interactive machine learning but whether they are realistic is rarely discussed. While there exists scenarios where these are reasonable assumptions (e.g. a medical doctor labelling data within their expertise), in many scenarios the assumptions are simplifications of the actual setting. An interactive machine learning strategy that performs well given the assumptions on the user might not necessarily perform as well if the user did not always provide a label or sometimes provided an incorrect label. We plan to explore how relaxing these assumptions on the user can affect performance in future work. Even though we kept the assumptions in the experiments, the taxonomy also covers scenarios where they are relaxed.

We assume in the experiments that there is only one user, or possibly multiple users, but with the same behavior, but there exists several settings with multiple users that all have individual characteristics. For instance in crowdsourcing, there are multiple users with the possibility to provide labels, but they might not always be available or willing to provide a label [18]. When they do provide a label, it might be incorrect. We also assume that only one label can be provided for each instance. In many scenarios where the classes are disjoint this is a reasonable assumption (e.g. is the room empty or not), but in other scenarios there might be a value in allowing multiple labels for one instance.

MT triggered by user factors was not included in the experiments because of the lack of necessary data. These types of factors are highly dependent on the user and strategies based upon them are difficult to model. The pattern of when the user is providing labels and not might not be known to an outside observer or even to the user themselves. Nevertheless, in many real-life scenarios they do have an impact on the result. For instance, a user might be more probable to provide a label when he or she is in a good mood and less likely to provide labels when in a bad mood or when they are stressed. Our aim is to explore these factors further in future work.

6. Conclusion and Future Work

In this work we have presented a taxonomy of interactive online machine learning strategies. We have also done implementations of the strategies presented in the taxonomy and performed experiments on one benchmark dataset and one synthetically generated dataset. The experiments show that the strategies where the user is triggered to provide labels when the state is changing or when an estimation is incorrect are overall better

performing than when an active learning strategy queries the user. The difference in performance is especially noticeable when there is a lower amount of labelled data. The results gives an indication that giving the user a more proactive role in labelling, unlike typical active learning, can be beneficial on performance.

In future work we plan to further validate our conclusions and further explore the taxonomy through experiments. Our aim is to test the taxonomy on a variety of datasets and other machine learning methods. We also aim to explore machine teaching that is triggered by user related factors further. We plan to test the robustness of interactive online machine learning strategies if the user does not always respond with a correct label.

Bibliography

- [1] B. Settles, “Active learning literature survey,” University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.
- [2] X. Zhu, “Machine teaching: An inverse problem to machine learning and an approach toward optimal education,” in Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
- [3] X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty, “An overview of machine teaching,” arXiv preprint arXiv:1801.05927, 2018.
- [4] Y. Fu, X. Zhu, and B. Li, “A survey on instance selection for active learning,” *Knowledge and information systems*, vol. 35, no. 2, pp. 249–283, 2013.
- [5] T. Miu, P. Missier, and T. Plötz, “Bootstrapping personalised human activity recognition models using online active learning,” in 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, IEEE, 2015, pp. 1138–1147.
- [6] E. Lughofer, “Single-pass active learning with conflict and ignorance,” *Evolving Systems*, vol. 3, no. 4, pp. 251–271, 2012.
- [7] Y. Chen, A. Singla, O. Mac Aodha, P. Perona, and Y. Yue, “Understanding the role of adaptivity in machine teaching: The case of version space learners,” in *Advances in Neural Information Processing Systems*, 2018, pp. 1476–1486.

