UNIVERSITÀ DI TRENTO

Department of Information Engineering and Computer Science

Doctoral Thesis

Human-Machine Alignment for Context Recognition in the Wild

Author:
Andrea Bontempelli

Supervisors:
Prof. Fausto Giunchiglia
Prof. Andrea Passerini

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

April 19, 2024

Contents

List of Figures

List of Tables

Acronyms

LTM	Long Term Memory
SM	Sensory Memory
WM	Working Memory
CINCER	Contrastive and InflueNt CounterExample stRategy
ISGP	Incremental Skeptical Gaussian Processes
TRCKD	TRaCking Knowledge Drift
KD	knowledge drift
PCR	personal context recognition
SKL	Skeptical Learning

Abstract

The premise for AI systems like personal assistants to provide guidance and suggestions to an end-user is to understand, at any moment in time, the personal context that the user is in. The context – where the user is, what she is doing and with whom – allows the machine to represent the world in user's terms. The context is not directly accessible to the machine, and thus, it must be inferred from a stream of sensor readings generated by smart wearables such as smartphones and smartwatches. We refer to this task as context recognition, a supervised learning task with the constraints that the target classes are user-specific, namely it encodes the user perspective (e.g., this user's home is not another user's home). In addition, given that the machine follows the user over time, it faces changes in the world and in the user (e.g., the user visits new places), generating changes in the vocabulary and in the data. Thus, the fundamental issues in this out-of-the-lab setting are *(i)* the unreliability of the supervision due to noisy labels that fool the machine and changes in the user vocabulary, and *(ii)* data shifts that lead the machine to systematic errors. To perform robust context prediction in this real-world scenario, the machine must handle the egocentric nature of the context, adapt to the changing world and user, and maintain a bidirectional interaction with the user to ensure the user-machine alignment of world representations. To this end, the machine must learn incrementally on the input stream of sensor readings and user supervision. Based on the above considerations, the contributions of this work are the following. *(i)* We introduce interactive classification in the wild, which is characterized by noisy labels, and the number of classes grows over time. Then, we present knowledge drift (KD), a special form of concept drift, occurring due to world and user changes. *(ii)* In order to tackle this task, we develop simple and robust ML methods: ISGP is an interactive learning approach to clean the example by asking the user to revise the label when sufficiently confident that the supervision is noisy, CINCER is an approach that selects training examples to explain and support the skepticism of the machine and allows fixing noisy examples that elude the cleaning step. TRCKD is a novel approach that tackles KD and combines automatic drift detection and knowledge-aware adaptation with interactive refinement of the machine understanding of the detected KD. *(iii)* We showcase the advantages of each of these methods in empirical evaluations on controlled synthetic and real-world data sets. *(iv)* We design a flexible and modular architecture that combines the methods above to support context recognition in the wild. *(v)* We evaluate ISGP with real users in a concrete social science use case.

Keywords: interactive machine learning, incremental learning, data streams, human-in-the-loop, concept drift, context recognition

Acknowledgements

The PhD is a long journey made up of a sequence of research and personal challenges. By looking back to the time I started, I realize how much I have learned and the improvement I made in addressing research and technical challenges. I would like to thank my advisors, Prof. Fausto Giunchiglia and Prof. Andrea Passerini, for giving me the opportunity to make this journey. I am very grateful to Stefano Teso for the valuable feedback and useful lessons he gave me during my doctoral journey. Their guidance has been fundamental to shaping how I think as a scientist and as a person. A significant portion of the knowledge I obtained during my doctoral studies arose from discussions with them, and they contributed significantly to various chapters of this thesis. I thank the reviewers Prof. Albert Bifet and Prof. Frank van Harmelen for their valuable feedback, and the committee members Prof. Frank van Harmelen, Prof. Vincenzo D'Andrea and Dr. Riccardo Guidotti for being part of this journey. The research period abroad at IDIAP has been possible thanks to Prof. Daniel Gatica-Perez and the social computing group. The experiment with real users would not have been possible without the contribution of Matteo Busso to the experiment design and research proposal for the Research Ethics Committee. I also thank Marcelo Rodas Britez, Leonardo Malcotti, and Ivan Kayongo for adapting the iLog platform to the experiment requirements. I would like to thank Alessio Zamboni, Simone Bocca and all the Knowdive members for the insightful discussions and support, and for building an inspiring and welcoming working environment. Last but not least, I want to give special thanks to my family and my friends outside academia for their immense and constant support.

The work compiled in this thesis has been supported by:

"DELPhi - DiscovEring Life Patterns" project funded by the MIUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) 2017 – DD n. 1062 del 31.05.2019.

"WeNet – The Internet of us' project funded by European Union's Horizon 2020 FET Proactive under grant agreement No 823783.

1 Introduction

Contents

This chapter provides an overview of the work of this thesis. We start by describing the motivation and the context that guided the development of the algorithms and architectures presented in the next chapters. Then, the thesis's contributions in tackling the relevant problems are outlined. Finally, we list the publications on which the thesis is based and also a brief summary of each chapter.

1.1 Motivation

Understanding the activity the person is performing and the place where he or she is fundamental for the machine to provide services that support timely and correctly the person in his or her everyday life [168]. The high penetration of smartphones and smartwatches in the life of the person is a privileged observation point of view. Indeed, these personal devices are always with the person and allow one to observe the world through the person's point of view. Let us introduce a motivating example. Ann's phone detects that she is driving her car, so it decides to mute incoming notifications to avoid distracting her. This simple but non-trivial example is common to many users and highlights the knowledge the machine needs to acquire about its user and the surrounding world to recognize this scenario. First, the machine needs to detect that the user is moving with a vehicle by sensing the world through sensor data like accelerometer and GPS coordinates. Second, the machine must disambiguate if Ann is on a car or bus. The machine infers from past data the habits of going to her workplace every day at 8 am. Moreover, her smartphone is connected via Bluetooth to her, so the machine infers that Ann is driving her car to go to work. Ann decides to start taking the bus in the morning to avoid getting stuck in the traffic. The machine recognizes that during the last week, Ann moved through a different route, but it is not able to know whether this is due to roadworks or because she is using a different means of transportation. Therefore, it asks Ann to disambiguate, and based on her answer, then the machine enables the notification during the morning trip.

The scenario above is an example of an AI working in lifelong symbiosis with a human and interacting with her or him via smart devices like smartphones, smartwatches and medical devices. Possible use cases are personal assistants, smart environments and medical applications. Existing personal assistants focus on a single dimension of the user's life, such as calendar management [117], time and task management [118], education and learning [157] and information access [81], and they can leverage Natural Language Processing [7] to interact with the user. The key requirement for a Human-AI symbiosis is to recognize the *personal situational context* at any moment in time [20]. In this work, we use the notion of context introduced in [69] as *"a theory of the world which encodes an individual's subjective perspective about it"*. Thus, the context models a subjective view, which is a partial view of the world representing a set of facts locally relevant to the current activity [24]. The personal situational context, for brevity context, describes the part of the world in which the person operates, and it is described in terms of space (locations), social context (other persons), object environment (surrounding objects) and functional relations (expected behaviour of persons and objects, and the performed activities). The sequence of the context at any point in time is a stream storing the past contexts.

1.2 Problem

The personal context is not directly accessible to the machine, which infers it from other sources such as wearable devices, knowledge graphs and the reference person. We refer to the task of recognizing the context of the person at any point of time as *personal context recognition* (PCR). From one point of view, PCR can be seen as a generalization of activity recognition [33] in which the task is to learn the mapping between the sensor data of personal devices and a single dimension of the context, namely the activity. The learned mapping is then used to recognize the activity in unseen situations. Two aspects characterize the personal context. First, its dimensions are correlated, thus making the context inherently structured [194]. For instance, the person's activity is strongly influenced by his/her location. Second, it is constrained by the context knowledge graph, which encodes the entities and their relations (*cfr.* 2.1). Given these desiderata, PCR can appear as a standard but highly non-trivial ML task under KR constraints. However, this task becomes challenging when moving to a lifelong setting. To deliver high recognition performance in this setting, the machine is required to be robust to known unknowns (uncertain aspects modelled by the machine) and unknown unknowns (unmodeled aspects of the world) [46]. In PCR, this requires the machine to be robust to changes *in the world* and *in the user*.

The user's description of the context changes. The personal context is subjective, and thus, the machine obtains the user's description of the context to learn how to recognize unseen context. This description, given that PCR is a supervised learning task, corresponds to the supervision given by the user to the machine. However, over time, the user describes the same situation in different ways, even if the world has not changed. For instance, the same person can be referred to as a friend, girl or Ann based on what is relevant in the current context. At the same time, this supervision is frequently unreliable due to mistakes, inattention and other response bias [188, 64] that affect the context recognition performance.

The world itself changes. Over time, the machine knowledge about the world evolves because both the world actually changes and new information becomes available. At the same time, the user changes her understanding of the world. For instance, if the user moves to another city, the structure of her personal context changes. From a statistical perspective, these changes are forms of *concept drifts* [166, 63] affecting the distribution of sensor observation and personal context supervision. However, in the lifelong context recognition task, the drift is more complex as it alters the context knowledge graph. Indeed, the entities and their relations become obsolete and updated over time, e.g., the set of friends or the apartment where the user is living.

Given the problems above, AI must ensure that its understanding of what is happening is aligned with that of its user; otherwise, the misrecognized context may lead the system to useless and harmful outputs [41].

1.3 Solution

To deliver high-quality personal context recognition performance, the goal is to ensure alignment between machine perception and human description over the lifespan of the AI. Thus, the key challenges to consider are:

- *egocentric* nature of user's context, crucial to provide useful suggestion;
- *lifelong* context recognition that is robust to changes in the world and to their users, which are impossible to anticipate;

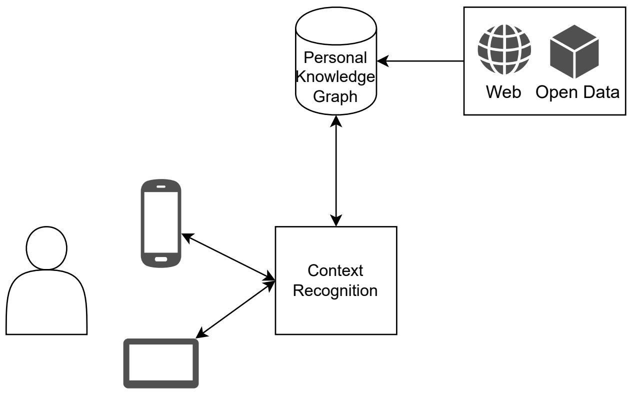


Figure 1.1: Simplified overview of the proposed solution. The input of the context recognition solution is the data generated by the user's devices and the knowledge about the world and the user. The bidirectional interaction between the user and the machine via the devices.

- continual *bidirectional interaction* between the user and the machine to ensure the alignment of their world representation. The direction from the user to the machine is what is usually done in Knowledge Representation and Machine Learning. The other direction has recently emerged under the term Explainable AI (XAI) [82, 116], considered as a major requirement to achieve human-centric and trustworthy AI.

Therefore, given the challenges above, the machine needs to be:

- *incremental*, to update the learned model to the new information as it becomes available over time;
- able to *detect* changes and *adapt* by *interacting* with the reference user;
- exploit *prior knowledge* stored in previous contexts.

Here, it is fundamental that the machine's understanding of what is happening is completely aligned with that of its reference users. This understanding is in the mind of the reference user, who is the AI's ultimate source of information. Hence, the AI must interact with the user to get information when something new and unforeseen happens.

Specifically, we are interested in predicting the user's context at any moment in time from the stream of data coming user's smartphone. Figure 1.1 shows an simplified overview of the scenario. The input of the context recognition solution is the data generated by the user's devices and the knowledge about the world and the user, structured as a knowledge graph. This knowledge graph is filled with the knowledge just learned by the model and, at the same time, can also be integrated with data from third parties such as large multilingual resources [70], geographic information and other open data. The PCR machine implements a bidirectional interaction between the user and the machine via the devices. This work focuses on the development of robust machine learning algorithms to tackle context recognition in the wild. Specifically, the contributions of this work are the following. We:

1. Introduce interactive classification in the wild, a novel form of interactive learning in which there is a substantial amount of labelling noise, and new classes are observed over time. In this setting, the user's description of the context is noisy due to mistakes and inattention. To address this task, we present a redesign of Skeptical Learning [188] that leverages exact uncertainty estimates to allocate queries to the user appropriately when suspicious about a label and avoids over-confident models even in the presence of noise;

2. Introduce an explanatory interactive label-cleaning strategy that leverages example-based explanations to identify inconsistencies in the data—as perceived by the model—and enable the annotator to fix them. This strategy builds upon Skeptical Learning by adding explanations. The explanation is a set of counter-examples that explain why the model is suspicious and that are highly informative. This contribution goes in the direction of the bidirectional interaction. Indeed, first, the user provides a label, then the machine explains its suspicion about this label, and finally, the user fixes the inconsistency in the data.

3. Identify *knowledge drift* (KD) as a complex phenomenon that affects the knowledge encoded in the context knowledge graph whenever the world changes. In KD, which we identified as a special kind of concept drift, concepts and their relations can become obsolete or irrelevant, and new ones can be added. To address this task, we design TRCKD, an approach for handling KD that combines automated detection and adaptation with interactive disambiguation and instantiates it on top of kNN-based classifiers by implementing a knowledge-aware adaptation strategy. Here, given the egocentric nature of the personal context, we identify the interaction with the reference user as the only way to disambiguate among the possible forms of knowledge drift. The reason is that different forms of drift have similar footprints on the observed data, which is thus insufficient to adapt the model and its knowledge graph to them properly.

4. Design a comprehensive architecture to solve PCR in the lifelong setting. Through the composition of Skeptical Learning and knowledge drift handling, the architecture addresses the problem of language and knowledge alignment. This compositionality strategy allows the architecture to be sufficiently flexible in terms of input data and reasoning strategy to adapt to specific designer's requirements and, thus to several application domains.

5. Design and execute an experiment with real users to evaluate Skeptical Learning in the wild. The experiment design investigates how Skeptical Learning can help tackling the respondent burden and answer quality problems that affect the longitudinal studies run in social science. In these studies, the researchers are interested in acquiring the context of the participants multiple times per day over a period of a few weeks by asking them multiple questions. The respondent burden causes the participants to leave the data collection or to provide unreliable answers. We also provide a description of the technological stack and the lessons learned. The result highlights that the assumptions made in the lab might be disrupted when going into real-world validation.

1.4 List of publications

This thesis is based on the following publications:

- **Andrea Bontempelli**, Stefano Teso, Fausto Giunchiglia, and Andrea Passerini. 2020. "Learning in the Wild with Incremental Skeptical Gaussian Processes". *The Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI).*

  https://doi.org/10.24963/ijcai.2020/399

- **Andrea Bontempelli**, Fausto Giunchiglia, Andrea Passerini and Stefano Teso. 2022. "Human-in-the-loop handling of knowledge drift". *Data Mining Knowledge Discovery.*

  https://doi.org/10.1007/s10618-022-00845-0

- **Andrea Bontempelli**, Marcelo Rodas Britez, Xiaoyue Li, Haonan Zhao, Luca Erculiani, Stefano Teso, Andrea Passerini, and Fausto Giunchiglia. 2022. "Lifelong Personal Context Recognition". *The First International Workshop on Human-Centered Design of Symbiotic Hybrid Intelligence co-located with The First International Conference on Hybrid Human-Artificial Intelligence (HHAI).*

  https://arxiv.org/abs/2205.10123.