- [8] E. Lughofer, "On-line active learning: A new paradigm to improve practical useability of data stream modeling methods," *Information Sciences*, vol. 415, pp. 356–376, 2017.
- [9] D. Kottke, G. Kremlpl, and M. Spiliopoulou, "Probabilistic active learning in datastreams," in *International Symposium on Intelligent Data Analysis*, Springer, 2015, pp. 145–157.
- [10] I. Žliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39, 2013.
- [11] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine learning*, vol. 15, no. 2, pp. 201–221, 1994.
- [12] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.
- [13] O. Banos, R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez, and C. Villalonga, "Mhealthdroid: A novel framework for agile development of mobile health applications," in *International workshop on ambient assisted living*, Springer, 2014, pp. 91–98.
- [14] O. Banos, C. Villalonga, R. Garcia, A. Saez, M. Damas, J. A. Holgado-Terriza, S. Lee, H. Pomares, and I. Rojas, "Design, implementation and validation of a novel open framework for agile development of mobile health applications," *Biomedical engineering online*, vol. 14, no. 2, S6, 2015.
- [15] B. Krawczyk, "Active and adaptive ensemble learning for online activity recognition from data streams," *Knowledge-Based Systems*, vol. 138, pp. 69–78, 2017.
- [16] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.
- [17] Z. A. Khan and A. Samad, "A study of machine learning in wireless sensor network," *Int. J. Comput. Netw. Appl*, vol. 4, pp. 105–112, 2017.

- [18] B. Shickel and P. Rashidi, “Art: An availability-aware active learning framework for data streams,” in The Twenty-Ninth International Flairs Conference, 2016.

PAPER IV: The Effects of Reluctant and Fallible Users in Interactive Online Machine Learning

Agnes Tegen, Paul Davidsson and Jan A. Persson

Abstract. In interactive machine learning it is important to select the most informative data instances to label in order to minimize the effort of the human user. There are basically two categories of interactive machine learning. In the first category, active learning, it is the computational learner that selects which data to be labelled by the human user, whereas in the second one, machine teaching, the selection is done by the human teacher. It is often assumed that the human user is a perfect oracle, i.e., a label will always be provided in accordance with the interactive learning strategy and that this label will always be correct. In real-world scenarios however, these assumptions typically do not hold. In this work, we investigate how the reliability of the user providing labels affects the performance of online machine learning. Specifically, we study reluctance, i.e., to what extent the user does not provide labels in accordance with the strategy, and fallibility, i.e., to what extent the provided labels are incorrect, in experiments on a benchmark dataset as well as a synthetically created dataset. By varying the degree of reluctance and fallibility of the user, the robustness of the different interactive learning strategies and machine learning methods is explored.

Submitted to conference.

1. Introduction

Active learning [1] and machine teaching [2], [3] are two different categories of interactive machine learning strategies that can be used to decrease the amount of labelled data needed to train a machine learning model, while still preserving a high performance. Labelling data is often costly and demands a lot of work from a human user that is meant to provide the labels. Moreover, in some cases it might not be feasible to label all instances. Instead a smaller selection of the instances are chosen for labelling, where the size of the selection is decided by a labelling budget. The aim of the interactive learning strategy is to pick the instances that will provide most information to the machine learning model.

Generally in interactive machine learning the assumptions of the user are that they will always provide a label when queried by an active learning strategy or in accordance with an machine teaching strategy. Furthermore, it is typically assumed that the label provided by the user always is correct. In some settings this can be reasonable assumptions, for instance a medical doctor labelling patient data within their expertise, but in many scenarios they do not hold. While the intent of the assumptions might be to create simplifications in experiments based on complex real-world settings, they may result in conclusions that are not valid. For instance, in an idealised setting, where the user always responds with a correct label, one approach might give the best performance compared to an alternative approach. This does not necessarily mean that the approach will still be the highest performing one if not all labels are correct. If the idealised setting is a simplification of the real setting and the user sometimes does provide an incorrect label, the alternative approach might be the better choice.

Interactive online learning is a special case of interactive learning, where the data arrives in a streaming fashion and each data instance can only be processed by the interactive learning strategy when it arrives. Thus, a decision has to be made at each point in time whether a label should be provided or not for the current data instance. The decision is based on the employed interactive learning strategy.

In this work we explore how the reliability of a user providing labels affects the performance of online machine learning. We look at the aspects of reluctance, how probable is it that a user will not provide a label in accordance with a given interactive learning strategy, and fallibility, how probable is it that the label provided by a user is incorrect. By varying the degree of reluctance and fallibility of the user we study how this influence the performance of different interactive online machine learning strategies.