- Stefano Teso, **Andrea Bontempelli**, Fausto Giunchiglia, and Andrea Passerini. 2021. "Interactive label cleaning with example-based explanations". *Advances in Neural Information Processing Systems (NeurIPS).* (SPOTLIGHT PRESENTATION)

  https://proceedings.neurips.cc/paper_files/paper/2021/file/6c349155b122aa8ad5c877007e05f24f-Paper.pdf

- Fausto Giunchiglia, Marcelo Rodas Britez, **Andrea Bontempelli**, and Xiaoyue Li. 2021. “Streaming and Learning the Personal Context”. *The Twelfth International Workshop Modelling and Reasoning in Context (MRC) co-located with IJCAI 2021.*

  https://ceur-ws.org/Vol-2995/paper3.pdf

The author has further contributed to the following publications:

- **Andrea Bontempelli**, Fausto Giunchiglia, Andrea Passerini and Stefano Teso. 2022. “Toward a Unified Framework for Debugging Gray-box Models”. *The AAAI-22 workshop on Interactive Machine Learning.*

  https://arxiv.org/abs/2109.11160

- **Andrea Bontempelli**, Stefano Teso, Katya Tentori, Fausto Giunchiglia and Andrea Passerini. 2023. “Concept-level Debugging of Part-Prototype Networks”. *The Eleventh International Conference on Learning Representations (ICLR).* (NOTABLE-TOP-25%)

  https://openreview.net/pdf?id=oiwXWPDTyNk

- Luca Erculiani, **Andrea Bontempelli**, Andrea Passerini and Fausto Giunchiglia. 2023. “Egocentric Hierarchical Visual Semantics”. *The Second International Conference on Hybrid Human-Machine Intelligence (HHAI).* (BEST WORKING PAPER AWARD)

  https://ebooks.iospress.nl/volumearticle/63343

1.5 Reproducibility statement

The experimental setup and the code of the algorithms presented in this work are publicly available. The code of the method presented in Chapter 3 is available at https://gitlab.com/abonte/incremental-skeptical-gp. The evaluation has been performed on both synthetic and real-world data. The former are published in the repository. The anonymized version the used real-world data sets and more recent data sets with the same sensor data and collection methodology can be requested and downloaded from https://datascientiafoundation.github.io/LivePeople/datasets/. The algorithm of Chapter 4 is published on https://github.com/abonte/cincer. The code of the method of Chapter 5 has been uploaded on https://gitlab.com/abonte/handling-knowledge-drift, and the experiment data sets are released by their respective authors.

1.6 Outline

Each chapter is self-contained and report an introduction, related works and contributions. Given that the work is cross-cutting to different research fields, each chapter provides related works, which allow the reader to position the contribution of the chapter with respect to the specific field to which it refers. The work is divided into two parts. In Part I, we introduce the theory about the proposed algorithms and evaluate each of them in a controlled setting (Chapters 3 to 5). Then, in Part II, we move the discussion towards the real world scenario by proposing a reference architecture (Chapter 6) and evaluating in the wild with real users (Chapter 7). The remainder of this work is organized as follows:

Chapter 2 frames the research problem by providing the definition of user personal context and exposing the challenges of recognizing it in the open world.

Chapter 3 presents skeptical learning as an algorithm that addresses the mislabeling problem that arises when interacting with the user to train the machine to recognize her or his personal context.

Chapter 4 improves skeptical learning algorithm by adding an example-based explanation allowing to clean training example that eluded the skeptical check. This explanation allows the user to observe and fix the reasons behind the model's suspicions.

Chapter 5 depicts the knowledge drift problem that occurs when the machine's knowledge about the world and the user becomes obsolete. The interaction with the user is crucial to align the machine with the occurred change.

Chapter 6 describes the logical architecture that allows the recognition of the personal context. The architecture combines the procedures presented in the previous chapters to learn in the wild and a knowledge representation of the personal context and machine memory that supports the recognition task. The solution is not stick to any specific technology.

Chapter 7 presents the evaluation of skeptical learning in a real-world experiment run with university students.

Chapter 8 summarizes the main contributions, presents the limitations, briefly argues the ethical and societal impacts of this work and depicts future research directions.

2
Background and problem setting

Contents

The goal of this thesis is to define a system that recognizes the user context, which can be exploited by personalized applications and services to help the user in his or her life. Section 2.1 provides an introduction to the personal context, which is crucial to ensure an alignment between the user and the machine , which is one of our motivation examples. Then, we introduce the knowledge graph representation of the context in Section 2.2. The recognition task is formalized in Section 2.3. However, several issues emerge when the task is performed in uncontrolled environments. Thus, Section 2.4 describes the main conditions that a machine has to face when deployed in the wild. First, what the machine learns about the surrounding environment from perception data must be aligned with the user's interpretation of the same space. Second, traditional machine learning approaches assume a closed word, i.e., the training set is given at the beginning, and the data distribution does not change at inference time. However, the world is evolving over time, and thus machine knowledge must be adapted accordingly.

2.1 Personal situational context

Context models the environment in which a person is from his or her point of view. Since the person has a partial view of the world all the time, the context, as defined in [69], "*is a theory of the world which encodes an individual's subjective perspective about it.*" A lot of work has been done to define the context in multiple research areas such as ubiquitous computing, computer science and sociology [51], and to study context-aware application [156, 41].

In our uncontrolled setting, the context model must address the open and changing world, in which is not possible to define *a-priori* the environments and the concepts. Thus, Giunchiglia et al. [71] model the context as the local view centred around the person and is composed of the following dimensions:

- *activity*, the main activity the person on which the context is centred is performing. This is derived from the question "what are you doing?", e.g., studying;
- *location*, the main place where the person is currently, and it is the answer to the question "where are you?" (library);
- *social*, the persons that are part of the current environment and, in this case, the question is "who are you with? (friends);
- *object*, the objects that are around or used by the person "what are you with?" (laptop and book).

Context subjectivity. From the above definitions, it is possible to note that the answers to the four questions are subjective. For instance, consider the following situation. The professor is teaching to the university students in the classroom. This scenario can be modelled differently according to which reference user the context is centred. Regarding the location, the classroom is the workplace for the professor, whereas a student interprets the same place as a study place. The same building can have different functions according to the point of view. In the case of the performed activity, the student is listening, and the professor is writing to the blackboard. The student and the professor have different objects. The former writes with the chalk on the blackboard, and the latter writes with the pen on their notebooks. The subjectivity of the context is encoded by teleologies, which were introduced in [74]. The concepts used to represent the context are organized in object, function and action. Objects represent concrete and finite objects like buildings, persons and cars. Functions represent the expected behaviour with respect to the user. Actions describe how the object changes in time in order to satisfy its functions.

The context is thus represented as a knowledge graph and encodes the prior knowledge about the user and the world. We formalize and detail the subjective representation of the context in the next section.

2.2 Context representation

This section's aim is to briefly explain the schema level and data level of the knowledge graph proposed in previous works by Giunchiglia et al. [75]. Based on these notions, we model the personal context as follows.

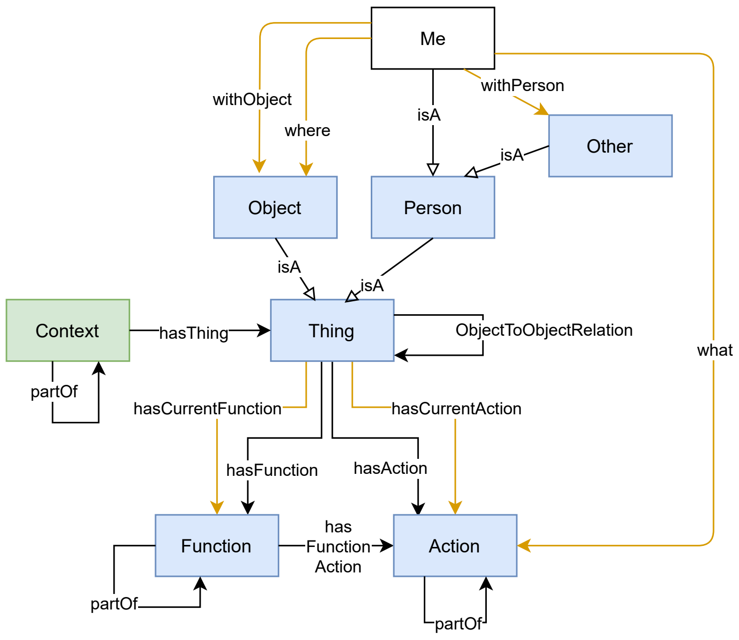


Figure 2.1: Entity Type Graph (*ETG*) is the schema of the personal situational context. The yellow arrows represent the predicates that model the current context of the user.

Entity type graph (*ETG*). The ETG is the schema that defines the relation between the different concepts, namely objects, functions and actions. Figure 2.1 shows the schema of the context, in which nodes are entity types and their links are object properties. The entity type `Me` represents the agent, whose context is modelled. The statements describing the context are:

- `withPerson(Me,Person)` defines the user's social context;
- `where(Me,Object)` identifies the location;
- `what(Me, Action)` represents the actions being performed by the user;
- `hasCurrentAction(Thing,Action)`, determine the actions performed by the objects and person in the current personal context.
- `hasCurrentFunction(Thing,Function)` represent the functions of the persons in the social context and the object defining the location;

Entity Graph (*EG*). The instantiation of the entity types and object properties of *ETG* with a specific value at a certain time t generates an entity graph EG_t. The nodes of this graph are entities and links are object properties. Figure 2.2 is an instantiation example, and the green boxes highlight the different possible contexts.

2.3 Context recognition in static world

The user's mobile devices like smartphones and smartwatches generate a stream of sensor readings (e.g., GPS coordinates, accelerometer and gyroscope). These devices are always with the user and generate a stream of perception data that captures the user's perspective. The machine's task is to recognize the user's context at regular intervals from a stream of data.

We are concerned with a learning task in which an instance $\mathbf{x} \in \mathcal{R}^d$ is associated with a set of concepts, and their relations are organized as a knowledge graph $K = (C, R)$, in which $C = \{c_1, \ldots, c_n\}$ encodes the entity types in the ETG and their instantiation in the EG, and $R = \{r_1, \ldots, r_k\}$ lists the relations between them $R \subseteq C \times C$. The instances are annotated with indicator vector $\mathbf{y} \in \{0, 1\}^{k+n}$, where the i-th element of $\mathbf{y}$, denoted y^i, is 1 if given $\mathbf{x}$, the i-th relation or concept is true and 0 otherwise. Both the true concepts and relations in any given moment in time represent the current personal situational context. The machine observes a stream of examples $\mathbf{z}_t = (\mathbf{x}_t, \mathbf{y}_t)$, for $t = 1, 2, \ldots$ drawn from a ground-truth distribution $P_t(\mathbf{X}, \mathbf{Y})$ which is always consistent with the ground-truth knowledge graph K_t. This means that if a relation s between two concepts i and j does not exist in the ground-truth K_t, then the probability $P_t(\mathbf{X}, \mathbf{y})$ of all $\mathbf{y}$ violating this statement ($y^s = 1$ and $y^i = 1$ and $y^j = 1$) is zero. The classification task is to find a classifier and a corresponding knowledge graph $\hat{K}_t$ that perform well on future instances. In order to output high-quality predictions, the acquired knowledge $\hat{K}_t$ must approximate the unobserved ground truth K_t.

2.4 Context recognition in the wild

In the lifelong context recognition in the wild, the ground-truth labels $\mathbf{y}$ are not available, and the system must ask the user. However, users are unreliable and might provide incorrect labels, as is well known in social sciences, and over time they change how they describe the world. Moreover, the world unpredictably changes, i.e., the unobserved ground-truth knowledge graph K_t and the data distribution can both change over time $t = 1, 2, \ldots$. If not addressed, these factors lead the machine to systematic prediction errors. Adaptation is thus crucial to provide useful services [41]. We detail these issues in the following paragraphs.

The user's description of the context changes. The user's world description changes over time even if the world is not changed. There are several reasons why this happens. First, the user may use words that describe the same object at different levels of granularity. For instance, a professor can alternatively say to be in his office, in a university building or in the city of Trento. Even if all these descriptions are correct, they appear as different concepts to the machine. Second, the user is inattentive or unwilling to report and thus, wrong descriptions are provided. This badly affects the performance of the machine to recognize the context. Third, the world is changed, and the user provides a different description according to this change, e.g., the professor's office moved to Rome.

The knowledge incompleteness. Partial knowledge about the world and the user is built-in into the machine. However, over time, new knowledge about the user and the world becomes available and needs to be integrated into the machine's knowledge. Formally, new concepts and new relations between them are observed and become available over time. If the acquired knowledge graph $\hat{K}$ and the classifier are not updated, the context recognized by the machine is imprecise, obsolete or wrong. In our learning task, new classes appear over time, namely the relations in the knowledge graphs changes.

The world changes. Once acquired, the knowledge graph becomes obsolete due to changes in the world and user behaviour. For example, the user graduates and start working in a company, or the user's favorite shop moves to another street. We called the addition or removal of concepts and relations between them in the knowledge graph K as *knowledge drift* (KD). These changes impact the ability of the classifier to recognize the context correctly.

Changes in the knowledge graph leave footprints on the data stream, i.e., the data distribution changes. Formally, the ground-truth data distribution drifts if there is a point in time t such that $P_t(\mathbf{X}, \mathbf{Y}) \neq P_{t+1}(\mathbf{X}, \mathbf{Y})$ due to a change in K_t. We don't consider the drifts in K that do not impact the data distribution unless the information can be derived from third-party services or provided by the user.

2.4.1 Learning from user

There is a misalignment between the information we can extract from sensor data and the interpretation the user gives to the situation measured by the sensor. This is a known problem, especially in image processing, where it was initially introduced. This is called the semantic gap problem, and Smeulders et al. [145] defines it as "*the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.*" To maintain an alignment between how the user thinks of the word and the machine perception, it is crucial to leverage the annotations and labels from the user herself. In this way, we ensure an egocentric point of view, and the machine can describe what it perceives in the users' own terms [53, 52]. This implies that the machine should recognize context dimension at the instance level (e.g., recognizing my own home) and not only at the class level (e.g., I'm in a building). More practically, it means training the context recognition system on the labels collected from the user.

The interaction with the user also brings some difficulties. First, the user may provide wrong labels due to inattention or unwillingness to respond. Second, the question budget must be limited to avoid the respondent burden, and thus the user may leave the system. This situation implies that supervision is scarce and, a limited number of training labels are obtained.

2.4.2 Learning from noisy data

AI agents learning in real-world scenarios have to consider errors and mistakes in the data. In the setting of this work, the task is to learn user-specific contextual dimension, and thus the user is the only annotator that can provide supervision to the machine. In practice, the user may make mistakes and provides wrong supervision during the learning process. The reasons behind these errors can be the misunderstanding of the task, inattention, cognitive bias, unwillingness to respond and reporting of more socially acceptable behaviour [165, 188].

Example 1 *Ann is studying at home, when she answers "University" to Ann's PA question "Where are you?".*

We assume the collaborative behaviour of the user since it is in his or her interest to have a useful service. The proportion of incorrect annotation can be significant, as shown by an experiment in the same setting as this research [193, Table 2]. Noise can misguide the machine and has a cascading effect on the correctness of the future predictions and on the acquired EG.