2. Related work

In most work on interactive machine learning, the assumptions are that the user will always provide a correct label when queried by an active teaching strategy or in accordance with a machine learning strategy. Furthermore, when the assumptions are made, it is rarely discussed whether they are realistic for the given scenario. Earlier work on interactive learning often focus on the user experience or interaction design, e.g. [4], [5], and not as much on how relaxing the assumptions on the oracle can affect performance of the learning model. There are however previous work that explore settings where these assumptions do not hold. In work focusing on active learning this type of user is often referred to as an imperfect oracle. The term is in contrast to the standard definition of oracle in active learning, which is always assumed to respond to a query with a correct label.

Miu et al. present an online active learning framework [6]. Annotations provided by a user is collected in real-time and used for Human Activity Recognition tasks. Apart from testing the proposed framework on benchmark datasets, it was also tested in user studies, by implementing it in a mobile app through which participants could provide labels. In the user study, the replies from the user were simulated to be incorrect 10 % of the time.

Shickel et al. also introduces a framework for active learning in an online setting with multiple imperfect oracles [7]. Generally in active learning, the oracle is always assumed to be available, but in a real-world setting this might not be the case. The framework presented can query multiple different oracles, based on when they are available, which is useful for instance when generating data from crowdsourcing. While different active learning strategies might work for the framework, the strategy used in the work is active learning triggered by uncertainty.

The effects of an imperfect oracle are explored by Donmez et al., both with regard to not always being correct and to not always being available [8]. To obtain labels, an active learning strategy based on uncertainty is used in the experiments. Unlike to the typical active learning setting, the oracle can be both fallible (i.e. provide incorrect labels) and reluctant (i.e. might not always respond when queried). The scenario discussed in the work is for batch learning however, and not streaming data.

Machine teaching where the user providing labels can have varying degree of reliability is an area that needs further investigation, especially for settings with streaming data. In our work we take a step towards exploring how the reliability of a user affects performance of interactive online machine learning, including both machine teaching and active learning strategies.

3. Experimental setup

The aim of the experiments is to explore how varying reliability of a user providing labels affects performance of different online machine learning methods and different interactive learning strategies. The online learning setting means that the data arrives in a streaming fashion, where each instance is presented and processed once by the learning algorithm. This can be compared to batch setting where all data instances typically are available at the same time.

In the experiments we have a cold start scenario, which means that there is no labelled data for the machine learning to train on at the start of the data stream. Labelled data has to be collected gradually over time to incrementally train the machine learning model.

Streaming data means that the total amount of data is not necessarily known and might even be infinite. This creates more issues that has to be taken into account for the interactive learning strategies and the machine learning methods. For instance, the labelled data that is gradually collected cannot be stored indefinitely, as the amount of labelled data theoretically could approach infinity. To counteract this, there is a limit of how many labelled data instances are stored for each class. If the maximum limit is reached for a given class and a new instance with the same label arrives, the oldest one from the collection is discarded.

The evaluation was done in a test-then-train fashion [9], i.e. where the model first attempts to estimate the incoming data instance and then, if a label is provided, uses the new labelled instance to incrementally train the model. The result thus becomes an accumulative accuracy that showcases how performance of the model has been over time. The results displayed in the next section are all average values of several runs of each experiments. How a run of the experiments was constructed is described in more detail in 3.4.

For all the experiments the following setup parameters are specified: level of reluctance and level of fallibility of the user, machine learning method, interactive learning strategy and dataset used. The individual parts of the setup of the experiments are described further below.

3.1. Machine Learning methods

Three separate machine learning methods were implemented in the experiments with the aim to study if there was a discernible difference in the effect of a fallible or reluctant user. The methods were chosen to be well-known off-the-shelf machine learning methods but also suitable for the setting at hand, i.e. online learning with a cold start scenario. Even though the experiments presented here are from simulations done on previously recorded or created datasets, the intention is that the experiment should be able to run in real-time, which means that the complexity of the machine learning method also has to be considered. The three

machine learning methods tested, Naïve Bayes classifier, Support Vector Machine and k-Nearest Neighbor, are described further below.

3.1.1. Naïve Bayes classifier

Naïve Bayes classifier was included in the experiments because it works well for online learning and has low computational complexity [10]. It also needs a relatively small amount of training data before it can start to produce estimations, which makes it suitable for a cold start scenario.

3.1.2. Support Vector Machine

Support Vector Machines (SVMs) are effective in dealing with high-dimensional data, which is of importance in many settings with streaming data, and have efficient memory usage [11]. SVMs aim to find hyperplanes that divides the classes with a margin that is maximized. As the number of fallible instances increases this will become increasingly difficult and the method will spend more time trying to optimize the classifier. To counteract that training time becomes a concern, a maximum number of iterations to optimize the hyperplanes is set to 1000. A polynomial kernel is used in the experiments, based on initial testing. These results are not included in this work due to space restrictions.

3.1.3. k-Nearest Neighbor

k-Nearest Neighbor is a fitting classifier for our scenario because of its simplicity, it is suitable for online learning and the computational work needed can be limited through the amount of data temporarily stored [12]. The method looks at the labels of the k nearest, previously collected, data instances to classify a new data instance. This means that only k instances needs to be collected before the method can be used, which is good for a cold start scenario. The value of k was set to 3, based on initial experiments that are not included in this work due to space restrictions.

3.2. Interactive Learning strategies

Two different interactive learning strategies were implemented and compared in the experiments. One of the strategies is an active learning strategy, where the learning model is deciding when to query the user for a label, the other is a machine teaching strategy, where the user decides which instances to provide labels for.

All type of interactive learning has to accommodate for a labelling budget. The labelling budget is set beforehand and decides how big portion of the total amount of data can be labelled by the user. In a batch setting, the use of the labelling budget is straight-forward. If the labelling budget is 10% for example, the 10% of the data that is considered most

informative according to the given strategy is chosen to be labelled. In an setting with streaming data only one data instance is processed at a time. While there is knowledge of the data instances that have already been processed, the instances that will appear in the future are unknown. The choice of whether to ask for or provide a label or not for a given instance has to be done as soon as it appears in the data stream. Since the total number of data instances is unknown at the start of processing the streaming data, the labelling budget can not be calculated the same way as it is done in a batch setting. Instead a sliding window containing information on which of the latest processed instances the user has provided a label for. This limited window is used to determine if there currently is enough labelling budget to pose a query or provide a label. The window size is set to 200 instances in the experiments.

3.2.1. Active learning triggered by uncertainty

The active learning strategy included in the experiments is triggered by uncertainty of the machine learning methods estimation. Along with each estimation produced by the machine learning method is also a measurement of how certain the model is of its own estimation. The active learning strategy compares the produced uncertainty measurement to a set threshold. If the measurement is below the threshold, the estimation is considered uncertain and a query is posed to the user to provide a label given that there is enough labelling budget. This is the most common type of active learning used and is sometimes referred to as Uncertainty sampling [1]. To implement this strategy, the measurement of uncertainty needs to be defined and this is dependent on the machine learning method used. As three different machine learning methods are employed in our experiments, each one needs their own uncertainty measurement.

The Naïve Bayes classifier produces a probability for each class and then picks the class with the highest probability for its estimation. The probability of the chosen class is then compared to a threshold. An initial value is set for the threshold, but the value can be lowered or increased over time, depending on whether a query is made or not. If many queries are made the threshold is gradually lowered, as it might indicate that the threshold is too high, or the opposite if few queries are made. The implementation is based on the Variable Uncertainty Strategy presented by Žliobaite et al. [13].

The measurement of uncertainty user for SVM is the distance from the new data instance to be measured to the hyperplanes. If the distance is short it means that the new instance is in close proximity to another class and thus might have a higher probability belonging to the other class compared to an instance further away. The distance is compared to a threshold that, like in the case of Naïve Bayes can be altered depending

on how many queries are made.

For the k-Nearest Neighbor the measurement is based on how many of the k instances nearest to the new instance, i.e. the instances deciding which class to estimate, have the same label. If more than two-thirds of the instances have the same label the estimation is considered certain otherwise not. The strategies used for SVM and k-Nearest Neighbor are further described by Pohl et al. [14].