2.4.3 Learning from data streams

Personal devices, such as smartphones and smartwatches, are always with the user and are sources of valuable information. These devices allow for monitoring the user's behaviour and observing their environment continuously. Hence, it is possible to capture the user's perspective, e.g., the places where the user is and the performed activity. The machine has some initial knowledge about the word and the user (e.g., the user answers some initialization questions during the installation). The machine receives a continuous stream of sensor data (e.g., GPS coordinates and nearby Bluetooth devices) and input from the user (e.g., answers to questions). Receiving data over time implies that the system continuously incorporates the new data into its knowledge and reacts to changes in the world and the user.

2.4.4 Learning in an open world

Learners in a closed-word setting assume that all the classes they see during testing were encountered in training. When an unknown input belonging to a new class is received, the learner assigns one of the known classes. In an open world, the learner must consider its own limited knowledge of the world and consider the existence of unknown classes. Scheirer et al. [138] identify three categories of classes: *(i) known concepts*, concepts for which there are positive examples in the training set, *(ii) known unknown concepts*, concepts for which there are examples but are not part of known concepts, and *(iii) unknown unknown concepts*, concepts that have no examples in the training set.

Example 2 *Ann goes to the university building for the first time. Ann's PA predicts that Ann is in a "Bar" because it has never observed the "University" concept.*

The *unknown unknown* concepts are regions of the feature space where the predictor misclassifies the examples with high-confidence [46]. Classifiers that use the distance from a decision boundary as a measure of confidence assign classes with high confidence to examples that are far from known training data [23]. This type of error is challenging because the system is unaware of it. As defined by Dietterich [46], robust AI systems need to consider both known unknowns and unknown unknowns. We can summarize the challenge of learning in an open world as the "[...] the ability to recognize when we do not know something, analyze the need to learn about it, and then, when needed, to adapt and learn it"[23].

In addition to observing new concepts, streaming data are commonly affected by other forms of data drifts: *i)* concept drift, the relation between input X and Y changes, *ii)* future drift, the distribution of X changes (e.g., the value range), *iii)* feature evolution, the number of variables increases or decreases. The current model adapts passively or actively to the drift. In the former, the model keeps the most recent data. In the latter, the drift detection triggers the model retraining on a combination of data collected since the drift detection and old data, to retain also previous knowledge.

Given the unboundedness of the data streams, traditional batch learning approaches are impractical due to expensive computation and increasing storage demand. To continuously react to drift and to incorporate new data, the focus must be on incremental learning, as described in the next section.

2.4.5 Learning incrementally

Many applications deal with continuous data streams, such as sensor readings, logs, news and images. As described above, our setting is also characterized by the arrival of new examples over time, and thus the machine learning model and the acquired knowledge need to be updated incrementally. Incremental learning is not limited to handling new training examples. [198] identify three categories of incremental learning tasks where new knowledge must be incorporated into the learned model:

- *example-incremental learning*, when a new example is available (e.g., a new sensor reading)
- *class-incremental learning*, when a new target concept appears in the stream (e.g., "electronics" is a new classification topic of the news)
- *attribute incremental-learning*, when a new attribute of the examples is introduced (e.g., the user enables the GPS, and latitude and longitude are available). New knowledge is incorporated over time but also removed when it becomes obsolete (decremental learning), e.g., the user's face is removed from a door lock with facial recognition, so he or she can no longer access the room.

This setting is challenging for standard batch learning techniques. These approaches assume that the model trained on an initial data set works appropriately, even in the future. This works fine if the examples are drawn from the same distribution as the training set. This assumption is hardly satisfied in real-world scenarios on data streams, and it can be costly or unfeasible to collect a sufficiently large data set before the learning process starts.

Incremental learning algorithms are designed to work on data streams and efficiently perform training and prediction each time a new example arrives. The main advantage is that they do not need to learn from scratch on the entire data set, saving time and computational costs [65]. Incremental learners extract knowledge from local information, i.e., from the most recent examples. This is especially important when we assume that the order of the data has some meaning and that the most recent data are aligned with the current state of the world [68]. Batch learning methods can be used to solve incremental tasks, but they have a computational overhead [68].

2.5 Related works

There is an enormous amount of work on using sensor data and self-reports to infer aspects of a person's life or a specific population. Many studies have investigated the use of passive sensor data in longitudinal studies that model human behaviours, such as depression detection algorithms [178] or educational performances [173]. Sensor data are used to perform activity recognition, i.e., provide information about users' activities [43], on evolving data streams [1], even in out-of-the-lab settings [54], and to recognize the context [170]. Mobile sensing works investigate, e.g., social interactions [134], recognition of flu-like symptoms [8] and eating event detection [6]. The main differences are that we are concerned with the egocentric point of view acquired by interacting with the user, and we handle noisy labels and changes in the world. Moreover, the work considers the lifelong setting, in which the machine and the reference user interact for long periods, even for the whole life. One last and crucial point is that in the first part of this work, we focus on designing robust ML algorithms that learn on tasks with the characteristics discussed above. Thus, they are not restricted to a specific domain like mobile sensing or context recognition.

Multiple studies proposed context or activity recognition on real-world data in which participants use the devices naturally, outside the lab and without instruction from the researchers. Thus, this setting implies unbalanced and incomplete labels, missing sensors [168], and, when dealing with streams, novel labels appear or become obsolete [2]. Given the variability of the patterns in real-world data, some approaches apply personalization on human activity recognition to generalize to other users [57, 155]. This work is concerned with noise handling and drift in hierarchical classification, and it considers interaction with human annotators. Thus, the machine interacts with the user to fix noisy examples or update the model to the undergone drift.

2.6 Conclusion

We introduced the personal context, and we described the challenges of its recognition in the wild. This means considering drift in the data due to changes in the world and noise. To this end, we must design a robust model that incrementally updates its knowledge based on the stream of input sensor data and subjective supervision of its users.

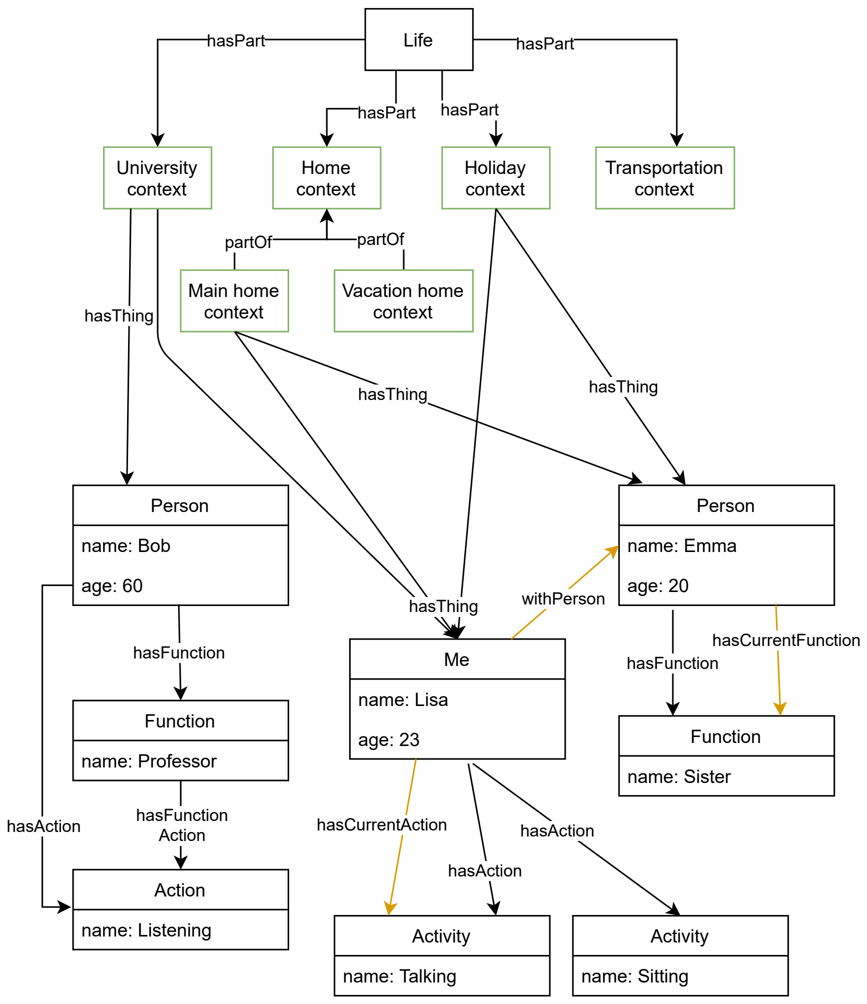


Figure 2.2: Entity Graph (EG) represents the actual entities used to define the personal context. The yellow arrows show the actual entities that describe the context in a specific moment in time.

Part I

Learning in the wild

3 Incremental Skeptical Learning

Contents

Changing conditions and noisy data are common problems for AI agents learning in open-world scenarios. Skeptical learning (SKL) is introduced in Section 3.3 and addresses the problem of noisy supervision by asking the user to revise the possible error. Noisy labels occur because the end-user is inattentive or unwilling to respond. However, the previous SKL design has some limitations that make it unsuitable for open-world scenarios in which there are changing conditions, like in our motivation example. To overcome these issues, Section 3.4 presents the SKL redesign, which leverages the uncertainty estimation of Gaussian processes. The experiments in Section 3.5 show that our redesign improves over the original formulation in terms of query allocation and performance.

Attribution This chapter includes material previously published as [22]. Stefano Teso, Fausto Giunchiglia and Andrea Passerini contributed significantly to the material presented in this chapter.

3.1 Introduction

Imagine a handheld personal assistant that provides guidance to an end-user. In order to give useful, timely suggestions (like "please take your insulin"), the agent must be aware of the user's context, for instance where she is ("at home"), what she is doing ("eating cake"), and with whom ("alone") [71]. The machine must infer this information from a stream of sensor readings (e.g., GPS coordinates, nearby Bluetooth devices), with the caveat that the target classes are user-specific (e.g., this user's home is not another user's home) and thus that the label vocabulary must be acquired from the user herself. Moreover, as the user visits new places and engages in new activities, the vocabulary changes. This simple example shows that, to be successful outside of the lab [46], AI agents must adapt to the changing conditions of the real world and to their end-users.

We study these challenges in a simplified but non-trivial setting, *interactive classification in the wild*, where an interactive learner requests labels from an end-user and the number of classes grows with time. A fundamental issue in this setting is that end-users often provide unreliable supervision [165, 174, 188]. This is especially problematic in the wild, as noisy labels may fool the machine into being under- or over-confident and into acquiring non-existent classes.

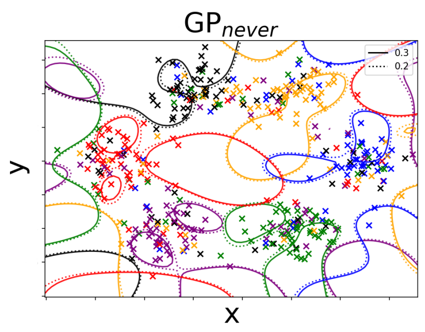
We address these issues by proposing Incremental Skeptical Gaussian Processes (ISGP), a redesign of skeptical learning [188] tailored for learning in the wild. In skeptical learning (SKL), if the interactive learner is confident that a newly obtained example is mislabeled, it immediately asks the annotator to reconsider her feedback. In stark contrast to other noise handling alternatives, SKL is designed specifically to retrieve the clean label from the annotator.

ISGP improves original SKL in four important ways. First, instead of relying on random forests, like SKL, ISGP builds on Gaussian Processes (GPs) [175]. Thanks to their explicit uncertainty estimates, GPs prevent pathological cases in which an overconfident learner 1) refuses to request the label of instances far from the training set, thus failing to learn, and 2) continuously challenges the user regardless of her past performance, estranging her. Second, ISGP makes use of the model's uncertainty to determine whether to be skeptical or credulous, while the original SKL uses an inflexible strategy that relies on the *number* of observed examples only. Third, while the previous SKL relies on several hard-to-choose hyper-parameters, ISGP makes use of a simple and robust algorithm that works well even without fine-tuning. Last, ISGP makes use of incremental learning techniques for improved scalability [109].

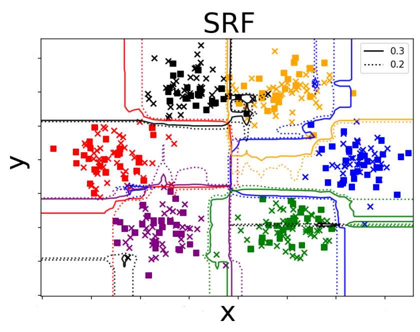
Contributions. Summarizing, we:

1. Introduce interactive classification in the wild, a novel form of interactive learning in which there is a substantial amount of labelling noise and new classes are observed over time;

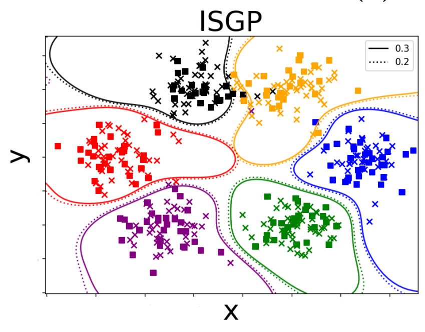
2. Develop ISGP, a simple and robust redesign of skeptical learning that leverages exact uncertainty estimates to allocate queries to the user appropriately and avoids over-confident models even in the presence of noise;



(a) Regular GP



(b) Skeptical learning



(c) ISGP

Figure 3.1: Illustration of ISGP on a 2D synthetic data set with six normally-distributed classes (in color) and noisy labels (corrupted at random with probability 0.4). The outlines enclose regions with high predictive probability (solid ≥ 0.3, dashed ≥ 0.2). Crosses and boxes are noisy examples; boxes have been cleaned by skeptical learning.

3. Showcase the advantages of ISGP – in terms of query budget allocation, prediction quality and efficiency – on a controlled synthetic task and on a real-world task.

3.2 Incremental classification in the wild

Interactive classification in the wild (ICW) is a sequential prediction task: in each round $t = 1, 2, \ldots,$ the learner receives an instance $x_t \in \mathcal{X}$ (e.g., a vector of sensor readings) and outputs a prediction $\hat{y}_t \in \mathcal{Y}$ (e.g., the user's location). The learner is also free to query a human annotator – usually the end-user herself – for the ground-truth label $y_t \in \mathcal{Y}$ (e.g., the true location). The goal of the learner is to *acquire a good predictor while keeping the number of queries at a minimum*, not to overload the annotator.

Two features make ICW unique: the amount of *label noise* and the presence of *task shift*.

Label noise. Label noise follows from the fact that human annotators are often subject to momentary inattention and may fail to understand the query [188]. The label $\tilde{y}_t$ fed back by the annotator is thus often wrong, i.e., $\tilde{y}_t \neq y_t$. Failure to handle noise can bloat the model and affect its accuracy [60]. Label noise is especially troublesome in ICW, as it can fool the model into being under- or over-confident. This, in turn, makes it difficult to identify informative instances and properly allocate labeling budget. Failing to detect mislabeled examples before using them for training prevents the model from spotting future mislabeled examples that fall in regions affected by these noisy examples.

Task shift. By task shift, we mean that newly received instances may belong to new and unanticipated classes. For this reason, we distinguish between the complete but unobserved set of classes $\mathcal{Y} \subseteq \mathbb{N}$ and the classes observed up to iteration t, that is[1] $\mathcal{Y}_t \subseteq \mathcal{Y}$. Hence, y_t belongs to $\mathcal{Y}$, $\tilde{y}_t$ to $\mathcal{Y}_t$, and $\hat{y}_t$ to $\mathcal{Y}_{t-1}$. To keep the task manageable, we assume that previously observed classes remain valid, i.e., $\mathcal{Y}_t \subseteq \mathcal{Y}_{t+1}$ for all t. In our personal aid example, this would imply that, for instance, the user's home remains the same over time. This is a reasonable assumption so long as the agent's lifetime is not too long. Chapter 5 studies other forms of task shift.

Note that ICW is not the same as open recognition [23]: both settings involve handling previously unseen classes, but open recognition is not interactive and it is unconcerned with noise. Thus, ICW is both easier, in new classes are annotated immediately, and harder, as noise makes can confuse the learner; see the Related Work for a discussion.

3.3 Original formulation of skeptical learning

Skeptical learning (SKL) is a noise-handling strategy designed for interactive learning [188]. SKL challenges the annotator about any suspicious examples it receives instead of blindly accepting the annotator's supervision In contrast with standard strategies for handling noise, like using robust models or discarding anomalies [60], SKL aims at recovering the ground truth.

In skeptical learning, an example is deemed suspicious if the learner is confident that the model's prediction is right and that the user's annotation is wrong. This requires the learner to assign a confidence level to its own predictions and to the user's annotations. SKL estimates these confidences using two separate heuristics. The confidence in the model is estimated using a combination of training set size and confidence reported by the model. The confidence in the user is based on the number of user mistakes spotted during past interaction rounds. To solve the conflict, the machine leverages previous

[1]It is assumed that $\mathcal{Y}_0$ is defined appropriately, e.g., $|\mathcal{Y}_0| \geq 1$.

knowledge to decide if the predicted and user labels are compatible. The knowledge has the forms of knowledge graphs that encode the relationships among the labels. For instance, one concept is the generalization of another one. The user is contradicted when the two labels are not compatible and the machine is confident enough. Then, the user is presented with both labels and can decide whether to accept one of the two or provide a new label. See [188] for more details.

The skeptical learner runs through three stages and the transition from one phase to another is defined by thresholds on the expectation of model confidence taken over all the past interaction rounds. The three stages are:

1. *train mode*, the learner always request supervision on new examples and never contradicts the user. The goal of the machine is to acquire enough knowledge about the user;

2. *refine mode*, once the model is confident enough, it begins to challenge the user. The models expected probability of querying the user exceeds a threshold;

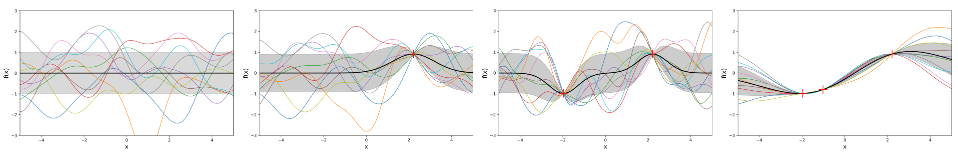
3. *regime mode*, the model begins to actively request labels when uncertain about its prediction and challenges the user on suspicious examples. It remains in this phase indefinitely.

3.3.1 Limitations

The original formulation of skeptical learning is not a good fit for ICW. First and foremost, SKL is based on random forests (RFs), which are robust to noise but also notoriously over-confident. This can be clearly seen in Figure 3.1a where RF is very confident even far away from the training set. This is a major issue, as over-confident predictors may stubbornly refuse to query novel and informative instances, compromising learning, and may keep challenging the user regardless of her past performance, overloading and estranging her.

The three-stage strategy fails in the wild, as new classes appear even in later learning stages, in which over-confident models may refuse to request supervision for them. This occurs frequently in our experiments. The active learning strategy in the last stage partially avoids over-confidence by requesting extra labels. In the first stage, the model may learn from noisy examples because the machine cannot challenge the user on suspicious examples.

Two other issues are that SKL requires to choose several hyper-parameters (like θ, which controls when to transition between stages), which is non-trivial in interactive settings, and that it retrains the RF from scratch in each iteration.



(a) prior **(b)** posterior **(c)** posterior **(d)** posterior

Figure 3.2: Example of Gaussian process regression over one dimension. Panel (a) shows ten functions drawn at random from the prior distribution. Panel (b) to (d) shows ten functions drawn at random from the prior conditioned on one, two and three observations, respectively. In all plots, the thick black line is the mean prediction and the grey-shaded area is twice the standard deviation.

3.4 Incremental Skeptical Gaussian Processes

ISGP is a redesign of skeptical learning based on Gaussian Processes (GPs) that avoids over-confident predictors and handles label noise. GPs are a natural choice in learning tasks like active learning [90, 133], online bandits [152], and preference elicitation [84], in which uncertainty estimates help to guide the interaction with the user. Our key observation is that skeptical learning is another such application.

3.4.1 Gaussian Processes

Introduction to GPs

Gaussian Processes (GPs) [175] are non-parametric distributions over functions $f : \mathcal{X} \to \mathbb{R}$. A GP is entirely specified by a mean function $\mu(x)$ and a covariance function $k(x, x')$. The latter encodes structural assumptions about the functions modeled by the GP and can be implemented with any kernel function. Without any observation, the average value over the sample functions at each x is assumed to be zero, i.e., $\mu(x) \equiv 0$.

The intuition behind Gaussian process is that a stochastic process defines the properties of functions. Figure 3.2 shows an example of Gaussian process for a 1-d regression task. Ten sample functions are drawn at random from the prior distribution over function and plotted in Figure 3.2a. This prior distribution encodes the assumption about the function before observing any training data. In Figures 3.2b to 3.2d, the model observes one, two and three new data points, respectively. The ten functions, drawn from the posterior distribution, pass through the data points. Hence, the uncertainty tends to decrease for instances close to the training examples (i.e., the similarity increases) and to increase far from known data. In the above plots, the shaded grey area denotes the uncertainty. The intuition is that similar instances should belong to the same class or have the same regression value.

This similarity between instances is defined by the covariance function.

Bayesian inference, that is, conditioning a GP on examples, produces another GP whose mean and covariance functions can be written in closed form. Letting $\mathbf{x}_t = (x_1, \ldots, x_t)^\top$ be the instances received so far and $\mathbf{y}_t = (y_1, \ldots, y_t)^\top$ their "scores" $y_t = f(x_t)$ (possibly perturbed by Gaussian noise), the mean and covariance functions conditioned on $(\mathbf{x}_t, \mathbf{y}_t)$ are:

$$\mu_t(x) = \mathbf{k}_t(x)^\top \Gamma_t \mathbf{y}_t \tag{3.1}$$

$$k_t(x, x') = k(x, x') - \mathbf{k}_t(x)^\top \Gamma_t \mathbf{k}_t(x') + \rho^2 \tag{3.2}$$

Here, we denote $\mathbf{k}_t(x)$ as the vector of covariances between the test point x and all the t training points, i.e., $\mathbf{k}_t(x) = (k(x_1, x), \ldots, k(x_t, x))^\top$, the $t \times t$ matrix of covariances between all pairs of training points is $K_t = [k(x, x') : x, x' \in \mathbf{x}_t]$, $\Gamma_t = (K_t + \rho^2 I)^{-1}$, and ρ, a smoothing parameter that models noise. Note that Eq. (3.2) depends only on the input data and not on the target value. Given a GP with parameters (μ, k) and x, the value of $f(x)$ is normally distributed with mean $\mu(x)$ and variance $k(x, x)$. Hence, the probability that $f(x)$ is non-negative is:

$$\mathbb{P}(f(x) \geq 0 \,|\, x) = \Phi\left(\frac{\mu(x)}{\sigma(x)}\right) \tag{3.3}$$

where Φ denotes the cdf of a standard normal distribution and $\sigma(x) = \sqrt{k(x, x)}$. This quantity is often used in classification tasks to model the probability of the positive class, that is, $\mathbb{P}(1 \,|\, x) = \mathbb{P}(f(x) \geq 0 \,|\, x)$, see [90]. Figure 3.3a shows an example on a binary classification task.

Incremental multi-class GPs

Incremental multi-class GPs (IMGPs) generalize Gaussian Processes to multi-class classification [109]. An IMGP can be viewed as a collection of GPs, one for each observed class $\ell \in \mathcal{Y}_t$, which share the same precision matrix Γ_t but have separate label vectors $\mathbf{y}_{\ell,t}$. The label vectors use a one-versus-all encoding: an element of $\mathbf{y}_{\ell,t}$ is 1 if the label of the corresponding example is ℓ and 0 otherwise. The posterior mean function of the ℓ-th GP is:

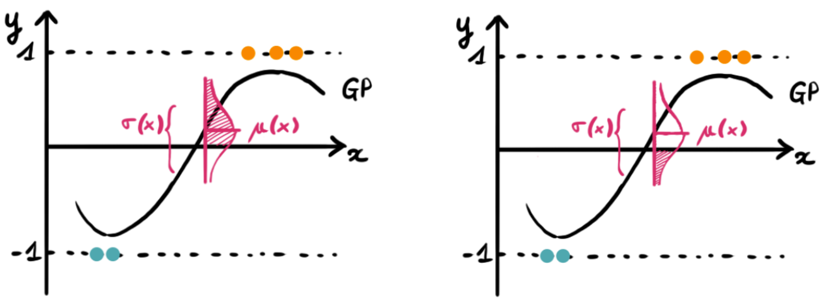
$$\mu_{\ell,t}(x) = \mathbf{k}_t(x)^\top \Gamma_t \mathbf{y}_{\ell,t} \tag{3.4}$$

Since the covariance function does not depend on the labels, it remains the same as in Eq. (3.2). The multi-class posterior is obtained by combining the GP posteriors with a soft-max:

$$\mathbb{P}(\ell \,|\, x_t) = \frac{1}{Z} \exp \mathbb{P}_\ell(1 \,|\, x_t), \tag{3.5}$$

where $\mathbb{P}_\ell(1 \,|\, x_t)$ is the posterior of the ℓ-th GP (Eq. (3.3)) and the normalization factor Z is

$$Z = \sum_{\ell'} \exp \mathbb{P}_{\ell'}(1 \,|\, x_t). \tag{3.6}$$



(a) uncertainty estimation **(b)** probability of asking a label

Figure 3.3: We consider the random variable of a binary classification task ($y = \pm 1$) for one-dimension input x. The plot shows the continuous hidden random variable $f(x)$, modeled by a GP, denoting the binary class of an unlabeled instance. The discrete label is generated according to the sign of $f(x)$. Solid black line is the mean $\mu(x)$, instances belonging to the positive class are denoted as orange points, whereas negative instances as blue points. **Panel (a)**: The probability that an instance x belongs to the positive class (Eq. (3.3)) equals the shaded purple area. The uncertainty in labeling x is maximum when $\mathbb{P}(f(x) \geq 0 \mid x)$ is close to 0.5, namely the quantity $\mu(x)/\sigma(x)$ is close to zero. **Panel (b)**: The plot shows the continuous hidden random variable $f(x)$ of the predicted class of an unlabeled instance. The probability of querying the annotator (Eq. (3.8)) is denoted by the shaded purple area. Intuitively, the probability of querying increases as the probability of the predicted class decreases. *Credits figures: Stefano Teso.*

IMGPs offer two major advantages. First, in IMGPs, the predictive variance is *guaranteed* to increase with the distance from the training set, as illustrated by Figure 3.1c. This guarantee prevents IMGPs from being over-confident about classes and instances that differ significantly from its previous experience, a key feature when learning in the wild. Another benefit is that IMGPs support incremental updates, i.e., in each iteration the updated precision matrix Γ_{t+1} is computed from Γ_t by exploiting the matrix-inversion lemma, without any matrix inversion [109]. This makes IMGPs scale much better than non-incremental learners and GPs; see Section 3.4.3 for a discussion.

3.4.2 The ISGP algorithm

In this section, we present ISGP, and the pseudo-code is listed in Algorithm 1. In each iteration t, the learner receives an instance x_t and predicts the most

Algorithm 1 Pseudo-code of ISGP. $\mathcal{Y}_0$ is provided as input. All branches are stochastic, see the relevant equations.

```
1:  for $t = 1, 2, \dots$ do
2:      receive $x_t$
3:      $\hat{y}_t \leftarrow \text{argmax}_{y \in \mathcal{Y}_{t-1}} \ \mu_y(x_t)$        ▷ Eq. (3.7)
4:      if uncertain about $\hat{y}_t$ then                                             ▷ Eq. (3.8)
5:          request label, receive $\tilde{y}_t$
6:          if skeptical about $\tilde{y}_t$ then                                       ▷ Eq. (3.9)
7:              challenge user with $\hat{y}_t$, receive $y'_t$
8:          else
9:              $y'_t \leftarrow \tilde{y}_t$
10:         add $(x_t, y'_t)$ to data set and update IMGP
11:         $\mathcal{Y}_t \leftarrow \mathcal{Y}_{t-1} \cup \{y'_t\}$
```

likely label (line 3):

$$\begin{aligned}\hat{y}_t &= \underset{\ell}{\text{argmax}}\, \mathbb{P}(\ell \,|\, x_t) \\ &= \underset{\ell}{\text{argmax}}\, \frac{1}{Z} \exp \mathbb{P}_\ell(1 \,|\, x_t) \\ &= \underset{\ell}{\text{argmax}}\, \Phi\left(\frac{\mu_{\ell,t}(x)}{\sigma_t(x)}\right) \\ &= \underset{\ell}{\text{argmax}}\, \mu_{\ell,t}(x_t) \end{aligned} \tag{3.7}$$

where $\ell \in \mathcal{Y}_{t-1}$. The last step holds because Φ is monotonically increasing and $\sigma_t(x)$ does not depend on ℓ.

Query the user. At this point, ISGP has to decide whether to request the label of x_t (line 4). In line with approaches to selective sampling [31, 16], ISGP prioritizes requesting the labels of uncertain instances, as these are more likely to impact the model. This also limits the labeling cost as the model improves. Intuitively, x_t is uncertain if either $\mu_{\hat{y}_t}(x_t)$ is small or $\sigma_t(x_t)$ is large; in either case, Eq. (3.3) ensures that $P_{\hat{y}_t}(1 \,|\, x_t)$ is small. Hence, ISGP queries the annotator with probability $\mathbb{P}_{\hat{y}_t}(0 \,|\, x_t)$. This is achieved by sampling a_t from a Bernoulli distribution with parameter α_t, defined as:

$$\begin{aligned}\alpha_t &= \mathbb{P}_{\hat{y}_t}(f(x_t) \leq 0 \,|\, x_t) \\ &= 1 - \Phi\left(\frac{\mu_{\hat{y}_t,t}(x_t)}{\sigma_t(x_t)}\right)\end{aligned} \tag{3.8}$$

and querying the user if $a_t = 1$. The choice is randomized so to prevent ISGP from trusting the model too much, which is problematic, especially during the first rounds of learning. Randomization is a key ingredient in online learning and selective sampling, cf. [31].

Challenge the user. If the check succeeds, ISGP has to decide whether to challenge the user's label (line 6). If the user and the machine agree on the label[2], the probability of challenging the user should be small; we set it to zero, for simplicity. Otherwise, it should increase with $\mathbb{P}_{\hat{y}_t}(1 \mid x_t)$ and decrease with $\mathbb{P}_{\tilde{y}_t}(1 \mid x_t)$. Since these probabilities come from different GPs, a direct comparison is not straightforward. In order to facilitate this, ISGP treats the GPs as if they were independent. Under this modeling assumption, letting f_ℓ be a sample from the ℓ-th GP, $\mathbb{P}(f_{\hat{y}_t}(x_t) \geq f_{\tilde{y}_t}(x_t))$ is a normal distribution with mean $\delta_t(x) = \mu_{\hat{y}_t}(x) - \mu_{\tilde{y}_t}(x)$ and variance $\sigma_t(x)$. ISGP determines whether to challenge the user by sampling from a Bernoulli with parameter γ_t:

$$\begin{aligned}\gamma_t &= \mathbb{P}(f_{\hat{y}_t}(x_t) - f_{\tilde{y}_t}(x_t) \geq 0) \\ &= \Phi\left(\frac{\delta_t(x_t)}{\sigma_t(x_t)}\right)\end{aligned} \tag{3.9}$$

This is analogous to the case of active queries discussed above. Despite relying on an (admittedly strong) modeling assumption, this strategy worked well in our experiments.