3.2.2. *Machine teaching triggered by error*

In machine teaching it is the user that employs a strategy of when to provide labels to the learning algorithm. In the strategy included in the experiments the user is aware of the current estimation produced by the model and whenever this estimation is incorrect the user is triggered to provide a label, given that there is enough labelling budget to do so. In this way the user can aid the model by correcting it when it makes a mistake.

3.3. Simulation of reliability in the user

In the experiments the aim was to explore how a varying degree of reliability in the user providing labels affects the results. How is performance affected if the assumptions on the user to always provide a correct label in accordance with the given strategy are relaxed? The two aspects of reliability studied in the experiments were reluctance and fallibility, both are further explained below. By simulating reluctance and fallibility in the user, the level of reliability could be controlled in each experiment.

3.3.1. *Reluctance*

A reluctant user is a user that does not always provide a label in accordance with the given learning strategy. In the case of the active learning strategy, the user will not always reply to a query and in the case of the machine teaching strategy, the user will sometimes not provide a label, even though the estimation is incorrect and there is enough labelling budget. In a real-world scenario the reluctant behavior can be explained by a user that is for instance distracted by something else, unwilling to provide labels or uncertain of which label to provide. The level of reluctance is varied between 0% and 50% in the experiments, during which fallibility is kept at 0%. The level of reluctance informs how big portion of the queries posed that the user will not respond to. For each new query posed, a random number between 0 and 1 is generated and if the generated number is lower than the level of reluctance, the user will not reply.

Since the window used to calculate the current labelling expenses only includes the cases when a label has been received, the expenses are

not increased when a query does not get a reply or when a user does not provide a label. Theoretically this could mean that for the very next instance the active learning strategy could pose a new query or the user could provide a label in the case of machine teaching. This would not be very realistic, as a user that for instance is distracted at one point in time will likely still be distracted the moment afterwards as well. To include this in the experiments, if a label that should have been provided was not due to reluctance, the algorithm had to wait the length of the labelling window before another data instance could be considered for labelling.

3.3.2. *Fallibility*

A fallible user does always provide a label when queried or triggered, but the label is not always correct. This could for example correspond to a user that does not know or is uncertain about the correct label or that makes a mistake. The experiments are constructed in a similar way to the experiments explained above for a reluctant user. The level of fallibility decides the probability of a label being incorrect. In the experiments the level is varied between 0% and 50%. When an incorrect label is provided, the false label to be attached to the data instance is chosen randomly from all the incorrect labels.

3.4. Datasets

To study the effects of reluctance and fallibility in the user on performance, experiments were performed on two separate datasets. The first is an activity recognition dataset and consists of recordings from a real-world scenario. The second is a synthetically constructed dataset.

3.4.1. *mHealth dataset*

The mHealth dataset consists of recordings of 10 subjects with wearable sensors performing a specific routine of physical exercises [15], [16]. The set of wearable sensors include gyroscope magnetometer, accelerometer and electrocardiogram sensor. Each recording contain data from one subject, resulting in 10 recordings in total. The specific routine consists of 12 different physical exercises that the subject is meant to perform in accordance with a specific sequence. The routine is constructed so that one exercise follows the other, but is never repeated. Because of the test-then-train evaluation, the different classes that are to be estimated, i.e. the physical exercises in this dataset, should appear more than once to result in any proper conclusion with regards to performance evaluation. To create a sequence where all exercises appear more than once, the recordings are all put after one another to create one longer data sequence. The order of which the different recordings are put is randomly

Figure 6.1.
A visualisation of the distributions of the classes of the synthetic dataset.

generated for each run. The result produced is the average of 20 separate runs.

3.4.2. Synthetic dataset

The synthetically generated dataset contains 50000 data instances in total. The dataset has two features and five different classes with 10000 instances belonging to each class. For each class a mean value was created for the two features. The instances was then generated by sampling from a 2D normal distribution with the given mean value of the given class and a set standard deviation. In Fig. 6.1 a visualisation of the dataset is displayed. For each run of the experiments using the synthetic dataset an order of all the data instances has to be established. This was done by first randomly choosing one of the classes, then draw a random sample from a normal distribution to decide how many samples of this class should be in the interval. If for instance 20 instances of class A is set for one interval, 20 random data instances belonging to class A are chosen and put after one another in the sequence. One interval is put after another until all the data instances are arranged in the sequence. The ordering of the instances is redone for each run, but the data instances themselves are the same for all of them. The result presented is the mean value of 100 separate runs.

3.5. Limitations

There are several aspects of the typical assumptions made in interactive learning that does not always hold in real-world scenarios. In this work we have chosen to focus on two aspects of reliability of the user, reluctance and fallibility, but there are other that could be of interest to study, depending on the application.

In the experiments it is assumed that there is one user, or multiple users with the same levels of reliability, providing labels. In certain settings however, there might be multiple users with different characteristics, all with the possibility to provide labels at least at some point in time. An example where this is highly relevant is in cases when crowdsourcing is used to collect labelled data.

Another assumption made in the experiments that could be challenged is that the cost of providing a label is uniform for all possible labels. In some settings this might not be the case. For instance, if a scenario has classes that are similar, there might be data instances that need a more thorough examination to find out which class they belong to, while other instances can be classified by the user at a glance. If the cost of labelling varies, this could also connect to reluctance and fallibility of the user. A more difficult, or costly, label can lead to a higher probability that the user does not provide any label. There could also be a case where there is an option to have more thorough and costly labelling by the user, resulting in a lower fallibility, or a quicker and cheaper labelling, but with a higher risk of being fallible. Depending on the application setting, one approach might be preferable over the other.

One relevant issue when discussing learning from streaming data is concept drift, which means that the statistical properties of the streaming data changes over time. It is a phenomenon that is present in many streaming data settings and there exists works that discusses it in more detail [9]. Concept drift is not the main focus of this work, but is handled by continuously updating the machine learning model with new incoming data instances and discarding old ones.

4. Results

In Figs. 6.2 and 6.3 the results from experiments on the mHealth dataset when varying the degree of reluctance and fallibility of the user can be seen. Figs. 6.4 and 6.5 show results from the corresponding experiments on the synthetic dataset. Each plotted line in the figures represent the average performance of a run with the given setup in terms of machine learning method, dataset, reliability and fallibility settings, etc. The performance is displayed as accumulated accuracy over the number of samples that have been estimated so far. This means that when the number of samples is lower it represents performance early in the average

run, i.e. when fewer estimations have been done and fewer labelled data instances have been gathered.

5. Discussion

From the results displayed in Figs. 6.2, 6.3, 6.4 and 6.5 it is clear that performance gets worse with an increased level of reluctance or fallibility of the user in all of the experiments. How big the decrease in performance is depends on the dataset, interactive learning strategy and machine learning method.

In the experiments on the mHealth dataset Naïve Bayes classifier and the machine teaching strategy triggered by error has the highest performance. If the level of reluctance is increased (Fig. 6.2) or if the level of fallibility is increased (Fig. 6.3) the combination of Naïve Bayes and the machine teaching strategy is still giving the best performance. In Fig. 6.2a, showing the results for this combination with an increasing reluctance level however, the trend still seems to be downwards towards the end of all samples. This can be compared to Naïve Bayes with active learning triggered by uncertainty, Fig. 6.2b, where the performance overall is lower, but seems to stabilize after a while. In the experiments with a varying degree of fallibility, SVM and k-Nearest Neighbor appear to be more affected compared to Naïve Bayes classifier. As the level of fallibility increases, the performance of SVM and k-Nearest Neighbor drops faster, especially compared to Naïve Bayes in combination with the active learning strategy.

Figs. 6.4 and 6.5 contain the results from the experiments on the synthetic dataset. In the experiments where the reluctance of the user is varied, Fig. 6.4, the biggest difference in performance can be found early, when the number of samples is low. The main reason for this is likely that a higher level of reluctance in the user leads to a longer time before enough labelled data instances have been gathered to result in a performance on the same level as a user with 0% reluctance to provide labels. A higher reluctance in the user results in a delay of a model with a performance that is representative of the given machine learning method. Towards the end of the average run, the performance approaches each other. When looking at the final accumulated accuracy of Fig. 6.4, Naïve Bayes classifier and k-Nearest Neighbor performs better than SVM. The figure also shows that at the start the machine teaching strategy performs better than the active learning strategy. This is consistent over all machine learning methods tested, but the difference decreases as the level of reluctance increases. For SVM at the highest level of reluctance, there is no significant difference at the start between the active learning strategy and the machine teaching strategy. Furthermore, after a while the active learning strategy outperforms the machine teaching strategy.