Once confronted by the learner, the user replies with a potentially cleaned label y'_t. As in the original formulation of SKL [188], this label is never contested by ISGP. The reason is that in our target applications the user is collaborative and label noise is mostly due to temporary inattention. Lastly, in line (10) the model is updated using the consensus example (x_t, y'_t) and the loop repeats.

3.4.3 Advantages and limitations

ISGP improves on the original formulation of skeptical learning [188] in several ways. A major benefit is that IMGPs are never over-confident in regions far away from the training set. This facilitates allocating the query budget and avoids pathological behaviors. Our empirical analysis shows that the original formulation has no such guarantees. ISGP is also simpler. ISGP uses the IMGP itself to model the confidence in the annotator's label, whereas the original implementation relies on a separate model trained heuristically. Also, learning is not heuristically split into stages and only two hyper-parameters are needed, namely k and ρ. Since hyper-parameters are

[2]The original formulation of SKL tackles hierarchical multi-class classification, in which the user and the machine can agree on a parent of the prediction and the annotation. For simplicity, we focus here on multi-class classification. The pathological behavior of SKL that our method fixes affects the hierarchical setting, too.

hard to tune properly in interactive tasks, this is a substantial advantage. The net effect is that ISGP performs better and more consistently.

A well-known weakness of GPs is their limited scalability, due to the need of storing all past examples and of performing a matrix inversion during model updates. The latter is avoided here by using incremental updates, which reduce the per-iteration cost from $O(t^3)$ to $O(t^2)$. This is enough for ISGP to run substantially faster than the original implementation of SKL and to handle weeks or months of interaction with no loss of reactivity, as shown by our real-world experiment. Sparse GP techniques [128] and stochastic variational inference for GP models [85] can speed up ISGP even further. Of course, GPs are not immediately applicable to lifelong tasks. However, as outlined in Chapter 6, we can introduce a forgetting or select mechanism that keeps only the relevant training data by discarding the obsolete or irrelevant to the task at hand.

Another limitation of ISGP is that the active and skeptical checks (that is, Eqs. (3.8) and (3.9)) rely on the GP of the predicted class only. The active check can be easily adapted to use information from all classes known to the IMGP by replacing $\mathbb{P}_\ell(1 \,|\, x_t)$ with $\mathbb{P}(\ell \,|\, x_t)$. An adaptation of the skeptical check, however, is non-trivial and left to future work. In practice, this does not seem to be an issue, as ISGP works much better than the original implementation of SKL.

Finally, both this redesign of SKL and the original implementation are completely black-box and do not attempt to explain to the supervisor why examples are considered suspicious by the machine, making it hard for him/her to establish or reject trust in the data and the model.

3.5 Experiments

We investigate the following research questions:

Q1 Does ISGP output better predictions than the original formulation of skeptical learning?

Q2 Does ISGP correctly identify mislabeled examples?

Q3 Does ISGP scale better than skeptical learning?

To address these questions, we implemented ISGP using Python 3 and compared it against three alternatives on a synthetic and a real-world data set. We compared ISGP against some alternatives:

- SRF: the original implementation of SKL [188] based on random forests;
- GP$_{\text{never}}$: active IMGP baselines that never challenge the user, i.e., the machine always trust the user label, thus the labels are never revised by the user. This can be considered as the performance of IMGP in the worst scenario.

- GP$_{\text{always}}$: active IMGP baselines that always the user, i.e., the machine never trust the user label, so for every incoming label, the user is asked to revised it. This is IMGP in the best scenario.

The experiments were run on a computer with a 2.2 GHz processor and 16 GiB of memory. The code and experimental setup can be downloaded from: `gitlab.com/abonte/incremental-skeptical-gp`.

3.5.1 Synthetic experiment

Experimental details. As a first experiment, we ran all methods on a synthetic data set with six classes, similar to Figure 3.1: 100 instances were sampled from six 2D normal distributions, one for each class, with different means and identical standard deviations (namely 1.5). As usual in active learning, the annotator's responses are simulated by an oracle. Our oracle replies to labeling queries with a wrong label η% of the time. We experimented with a low- ($\eta = 10$) and a high-noise regime ($\eta = 40$). (Notice that 40% noise rate is very high: 50% is the limit for learnability in binary classification [4].) While in [188] the oracle always replies to contradiction queries with the correct label, our oracle answers with a wrong label η% of the time (unless the label being contested is correct, in which case no mistake is possible). This is meant to better capture the behavior of human annotators, as the answer to contradiction queries can be incorrect. Results obtained using the original oracle are not substantially different from the ones below.

All results are 10-fold cross-validated. For each fold, training examples are supplied in a fixed order to all methods. The order has a noticeable impact on performance, so we studied two alternatives: *a)* instances chosen uniformly at random; *b)* instances chosen randomly from sequential clusters (red, then blue, *etc.*). This captures task shift, i.e., increasing number of classes. $\mathcal{Y}_0$ matches the first example provided. All GP learners used a squared exponential kernel

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right) \tag{3.10}$$

with the length scale ℓ of 2, the smoothing parameter that models noise $\rho = 10^{-8}$, and $\|x - x'\|^2$ is the squared Euclidean distance between the two feature vectors. The hyperparameters are not optimized. The number of trees of SRF was set to[3] 100. The methods were evaluated based on their F_1 score and query budget usage. For simplicity, the cost of skeptical queries was assumed to be similar to that of labeling queries.

[3]This matches the original paper. With 100 trees, SRF is already more computationally expensive than ISGP, so we didn't increase it.

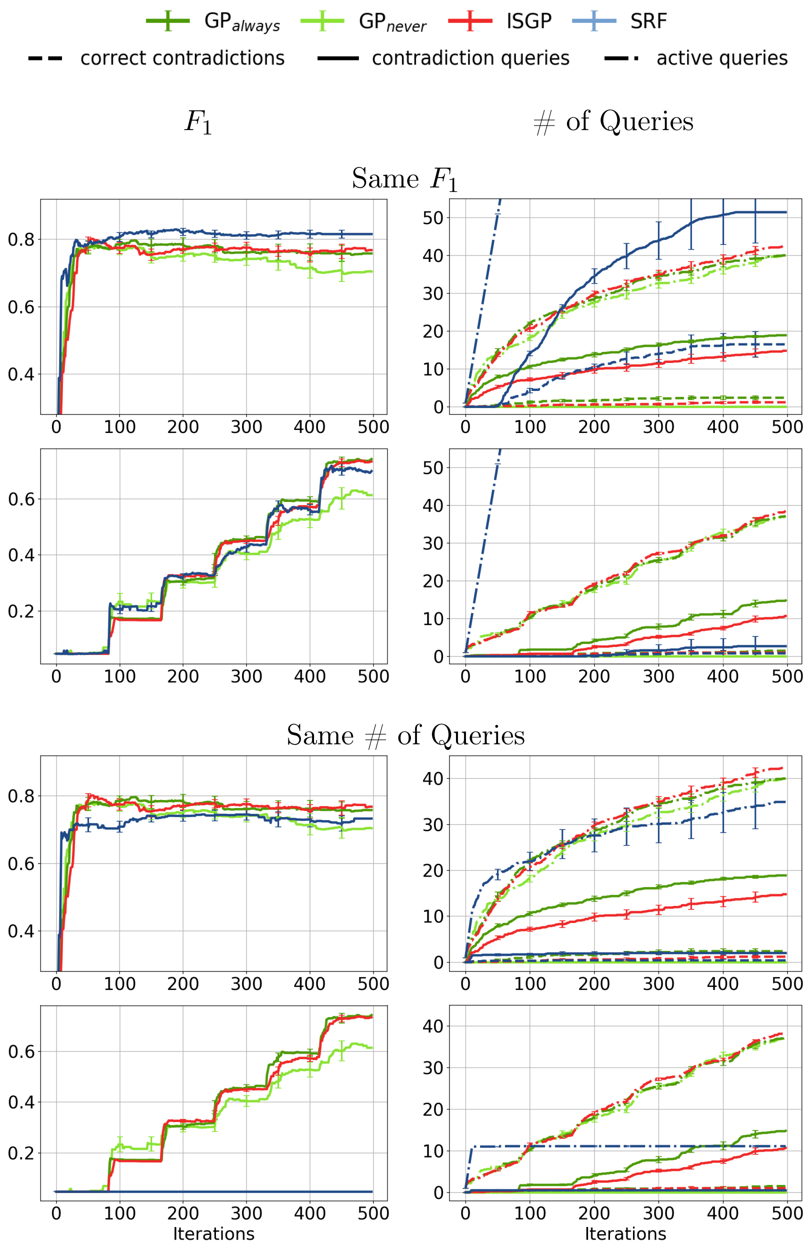


Figure 3.4: Results on synthetic data with 10% noise. The left column reports the F_1, and the right column reports # of labeling and contradiction queries (bars indicate the standard error). Top two rows: SRF tuned to match F_1 of ISGP. Bottom two rows: SRF tuned to match # of queries of ISGP (and forced to query at least 10 labels). Odd/even rows are random/sequential clusters, respectively.

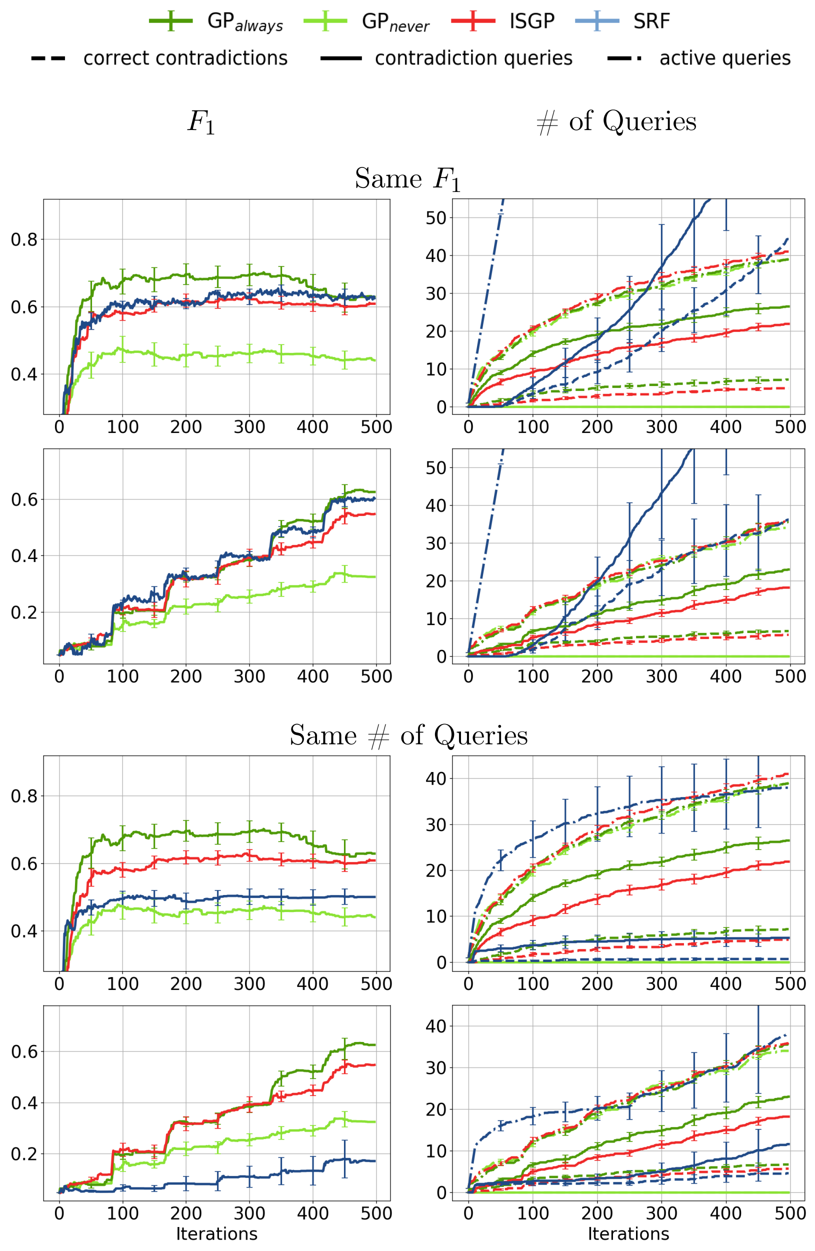


Figure 3.5: Results on synthetic data with 40% noise. The leftmost column reports the F_1, and the rightmost column shows the # of labeling and contradiction queries (bars indicate the standard error). Top two rows: SRF tuned to match F_1 of ISGP. Bottom two rows: SRF tuned to match # of queries of ISGP (and forced to query at least 10 labels). Odd/even rows are random/sequential clusters, respectively.

Results. The cross-validated results can be viewed in Figures 3.4 and 3.5. The plots show the F_1 score on the left-out fold and the cumulative number of queries made (dash-dot line: active queries; solid line: contradiction queries; dashed line: contradiction queries that uncovered an actual mistake). Figure 3.4 reports the performance of all methods at $\eta = 10\%$ noise, Figure 3.5 at 40%. In order to enable a fair comparison, we tuned SRF to either match the F_1 score of ISGP or its query budget utilization. This was achieved by tuning the hyper-parameter θ of SRF, which controls the length of the training and refinement stages of SRF, cf. Section 3.3: the longer the stages, the better the estimates acquired by the random forest but the worse the query usage. These two settings are illustrated by the top two and bottom two rows of Figures 3.4 and 3.5. Finally, odd rows refer to random instance selection order, and even rows to sequential order.

SRF worked well only in the low-noise, random order case (left columns, first row). Here, it managed to outperform our method by about 5%. This setting, however, is not very representative of ICW, as the user is quite consistent and examples from all classes are quickly obtained. In all other cases, SRF fails completely. Two trends are clearly visible. If tasked with reaching the F_1 score of ISGP, SRF tends to request the label of all new instances: the blue curve increases linearly beyond the plot y range. This is because the value of θ needed to reach a high enough F_1 score also forces SRF to remain in refine and train stage for most iterations. On the other hand, if the query budget is limited (bottom two rows), SRF quickly becomes over-confident and refuses to query the user. This is especially troublesome with task shift (bottom row), as the random forest becomes confident after seeing examples from mostly one class, leading to abysmal performance.

Our method does considerably better. Most importantly, ISGP does not suffer from pathological behavior and performs consistently across the board. The F_1 score typically increases with the number of queries made, even in the high-noise scenarios, while querying is not too aggressive – definitely not as aggressive as SRF. The F_1 and query curves also show much lower variance compared to SRF in most cases, as shown by the narrower error bars. It is easy to see that ISGP usually achieves F_1 score almost indistinguishable (in low-noise conditions, left two columns) or close (high-noise, right columns) to the F_1 of $\text{GP}_{\text{always}}$ with a comparable number of active queries and a smaller number of skeptical ones. Moreover, ISGP always outperforms GP_{never} in terms of F_1, as expected, while asking only 10–20 extra queries.