(a) Machine teaching, NB

(b) Active learning, NB

(c) Machine teaching, SVM

(d) Active learning, SVM

(e) Machine teaching, kNN

(f) Active learning, kNN

Figure 6.2.

The results from the experiments on the mHealth dataset when the level of reluctance is varied. The left column (a, c and e) displays the result for the active learning strategy triggered by uncertainty and the right column (b, d and f) for the machine teaching strategy triggered by error for Naïve Bayes (NB), Support Vector Machine (SVM) and k-Nearest Neighbor (kNN) respectively.

Figure 6.3.

The results from the experiments on the *mHealth* dataset when the level of fallibility is varied. The left column (a, c and e) displays the result for the active learning strategy triggered by uncertainty and the right column (b, d and f) for the machine teaching strategy triggered by error for Naïve Bayes (NB), Support Vector Machine (SVM) and *k*-Nearest Neighbor (kNN) respectively.

(a) Machine teaching, NB

(b) Active learning, NB

(c) Machine teaching, SVM

(d) Active learning, SVM

(e) Machine teaching, kNN

(f) Active learning, kNN

Figure 6.4.

The results from the experiments on the synthetic dataset when the level of reluctance is varied. The left column (a, c and e) displays the result for the active learning strategy triggered by uncertainty and the right column (b, d and f) for the machine teaching strategy triggered by error for Naïve Bayes (NB), Support Vector Machine (SVM) and k-Nearest Neighbor (kNN) respectively.

Figure 6.5.

The results from the experiments on the synthetic dataset when the level of falibility is varied. The left column (a, c and e) displays the result for the active learning strategy triggered by uncertainty and the right column (b, d and f) for the machine teaching strategy triggered by error for Naïve Bayes (NB), Support Vector Machine (SVM) and k-Nearest Neighbor (kNN) respectively.

In the experiments where the effects of fallibility of the user was tested on the synthetic dataset, displayed in Fig. 6.5, the effect that the choice of interactive learning strategy and machine learning method can have on performance is clear. The best performing and most robust combination is also here Naïve Bayes classifier with the machine teaching strategy triggered by error. When the level of fallibility is at 0%, there is not a significant difference between the different interactive learning strategies and machine learning methods. When the fallibility level is higher than 0% however, the difference becomes noticeable. The biggest decrease in performance can be seen in the experiments using SVM, displayed in Fig. 6.5c and Fig. 6.5d. Since SVM tries to optimize the positioning of hyperplanes to as much as possible separates the different classes, the task will get increasingly difficult as the number of incorrect labelled instances increases in the dataset used for training. The steepest drop in performance of these two can be seen in Fig. 6.5d, where the active learning strategy triggered by uncertainty is employed. The measurement of uncertainty implemented for SVM is based on the distance from the new data instance to be tested, to the hyperplanes. If the SVM classifier has trouble positioning the hyperplanes correctly due to faulty training data, the uncertainty measurement which is dependent on this position will also be inadequate. While less extreme, the effect of an increasingly large part of the labelled data used for training being incorrect is visible for the Naïve Bayes classifier and k-Nearest Neighbor as well.

In the experiments exploring the effects of fallibility of the user, a portion of the provided labels are incorrect. The experiments are meant to simulate a user that is not always correct in assessing what label currently is representative of the state. In a real-world scenario, a user that is sometimes incorrect in this assessment might not always recognize when the estimation of the machine learning method is incorrect either. This is not included in the experiments where the machine teaching strategy is employed and might therefore not portray the entire spectrum of possible effects of a fallible user.