3.5.2 Location prediction

Experimental details. Next, we applied the methods to the location prediction task introduced in [188], which is reminiscent of our running example. The data includes 20 billion readings from up to 30 sensors collected from the smartphones of 72 users monitored over a period of two weeks us-

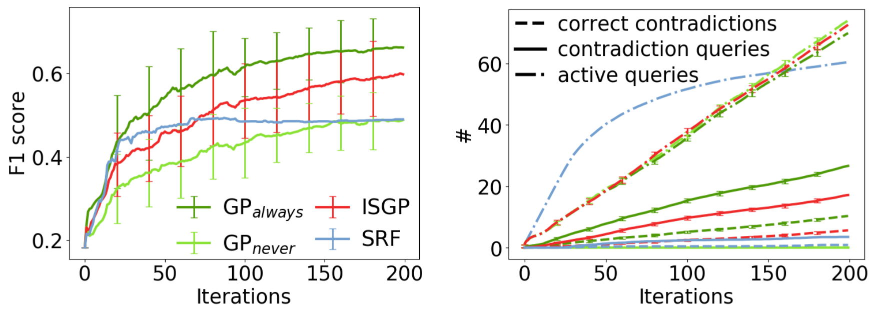


Figure 3.6: Results on location prediction. Left to right: F_1 score, # of queries (cumulative).

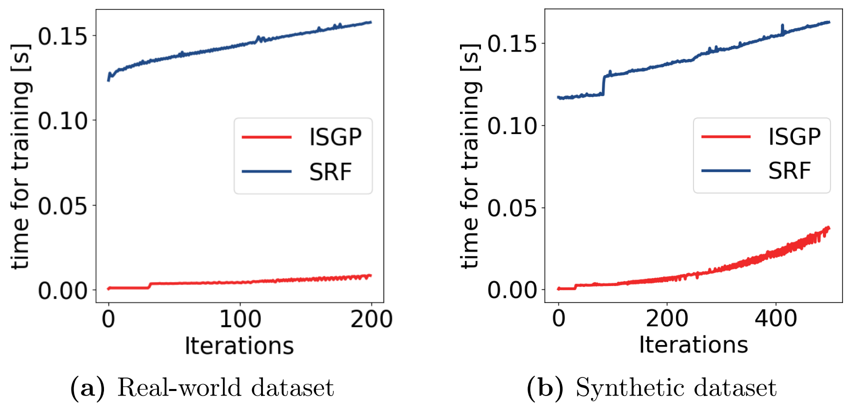


Figure 3.7: Run-time (not cumulative) as learning proceeds on real-world and synthetic dataset (the training step is performed at each iteration).

ing a mobile app (I-Log [187]), for a total of 110 GiB. The sensors are both hardware (i.e., gravity and temperature) and software (e.g., screen status, incoming calls). The mobile app also asks every 30 minutes the user what he or she is doing, where and with whom. We focus on location labels, for which an oracle exists capable of providing reliable ground truth annotations. The task consists in predicting the location of the user as *Home*, *University* or *Others*. The oracle identifies *Home* by clustering the locations labelled as home by the user via DBSCAN [55], and choosing the cluster where she spends most of the time during the night. *University* is identified using the maps of the University buildings, while all remaining locations are identified as *Others*. Please see [188] for the list of sensors and the pre-processing pipeline. The GP-based methods use a combination of constant, rational quadratic and squared exponential:

$$k_1(x, x') = \mathrm{C}(x, x') \times \left(\mathrm{RQ}\left(x, x'\right) + \mathrm{SE}\left(x, x'\right)\right) \tag{3.11}$$

where

$$\mathrm{C}(x, x') = \sigma_0^2 \tag{3.12}$$

$$\mathrm{RQ}(x, x') = \left(1 + \frac{\|x - x'\|^2}{2\alpha \ell_{rq}^2}\right)^{-\alpha} \tag{3.13}$$

$$\mathrm{SE}(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\ell_{se}^2}\right) \tag{3.14}$$

with $\alpha = 1$, $\ell_{rq} = 0.2$, $\ell_{se} = 1$ and $\sigma_0^2 = 1$. SRF uses 100 decision trees, as in the synthetic experiments.

Results. Figure 3.6 shows the result on the real-world dataset. The left-most plot highlights the F_1 scores and the subsequent one the number of queries. In this experiment, SRF is tuned to match the same number of queries of ISGP. The case where the two methods have a similar F_1 score is not reported since SRF shows the same behaviour as in the synthetic experiments (i.e., the number of active queries tends to increase rapidly). As in the synthetic experiments, the predictive performance of ISGP lies between $\text{GP}_{\text{always}}$ and GP_{never}, as expected. The number of active queries is also in line with the baselines, while the number of skeptical queries is very limited, roughly 15. Notice that the F_1 of SRF plateaus at roughly 70 iterations, while the performance of ISGP keeps increasing up to iteration 200. This trend is again explained by the fact that SRF becomes over-confident and requests the label of new instances very infrequently (second graph from left). All in all, these results confirm the considerations made in the synthetic experiment on a more challenging real-world ICW task. Finally, Figure 3.7 shows the training times of ISGP and SRF in the real world and the synthetic task. The advantage of the incremental updates is immediately apparent: SRF is substantially more computationally expensive in both tasks, making it a poor candidate for ICW with thousands of data points. Moreover, ISGP enjoys a reduction of about 70% of the predicting time in the location prediction task (data not shown).

3.6 Related work

Our work generalizes skeptical learning (SKL) [188] to incremental classification in the wild; the relationship between the two is analyzed in detail in Section 3.4.3. Below, we describe other related areas.

3.6.1 Open recognition

Open recognition (OR) [23] refers to learning problems like face verification, in which not all classes are observed during training. The goal is to attain low risk also on the unknown classes [139]. To this end, the learner attempts to distinguish between instances that belong to known classes (for which a prediction can be made) and instances that do not. This typically amounts to rejecting instances that lie away from the training set, thus bounding the chance of unjustified high-probability predictions [138, 136, 23]. Generalizations prescribe to annotate the detected unknown-class instances and re-train the model accordingly [15] and to employ incremental learning [42], as we do. ICW is not open in the above sense: while not all target classes are known, all incoming instances are *labeled*. What makes ICW hard is that the annotations are noisy, while OR is not concerned with shielding the model from noise. An additional difference is that in ICW there is no distinction between training and testing stages, as prediction and learning are interleaved. Moreover, skeptical learning requires and exploits interaction with human annotators, which is absent in OR.

3.6.2 Lifelong learning

In lifelong learning [162, 14] the learner witnesses a sequence of different but correlated classification tasks and the goal is to transfer knowledge from the previous tasks to the new ones. This is related to multi-task learning [144, 125]. Surprisingly, most existing algorithms either assume batch learning, although some do support incremental or online learning; cf. the discussion in [45]. The main differences with ICW are that lifelong learning is unconcerned with noise handling and does not consider interaction with human annotators.

3.6.3 Learning under noise

Label noise badly affects the predictor performance. Three typical strategies for learning from noisy labels are *(i)* label noise-robust models, e.g., decision tree [67] and ensemble methods, *(ii)* data cleansing methods, i.e., suspicious examples are discarded or down-weighted, *(iii)* models robust to noise like Bayesian approaches [4, 60, 119, 148]. Often a non-trivial noise ratio estimation is required step [120]. These approaches make no attempt to recover the ground-truth label and are not ideal in interactive learning settings characterized by high noise rate/cost and small data sets. Most works on interactive learning under noise are designed for crowd-sourcing applications in which different annotators of varying quality label items and the goal is to aggregate weak annotations into a high-quality consensus label [190]. In the case of multiple annotators with different reliability, the optimal one can be chosen [48]. Son et al. [147] propose a re-labeling of previously labeled examples for regression problems when the annotator is noisy. In our setting, the user is the only annotator available, and we can recover the ground truth only from her or him. Moreover, the timing of the interaction is crucial because the user may forget the ground truth if too much time has elapsed (e.g., ask the user about the activity performed seven days ago in the morning).

3.7 Conclusion

In this chapter, we introduced interactive classification in the wild (ICW), and ISGP, a redesign of skeptical learning based on Gaussian Processes. AI agents, such as personal assistants, are deployed in real-world scenarios where input data are sensor measurements and user annotations. The data arrives over time, thus, the stream contains new unforeseen labels that must be incorporated into the model. The characterizing features of ICW are noisy user annotations and the appearance of new unanticipated classes. ISGP solves ICW while avoiding pathological scenarios in which the learner always or never queries the annotator. Our empirical results showcase the benefits of our approach.

Our work can be further developed and improved in several directions. First, ISGP is applied to settings where the input and target are user-specific, like personal assistants, where the main requirement is to ensure the personalization of the model. However, applications like citizen science and crowdsourcing can also benefit from the presented approach. In that case, the correct annotation can be recovered from other users, domain experts or external systems. Second, a promising direction is to explain the machine skepticism by going beyond indicating only which example looks suspicious. Indeed, in Chapter 4, we propose to serve as an explanation of the model's suspicion training instances that are maximally incompatible with the suspicious example. The advantage of this approach is that the user can clean both the training examples that eluded the skeptical check and the suspicious example. In the case of visual input like images, video, or texts, the user can directly spot the incompatibility by looking at the example. However, it is not trivial how this can be replicated on inputs that encode sensor data (e.g., accelerometer, orientation). Third, exploiting semantic knowledge for conflict resolution as in [188] is crucial especially when the annotations are provided as free text. Hence, ISGP must be extended to the hierarchical classification.

Last, most of the research focus is on incremental and continuous learning in which new data must be included without forgetting what has been learn so far. However, in lifelong learning the amount of training data increases over time and becomes partially obsolete. The incremental multi-class GPs [109] underlying ISGP implement an efficient decremental learning, which can be leveraged to keep the only the most relevant data. Indeed, in Chapter 5, we propose an approach in the case of hierarchical classification to remove or update current training data.

In this chapter, we showed the benefit of relabeling suspicious examples and that ISGP overcomes some limitations of the previous formulation of skeptical learning. This evaluation are run in a controlled environment, namely using synthetic and real-world data where the user is replaced by an oracle and where the models performance can be easily evaluated on future data. In Chapter 7, we evaluate ISGP in a real-world scenario with real users for one month.

4 Explainable Skeptical Learning

Contents

We tackle sequential learning under label noise in applications where a human supervisor can be queried to relabel suspicious examples. The approach in Chapter 3 only relabel incoming examples that look "suspicious" to the model. As a consequence, those mislabeled examples that elude (or don't undergo) this cleaning step end up tainting the training data and the model with no further chance of being cleaned. We highlight these limitations in Section 4.2. In Section 4.3, we propose CINCER, a novel approach that cleans both new and past data by identifying *pairs of mutually incompatible examples*. Whenever it detects a suspicious example, CINCER identifies a counter-example in the training set that—according to the model—is maximally incompatible with the suspicious example, and asks the annotator to relabel either or both examples, resolving this possible inconsistency. The counter-examples are chosen to be maximally incompatible, so to serve as *explanations* of the model's suspicion, and highly influential, so to convey as much information as possible if relabeled. CINCER achieves this by leveraging an efficient and robust approximation of influence functions based on the Fisher information matrix (FIM). Our extensive empirical evaluation in Section 4.4 shows that clarifying the reasons behind the model's suspicions by cleaning the counter-examples helps in acquiring substantially better data and models, especially when paired with our FIM approximation. We conclude with the related work (Section 4.5) and a final discussion (Section 4.6).

Attribution This chapter includes material previously published as [159]. Stefano Teso, Fausto Giunchiglia and Andrea Passerini contributed significantly to the material presented in this chapter.

4.1 Introduction

Label noise is a major issue in machine learning as it can lead to compromised predictive performance and unreliable models [60, 148]. We focus on sequential learning settings in which a human supervisor, usually a domain expert, can be asked to double-check and relabel any potentially mislabeled example. Applications include crowd-sourced machine learning and citizen science, where trained researchers can be asked to clean the labels provided by crowd-workers [190, 99], and interactive personal assistants [22], where the user self-reports the initial annotations (e.g., about activities being performed) and unreliability is due to memory bias [165], unwillingness to report [39], or conditioning [174].

This problem is often tackled by monitoring for incoming examples that are likely to be mislabeled, *aka* suspicious examples, and ask the supervisor to provide clean (or at least better) annotations for them. Existing approaches and the one in Chapter 3, however, focus solely on cleaning the incoming examples [167, 99, 188]. This means that noisy examples that did not undergo the cleaning step (e.g., those in the initial bootstrap data set) or that managed to elude it are left untouched. This degrades the quality of the model and prevents it from spotting future mislabeled examples that fall in regions affected by noise.

We introduce CINCER (Contrastive and InflueNt CounterExample stRategy), a new explainable interactive label cleaning algorithm that lets the annotator observe and fix the *reasons* behind the model's suspicions. For every suspicious example that it finds, CINCER identifies a counter-example, i.e., a training example that maximally supports the machine's suspicion. The idea is that the example/counter-example pair captures a potential inconsistency in the data—as viewed from the model's perspective—which is resolved by invoking the supervisor. More specifically, CINCER asks the user to relabel the example, the counter-example, or both, thus improving the quality of, and promoting consistency between, the data and the model. Two hypothetical rounds of interaction on a noisy version of MNIST are illustrated in Figure 4.1.

CINCER relies on a principled definition of counter-examples derived from few explicit, intuitive desiderata, using influence functions [37, 96]. The resulting counter-example selection problem is solved using a simple and efficient approximation based on the Fisher information matrix [104] that consistently outperforms more complex alternatives in our experiments.

Contributions: Summarizing, we:

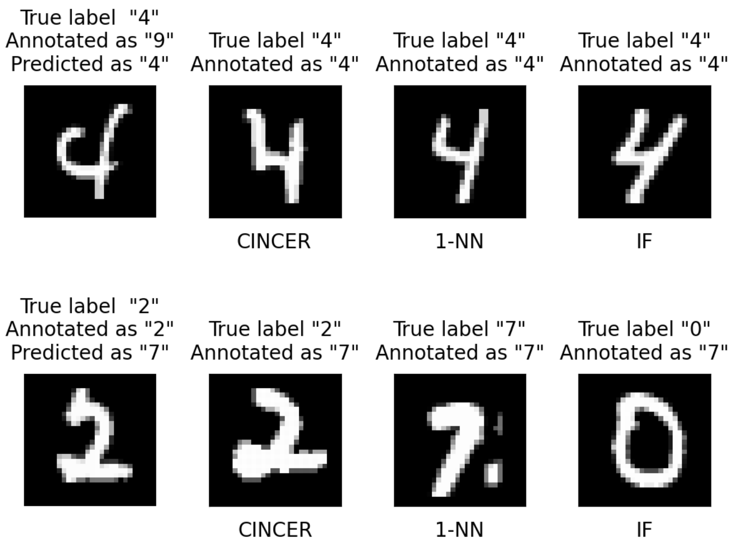


Figure 4.1: Suspicious example and counter-examples selected using (from left to right) CINCER, 1-NN and influence functions (IF), on noisy MNIST. **Top**: the suspicious example is mislabeled, the machine's suspicion is supported by a clean counter-example. **Bottom**: the suspicious example is not mislabeled, the machine is wrongly suspicious because the counter-example is mislabeled. CINCER's counter-example is contrastive and influential; 1-NN's is not influential and IF's is not pertinent, see desiderata D1–D3 below.