Another factor of the experimental setup that might affect the results is the choice of which data instances that are affected by fallibility or reluctance. As explained in 3.3, the data instances that are either not provided, in the case of a reluctant user, or provided with incorrect labels, in the case of a fallible user, are chosen at random. In certain scenarios it might be reasonable to assume that the probability of all the instances to be chosen are evenly distributed. For instance, if the user is distracted by another task they are performing, they might sometimes, i.e. in a random pattern, miss to provide a label in accordance with the given learning strategy. For a user that is attentive but not as knowledgeable of what the correct label is on the other hand, the probability of which

labels are not provided or given an incorrect label might be correlated to the data instance itself. For example, a data instance belonging to one label, but that is close to the boundary of another might be more difficult for the user to provide a label to than a data instance that is a typical example of the same class. In a real-world scenario the user might have a higher probability to not provide a label or provide an incorrect label if they are uncertain, i.e. if the instance is close to the boundary.

6. Conclusion and Future Work

In this work we explored how the reliability of the user providing labels affects the performance of online machine learning. We also studied the robustness of different interactive learning strategies and different machine learning methods with regards to a user that can be fallible and reluctant respectively. The results show that the choice of interactive learning strategy and machine learning method has an effect on performance in the experiments, where the combination of Naïve Bayes classifier and the machine teaching strategy triggered by error overall resulted the highest performance. This combination is also most robust towards increased levels of fallibility and reluctance of the user. The overall least robust machine learning method was SVM, specifically for a fallible user.

In future work we plan to further validate our conclusions by testing on other datasets and more machine learning methods. We also aim to further explore how varying the level of reliability of a user can affect performance.

Bibliography

- [1] B. Settles, “Active learning literature survey,” University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.
- [2] X. Zhu, “Machine teaching: An inverse problem to machine learning and an approach toward optimal education,” in Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
- [3] X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty, “An overview of machine teaching,” arXiv preprint arXiv:1801.05927, 2018.
- [4] E. Johns, O. Mac Aodha, and G. J. Brostow, “Becoming the expert-interactive multi-class machine teaching,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2616–2624.
- [5] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, “Power to the people: The role of humans in interactive machine learning,” *AI Magazine*, vol. 35, no. 4, pp. 105–120, 2014.
- [6] T. Miu, P. Missier, and T. Plötz, “Bootstrapping personalised human activity recognition models using online active learning,” in 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, IEEE, 2015, pp. 1138–1147.
- [7] B. Shickel and P. Rashidi, “Art: An availability-aware active learning framework for data streams,” in The Twenty-Ninth International Flairs Conference, 2016.

- [8] P. Donmez and J. G. Carbonell, "Proactive learning: Cost-sensitive active learning with multiple imperfect oracles," in *Proceedings of the 17th ACM conference on Information and knowledge management*, ACM, 2008, pp. 619–628.
- [9] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.
- [10] B. Krawczyk, "Active and adaptive ensemble learning for online activity recognition from data streams," *Knowledge-Based Systems*, vol. 138, pp. 69–78, 2017.
- [11] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.
- [12] Z. A. Khan and A. Samad, "A study of machine learning in wireless sensor network," *Int. J. Comput. Netw. Appl*, vol. 4, pp. 105–112, 2017.
- [13] I. Žliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39, 2013.
- [14] D. Pohl, A. Bouchachia, and H. Hellwagner, "Batch-based active learning: Application to social media data for crisis management," *Expert Systems with Applications*, vol. 93, pp. 232–244, 2018.
- [15] O. Banos, R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez, and C. Villalonga, "Mhealthdroid: A novel framework for agile development of mobile health applications," in *International workshop on ambient assisted living*, Springer, 2014, pp. 91–98.
- [16] O. Banos, C. Villalonga, R. Garcia, A. Saez, M. Damas, J. A. Holgado-Terriza, S. Lee, H. Pomares, and I. Rojas, "Design, implementation and validation of a novel open framework for agile development of mobile health applications," *Bio-medical engineering online*, vol. 14, no. 2, S6, 2015.

ISBN 978-91-7877-084-7 (print)

ISBN 978-91-7877-085-4 (pdf)

MALMÖ UNIVERSITY
205 06 MALMÖ, SWEDEN
WWW.MAU.SE