1. Introduce CINCER, an explanatory interactive label cleaning strategy that leverages example-based explanations to identify inconsistencies in the data—as perceived by the model—and enable the annotator to fix them.
2. Show how to select counter-examples that at the same time explain why the model is suspicious and that are highly informative using (an efficient approximation of) influence functions.
3. Present an extensive empirical evaluation that showcases the ability of CINCER of building cleaner data sets and better models.

4.2 Background

We are concerned with sequential learning under label noise. In this setting, the machine receives a sequence of examples $\tilde{z}_t := (\mathbf{x}_t, \tilde{y}_t)$, for $t = 1, 2, \ldots$, where $\mathbf{x}_t \in \mathbb{R}^d$ is an instance and $\tilde{y}_t \in [c]$ is a corresponding label, with $[c] := \{1, \ldots, c\}$. The label $\tilde{y}_t$ is unreliable and might differ from the ground-truth label y_t^*. The key feature of our setting is that a *human supervisor* can be asked to double-check and relabel any example. The goal is to acquire a clean dataset and a high-quality predictor while asking few relabeling queries, so to keep the cost of interaction under control.

The state-of-the-art for this setting is skeptical learning (SKL), introduced by Zeni et al. [188] and revised in Chapter 3. To briefly summarize, SKL is designed primarily for smart personal assistants that must learn from unreliable users. SKL follows a standard sequential learning loop: in each iteration, the machine receives an example and updates the model accordingly. However, for each example that it receives, the machine compares (an estimate of) the quality of the annotation with that of its own prediction, and if the prediction looks more reliable than the annotation by some factor, SKL asks the user to double-check his/her example. The details depend on the implementation: in [188] label quality is estimated using the empirical accuracy for the classifier and the empirical probability of contradiction for the annotator, while in Chapter 3 the machine measures the margin between the user's and machine's labels. The approach of this chapter follows the latter strategy.

Another very related approach is learning from weak annotators (LWA) [167, 99], which focuses on querying domain experts rather than end-users. The most recent approach [99] jointly learns a prediction pipeline composed of a classifier and a noisy channel, which allows it to estimate the noise rate directly. Moreover, the approach in [99] identifies suspicious examples that have a large impact on the learned model. A theoretical foundation for LWA is given in [167]. LWA is however designed for pool-based scenarios, where the training set is given rather than obtained sequentially. For this reason, in the remainder of the chapter, we will chiefly build on and compare to the previous work on SKL.

Limitations of existing approaches. A major downside of the previous work on SKL is that it focuses on cleaning the incoming examples only. This means that if a mislabeled example manages to elude the cleaning step and gets added to the training set, it is bound to stay there forever. This situation is actually quite common during the first stage of skeptical learning, in which the model is highly uncertain and trusts the incoming examples—even if they are mislabeled. The same issue occurs if the initial training set used to bootstrap the classifier contains mislabeled examples. As shown by our experiments, the accumulation of noisy data in the training set may have a detrimental effect on the model's performance (cf. Figure 4.2). In addition, it can also affect the model's ability to identify suspicious examples: a noisy data point can fool the classifier into trusting incoming mislabeled examples that fall close to it, further aggravating the situation.

4.3 Explainable interactive label cleaning

We consider a very general class of probabilistic classifiers $f : \mathbb{R}^d \to [c]$ of the form $f(\mathbf{x}; \theta) := \text{argmax}_{y \in [c]} \; P(y \mid \mathbf{x}; \theta)$, where the conditional distribution $P(Y \mid \mathbf{X}; \theta)$ has been fit on training data by minimizing the cross-entropy loss $\ell((\mathbf{x}, y), \theta) = -\sum_{i \in [c]} \mathbb{1}\{i = y\} \log P(i \mid \mathbf{x}, \theta)$. In our implementation, we also assume P to be a neural network with a softmax activation at the top layer, trained using some variant of SGD and possibly early stopping.

4.3.1 The CINCER algorithm

The pseudo-code of CINCER is listed in Algorithm 2. At the beginning of iteration t, the machine has acquired a training set $S_{t-1} = \{z_1, \ldots, z_{t-1}\}$ and trained a model with parameters θ_{t-1} on it. At this point, the machine receives a new, possibly mislabeled example $\tilde{z}_t$ (line 3) and has to decide whether to trust it.

Following skeptical learning [22], CINCER does so by computing the *margin* $\mu(\tilde{z}_t, \theta_{t-1})$, i.e., the difference in conditional probability between the model's prediction $\hat{y}_t := \text{argmax}_y P(y \mid \mathbf{x}_t, \theta_{t-1})$ and the annotation $\tilde{y}_t$. More formally:

$$\mu(\tilde{z}_t, \theta_{t-1}) := P(\hat{y}_t \mid \mathbf{x}_t, \theta_{t-1}) - P(\tilde{y}_t \mid \mathbf{x}_t, \theta_{t-1}) \tag{4.1}$$

The margin estimates the incompatibility between the model and the example: the larger the margin, the more suspicious the example. The example $\tilde{z}_t$ is deemed compatible if the margin is below a given threshold τ and suspicious otherwise (line 4); possible choices for τ are discussed in Section 4.3.5.

If $\tilde{z}_t$ is compatible, it is added to the data set as-is (line 5). Otherwise, CINCER computes a counter-example $z_k \in S_{t-1}$ that maximally supports the machine's suspicion. The intuition is that the pair $(\tilde{z}_t, z_k)$ captures a potential *inconsistency* in the data. For instance, the counter-example might be a correctly labeled example that is close or similar to $\tilde{z}_t$ but has a different label, or a distant noisy outlier that fools the predictor into assigning low probability to $\tilde{y}_t$. How to choose an effective counter-example is a major focus of this chapter and discussed in detail in Section 4.3.2 and following.

Next, CINCER asks the annotator to double-check the pair $(\tilde{z}_t, z_k)$ and relabel the suspicious example, the counter-example, or both, thus resolving the potential inconsistency. The data set and model are then updated accordingly (line 9) and the loop repeats.

4.3.2 Counter-example selection

Counter-examples are meant to illustrate why a particular example $\tilde{z}_t$ is deemed suspicious by the machine in a way that makes it easy to elicit useful corrective feedback from the supervisor. We posit that a good counter-example z_k should be:

Algorithm 2 Pseudo-code of CINCER. **Inputs**: initial (noisy) training set S_0; threshold τ.

```
1: fit $\theta_0$ on $S_0$
2: for $t = 1, 2, \ldots$ do
3:     receive new example $\tilde{z}_t = (\mathbf{x}_t, \tilde{y}_t)$
4:     if $\mu(\tilde{z}_t, \theta_{t-1}) < \tau$ then
5:         $S_t \leftarrow S_{t-1} \cup \{\tilde{z}_t\}$                ▷ $\tilde{z}_t$ is compatible
6:     else
7:         find counterexample $z_k$ using Eq. 4.14          ▷ $\tilde{z}_t$ is suspicious
8:         present $\tilde{z}_t, z_k$ to the user, receive possibly cleaned labels $y'_t, y'_k$
9:         $S_t \leftarrow (S_{t-1} \setminus \{z_k\}) \cup \{(\mathbf{x}_t, y'_t), (\mathbf{x}_k, y'_k)\}$
10:    fit $\theta_t$ on $S_t$
```

D1. *Contrastive*: z_k should explain why $\tilde{z}_t$ is considered suspicious by the model, thus highlighting a potential inconsistency in the data.

D2. *Influential*: if z_k is mislabeled, correcting it should improve the model as much as possible, so to maximize the information gained by interacting with the annotator.

In the following, we show how, for models learned by minimizing the cross-entropy loss, one can identify counter-examples that satisfy *both* desiderata.

What is a contrastive counter-example? We start by tackling the first desideratum. Let θ_{t-1} be the parameters of the current model. Intuitively, $z_k \in S_{t-1}$ is a contrastive counter-example for a suspicious example $\tilde{z}_t$ if *removing* it from the data set and retraining leads to a model with parameters θ_{t-1}^{-k} that assigns *higher* probability to the suspicious label $\tilde{y}_t$. The most contrastive counter-example is then the one that maximally affects the change in probability:

$$\operatorname*{argmax}_{k \in [t-1]} \left\{ P(\tilde{y}_t \,|\, \mathbf{x}_t; \theta_{t-1}^{-k}) - P(\tilde{y}_t \,|\, \mathbf{x}_t; \theta_{t-1}) \right\} \tag{4.2}$$

While intuitively appealing, optimizing Eq. 4.2 directly is computationally challenging as it involves retraining the model $|S_{t-1}|$ times. This is impractical for realistically sized models and data sets, especially in our interactive scenario where a counter-example must be computed in each iteration.

Influence functions. We address this issue by leveraging influence functions (IFs), a computational device that can be used to estimate the impact of specific training examples without retraining [37, 96]. Let θ_t be the empirical risk minimizer on S_t and $\theta_t(z, \epsilon)$ be the minimizer obtained after adding an example z with weight ϵ to S_t, namely:

$$\theta_t := \operatorname*{argmin}_\theta \frac{1}{t}\sum_{k=1}^{t} \ell(z_k, \theta)$$

$$\theta_t(z, \epsilon) := \operatorname*{argmin}_\theta \frac{1}{t}\left(\sum_{k=1}^{t} \ell(z_k, \theta)\right) + \epsilon\ell(z, \theta)$$

Taking a first-order Taylor expansion, the difference between $\theta_t = \theta_t(z, 0)$ and $\theta_t(z, \epsilon)$ can be written as $\theta_t(z, \epsilon) - \theta_t(z, 0) \approx \epsilon \cdot \left(\frac{d}{d\epsilon}\theta_t(z, \epsilon)\big|_{\epsilon=0}\right)$. The derivative appearing on the right-hand side is the so-called influence function, denoted $\mathcal{I}_{\theta_t}(z)$. It follows that the effect on θ_t of adding (resp. removing) an example z to S_t can be approximated by multiplying the IF by $\epsilon = 1/t$ (resp. $\epsilon = -1/t$). Crucially, if the loss is strongly convex and twice differentiable, the IF can be written as:

$$\mathcal{I}_{\theta_t}(z) = -H(\theta_t)^{-1}\nabla_\theta \ell(z, \theta_t) \tag{4.3}$$

where the curvature matrix $H(\theta_t) := \frac{1}{t}\sum_{k=1}^{t} \nabla^2_\theta \ell(z_k, \theta_t)$ is positive definite and invertible. IFs were shown to capture meaningful information even for neural networks and other non-convex models [96].

Identifying contrastive counter-examples with influence functions. To see the link between contrastive counter-examples and influence functions, notice that the second term of Eq. 4.2 is independent of z_k, while the first term can be conveniently approximated with IFs by applying the chain rule:

$$-\frac{1}{t-1}\left(\frac{d}{d\epsilon}P(\tilde{y}_t \,|\, \mathbf{x}_t; \theta_{t-1}(z_k, \epsilon))\bigg|_{\epsilon=0}\right) \tag{4.4}$$

$$= -\frac{1}{t-1}\left(\nabla_\theta P(\tilde{y}_t \,|\, \mathbf{x}_t; \theta_{t-1})^\top \frac{d}{d\epsilon}\theta_{t-1}(z_k, \epsilon)\bigg|_{\epsilon=0}\right) \tag{4.5}$$

$$= -\frac{1}{t-1}\nabla_\theta P(\tilde{y}_t \,|\, \mathbf{x}_t; \theta_{t-1})^\top \mathcal{I}_{\theta_{t-1}}(z_k) \tag{4.6}$$

The constant can be dropped during the optimization. This shows that Eq. 4.2 is equivalent to:

$$\operatorname*{argmax}_{k \in [t-1]} \; \nabla_\theta P(\tilde{y}_t \,|\, \mathbf{x}_t; \theta_{t-1})^\top H(\theta_{t-1})^{-1}\nabla_\theta \ell(z_k, \theta_{t-1}) \tag{4.7}$$

Eq. 4.7 can be solved efficiently by combining two strategies [96]: i) Caching the inverse Hessian-vector product (HVP) $\nabla_\theta P(\tilde{y}_t \,|\, \mathbf{x}_t; \theta_{t-1})^\top H(\theta_{t-1})^{-1}$, so that evaluating the objective on each z_k becomes a simple dot product, and ii) Solving the inverse HVP with an efficient stochastic estimator like LISSA [3]. This gives us an algorithm for computing contrastive counter-examples.

Contrastive counter-examples are highly influential. Can this algorithm be used for identifying influential counter-examples? It turns out that, as long as the model is obtained by optimizing the cross-entropy loss, the answer is affirmative. Indeed, note that if $\ell(z, \theta) = -\log P(y \mid \mathbf{x}; \theta)$, then:

$$\nabla_\theta P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) \tag{4.8}$$

$$= P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) \frac{\nabla_\theta P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1})}{P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1})} \tag{4.9}$$

$$= P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) \nabla_\theta \log P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) \tag{4.10}$$

$$= -P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) \nabla_\theta \ell(\tilde{z}_t, \theta_{t-1}) \tag{4.11}$$

Hence, Eq. 4.6 can be rewritten as:

$$-P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) \nabla_\theta \ell(\tilde{z}_t, \theta_{t-1})^\top H(\theta_{t-1})^{-1} \nabla_\theta \ell(z_k, \theta_{t-1}) \tag{4.12}$$

$$\propto -\nabla_\theta \ell(\tilde{z}_t, \theta_{t-1})^\top H(\theta_{t-1})^{-1} \nabla_\theta \ell(z_k, \theta_{t-1}) \tag{4.13}$$

It follows that, under the above assumptions and as long as the model satisfies $P(\tilde{y}_t \mid \mathbf{x}_t; \theta_{t-1}) > 0$, Eq. 4.2 is equivalent to:

$$\operatorname*{argmax}_{k \in [t-1]} \; -\nabla_\theta \ell(\tilde{z}_t, \theta_{t-1})^\top H(\theta_{t-1})^{-1} \nabla_\theta \ell(z_k, \theta_{t-1}) \tag{4.14}$$

This equation recovers *exactly* the definition of influential examples given in [96, Eq. 2] and shows that, for the large family of classifiers trained by cross-entropy, highly influential counter-examples are highly contrastive and vice versa, so that no change to the selection algorithm is necessary.

4.3.3 Counter-example selection with the Fisher information matrix

Unfortunately, we found the computation of IFs to be unreliable in practice, cf. [12]. This leads to unstable ranking of candidates and reflects on the quality of the counter-examples, as in Figure 4.1. The issue is that, for non-convex classifiers trained using gradient-based methods (and possibly early stopping), θ_{t-1} is seldom close to a local minimum, rendering the Hessian non-positive definite. In our setting, the situation is further complicated by the presence of noise, which dramatically skews the curvature of the empirical risk. Remedies like fine-tuning the model with L-BFGS [96, 181], preconditioning and weight decay [12] proved unsatisfactory in our experiments.

We take a different approach. The idea is to replace the Hessian by the Fisher information matrix (FIM). The FIM $F(\theta)$ of a discriminative model

$P(Y \mid \mathbf{X}, \theta)$ and training set S_{t-1} is [111, 101]:

$$F(\theta) := \frac{1}{t-1} \sum_{k=1}^{t-1} \mathbb{E}_{y \sim P(Y \mid \mathbf{x}_k, \theta)} \left[\nabla_\theta \log P(y \mid \mathbf{x}_k, \theta) \nabla_\theta \log P(y \mid \mathbf{x}_k, \theta)^\top \right] \tag{4.15}$$

It can be shown that, if the model approximates the data distribution, the FIM approximates the Hessian, cf. [163, 11]. Even when this assumption does not hold, as is likely in our noisy setting, the FIM still captures much of the curvature information encoded into the Hessian [111]. Under this approximation, Eq. 4.14 can be rewritten as:

$$\operatorname*{argmax}_{k \in [t-1]} \; -\nabla_\theta \ell(\tilde{z}_t, \theta_{t-1})^\top F(\theta_{t-1})^{-1} \nabla_\theta \ell(z_k, \theta_{t-1}) \tag{4.16}$$

The advantage of this formulation is twofold. First of all, this optimization problem also admits caching the inverse FIM-vector product (FVP), which makes it viable for interactive usage. Second, and most importantly, the FIM is positive semi-definite by construction, making the computation of Eq.4.16 much more stable.

The remaining step is how to compute the inverse FVP. Naïve storage and inversion of the FIM, which is $|\theta| \times |\theta|$ in size, is impractical for typical models, so the FIM is usually replaced with a simpler matrix. Three common options are the identity matrix, the diagonal of the FIM, and a block-diagonal approximation where interactions between parameters of different layers are set to zero [111]. Our best results were obtained by restricting the FIM to the top layer of the network. We refer to this approximation as "Top Fisher". While more advanced approximations like K-FAC [111] exist, the Top Fisher proved surprisingly effective in our experiments.

4.3.4 Selecting pertinent counter-examples

So far, we have discussed how to select contrastive and influential counter-examples. Now we discuss how to make the counter-examples easier to interpret for the annotator. To this end, we introduce the additional desideratum that counter-examples should be:

D3 *Pertinent*: it should be clear *to the user* why z_k is a counter-example for $\tilde{z}_t$.

We integrate D3 into CINCER by restricting the choice of possible counter-examples. A simple strategy, which we do employ in all of our examples and experiments, is to restrict the search to counter-examples whose label in the training set is the same as the prediction for the suspicious example, i.e., $y_k = \hat{y}_t$. This way, the annotator can interpret the counter-example as being in support of the machine's suspicion. In other words, if the counter-example is labeled correctly, then the machine's suspicion is likely right and the incoming example needs cleaning. Otherwise, if the machine is wrong and the suspicious example is not mislabeled, it is likely the counter-example – which backs the machine's suspicions – that needs cleaning.

Finally, one drawback of IF-selected counter-examples is that they may be perceptually different from the suspicious example. For instance, outliers are often highly influential as they fool the machine into mispredicting many examples, yet they have little in common with those examples [11]. This can make it difficult for the user to understand their relationship with the suspicious examples they are meant to explain. This is not necessarily an issue: first, a motivated supervisor is likely to correct mislabeled counter-examples regardless of whether they resemble the suspicious example; second, highly influential outliers are often identified (and corrected if needed) in the first iterations of CINCER (indeed, we did not observe a significant amount of repetitions among suggested counter-examples in our experiments). Still, CINCER can be readily adapted to acquire more perceptually similar counter-examples. One option is to replace IFs with *relative* IFs [11], which trade-off influence with locality. Alas, the resulting optimization problem does not support efficient caching of the inverse HVP. A better alternative is to restrict the search to counter-examples z_k that are similar enough to $\tilde{z}_t$ in terms of some given perceptual distance $\|\cdot\|_{\mathcal{P}}$ [86] by filtering the candidates using fast nearest neighbor techniques in perceptual space. This is analogous to FastIF [83], except that the motivation is to encourage perceptual similarity rather than purely efficiency, although the latter is a nice bonus.

4.3.5 Advantages and limitations

The main benefit of CINCER is that, by asking a human annotator to correct potential inconsistencies in the data, it acquires substantially better supervision and, in turn, better predictors. In doing so, CINCER also encourages consistency between the data and the model. Another benefit is that, by explaining the reasons behind the model's skepticism, CINCER allows the supervisor to spot bugs and justifiably build – or, perhaps more importantly, reject [137, 160] – trust into the prediction pipeline.

CINCER only requires to set a single parameter, the margin threshold τ, which determines how frequently the supervisor is invoked. The optimal value is highly application-specific, but generally speaking, it depends on the ratio between the cost of a relabeling query and the cost of noise. If the annotator is willing to interact (for instance, because it is paid to do so) then the threshold can be quite generous.

4.4 Experiments

We empirically address the following research questions:

Q1 Do counter-examples contribute to cleaning the data?

Q2 Which influence-based selection strategy identifies the most mislabeled counter-examples?

Q3 What contributes to the effectiveness of the best counter-example selection strategy?

We implemented CINCER using Python and Tensorflow [112] on top of three classifiers and compared different counter-example selection strategies on five data sets. The IF code is adapted from [180]. All experiments were run on a 12-core machine with 16 GiB of RAM and no GPU. The code for all experiments is available at: `https://github.com/abonte/cincer`.

4.4.1 Data sets

We used a diverse set of classification data sets:

- **Adult** [50]: data set of 48,800 persons, each described by 15 attributes; the goal is to discriminate customers with an income above/below $50K.
- **Breast** [50]: data set of 569 patients described by 30 real-valued features. The goal is to discriminate between benign and malignant breast cancer cases.
- **20NG** [50]: data set of newsgroup posts categorized in twenty topics. The documents were embedded using a pre-trained Sentence-BERT model [131] and compressed to 100 features using PCA.
- **MNIST** [103]: handwritten digit recognition data set from black-and-white, 28×28 images with pixel values normalized in the $[0, 1]$ range. The data set consists of 60K training and 10K test examples.
- **Fashion** [177]: fashion article classification dataset with the same structure as MNIST.

For adult and breast, a random $80 : 20$ training-test split is used while for MNIST, fashion and 20 NG the split provided with the data set is used. The labels of 20% of training examples were *corrupted* at random. The experiments were repeated five times, each time changing the seed used for corrupting the data. Performance was measured in terms of F_1 score on the (uncorrupted) test set. Error bars in the plots indicate the standard error. All competitors received exactly the same examples in exactly the same order.

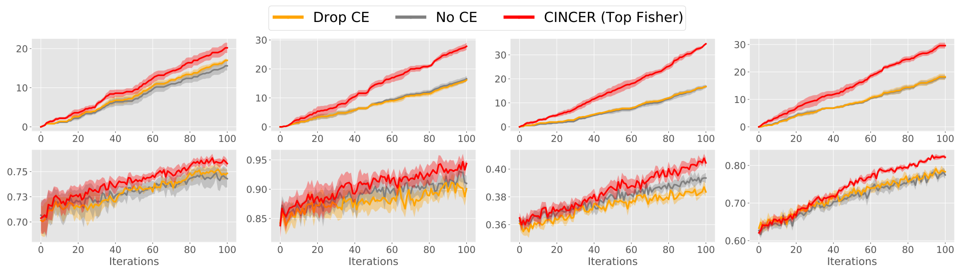


Figure 4.2: CINCER using Top Fisher vs. drop CE and no CE. **Left to right**: results for FC on adult, breast and 20NG, CNN on MNIST. **Top row**: # of cleaned examples. **Bottom row**: F_1 score.

4.4.2 Models

We applied CINCER to three models: **LR**, a logistic regression classifier; **FC**, a feed-forward neural network with two fully connected hidden layers with ReLU activations; and **CNN**, a feed-forward neural network with two convolutional layers and two fully connected layers. For all models, the hidden layers have ReLU activations and 20% dropout while the top layer has a softmax activation. LR was applied to MNIST, FC to both the tabular data sets (namely: adult, breast, german, and 20NG) and image data sets (MNIST and fashion), and CNN to the image data sets only. Upon receiving a new example, the classifier is retrained from scratch for 100 epochs using Adam [94] with default parameters, with early stopping when the accuracy on the training set reaches 90% for FC and CNN, and 70% for LR. This helps substantially to stabilize the quality of the model and speeds up the evaluation. Before each run, the models are trained on a bootstrap training set (containing 20% mislabeled examples) of 500 examples for 20NG and 100 for all the other data sets. The margin threshold is set to $\tau = 0.2$. Due to space constraints, we report the results on one image data set and three tabular data, and we focus on FC and CNN. The other results are consistent with what is reported below; these plots are reported in the Supplementary Material.

4.4.3 Q1: Counter-examples improve the quality of the data

To evaluate the impact of cleaning the counter-examples, we compare CINCER combined with the Top fisher approximation of the FIM, which works best in practice, against two alternatives, namely: **No CE**: an implementation of skeptical learning [22] that asks the user to relabel any incoming suspicious examples identified by the margin and presents no counter-examples. **Drop CE**: a variation of CINCER that identifies counter-examples using Top Fisher but drops them from the data set if the user considers the incoming example correctly labeled. The results are reported in Figure 4.2. The plots

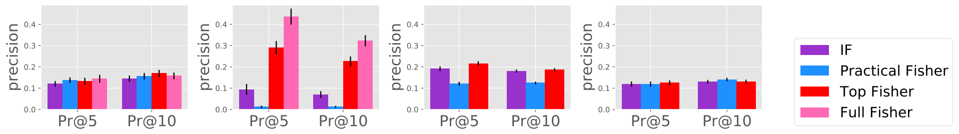


Figure 4.3: Counter-example Pr@5 and Pr@10. Standard error information is reported. **Left to right**: results for FC on adult, breast and 20NG, and CNN on MNIST.

show that CINCER cleans by far the most examples on all data sets, between 33% and 80% more than the alternatives (top row in Figure 4.2). This translates into better predictive performance as measured by F_1 score (bottom row). Notice also that CINCER consistently outperforms the drop CE strategy in terms of F_1 score, suggesting that relabeling the counter-examples provides important information for improving the model. These results validate our choice of identifying and relabeling counter-examples for interactive label cleaning compared to focusing on suspicious incoming examples only, and allow us to answer **Q1** in the affirmative.

4.4.4 Q2: Fisher Information-based strategies identify the most mislabeled counter-examples

Next, we compare the ability of alternative approximations of IFs to discover mislabeled counter-examples. To this end, we trained a model on a noisy bootstrap data set, selected 100 examples from the remainder of the training set, and measured how many truly mislabeled counter-examples are selected by alternative strategies. In particular, we computed influence using the IF LISSA estimator of [96], the actual FIM (denoted "full Fisher" and reported for the simpler models only for computational reasons) and its approximations using the identity matrix (*aka* "practical Fisher" [87]), and Top Fisher. We computed the precision at k for $k \in \{5, 10\}$, i.e, the fraction of mislabeled counter-examples within five or ten highest-scoring counter-examples retrieved by the various alternatives, averaged over 100 iterations for five runs. The results in Figure 4.3 show that, in general, FIM-based strategies outperform the LISSA estimator, with Full Fisher performing best and Top Fisher a close second. Since the full FIM is highly impractical to store and invert, this confirms our choice of Top Fisher as the best practical strategy.

4.4.5 Q3: Both influence and curvature contribute to the effectiveness of Top Fisher

Finally, we evaluate the impact of selecting counter-examples using Top Fisher on the model's performance, in terms of use of influence, by comparing it to an intuitive nearest neighbor alternative (**NN**), and modelling of the curvature, by comparing it to the Practical Fisher. **NN** simply selects

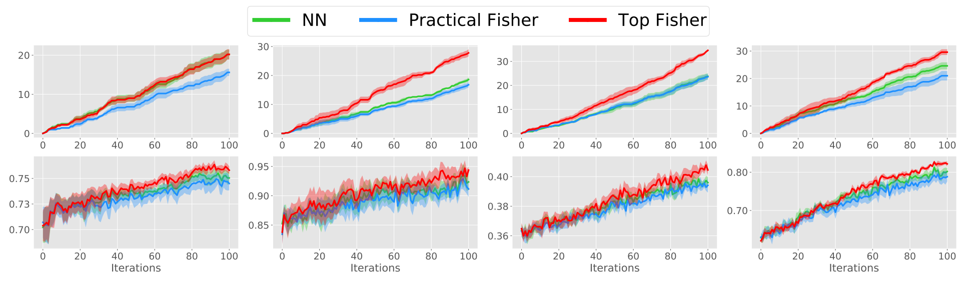


Figure 4.4: Top Fisher vs. practical Fisher vs. NN. **Left to right**: results for FC on adult, breast and 20NG, CNN on MNIST. **Top row**: # of cleaned examples. **Bottom row**: F_1 score.

the counter-example that is closest to the suspicious example. The results can be viewed in Figure 4.4. Top Fisher is clearly the best strategy, both in terms of number of cleaned examples and F_1 score. **NN** is always worse than Top Fisher in terms of F_1, even in the case of adult (first column) when it cleans the same number of examples, confirming the importance of influence in selecting impactful counter-examples. Practical Fisher clearly underperforms compared with Top Fisher, and it shows the importance of having the curvature matrix. For each data set, all methods make a similar number of queries: 58 for 20NG, 21 for breast, 31 for adult and 37 for MNIST. In general, CINCER detects around 75% of the mislabeled examples (compared to 50% of the other methods) and only about 5% of its queries do not involve a corrupted example or counter-example. The complete values are reported in the Supplementary Material. As a final remark, we note that CINCER cleans more suspicious examples than counter-examples (in a roughly 2 : 1 ratio), as shown by the number of cleaned suspicious examples vs. counter-examples reported in the Supplementary Material. Comparing this to the curve for Drop CE shows that proper data cleaning improves the ability of the model of being suspicious for the right reasons, as expected.

4.5 Related work

4.5.1 Influence functions and Fisher information

It is well known that mislabeled examples tend to exert a larger influence on the model [176, 96, 91, 11] and indeed IFs may be a valid alternative to the margin for identifying suspicious examples. Building on the seminal work of Koh and Liang [96], we instead leverage IFs to define and compute *contrastive* counter-examples that explain why the machine is suspicious. The difference is that noisy training examples influence the model as a whole, whereas contrastive counter-examples influence a specific suspicious example. To the best of our knowledge, this application of IFs is entirely novel. Notice also that empirical evidence that IFs recover noisy examples is restricted to offline learning [96, 91]. Our experiments extend this to a less forgiving interactive setting in which only one counter-example is selected per iteration and the model is trained on the whole training set. The idea of exploiting the FIM to approximate the Hessian has ample support in the natural gradient descent literature [111, 101]. The FIM has been used for computing example-based explanations by Khanna *et al.* [91]. However, their approach is quite different from ours. CINCER is equivalent to maximizing the Fisher kernel [87] between the suspicious example and the counter-example (Eq. 4.16) for the purpose of explaining the model's margin, and this formulation is explicitly derived from two simple desiderata. In contrast, Khanna *et al.* maximize a *function* of the Fisher kernel (namely, the *squared* Fisher kernel between z_k and $\tilde{z}_t$ divided by the norm of z_k in the RKHS). This optimization problem is not equivalent to Eq. 4.16 and does not admit efficient computation by caching the inverse FIM-vector product.

4.5.2 Other works

CINCER draws inspiration from explanatory active learning, which integrates local [160, 143, 105, 141] or global [127] explanations into interactive learning and allows the annotator to supply corrective feedback on the model's explanations. These approaches differ from CINCER in that they neither consider the issue of noise nor perform label cleaning, and indeed they explain the model's *predictions* rather than the model's *suspicion*. Another notable difference is that they rely on attribution-based explanations, whereas the backbone of CINCER is example-based explanations, which enable users to reason about labels in terms of concrete (training) examples [116, 88]. Following these works, saliency maps – which provide complementary information about relevant attributes – could potentially be integrated into CINCER to provide more fine-grained explanations and control.

4.6 Conclusion

We introduced CINCER, an approach for handling label noise in sequential learning that asks a human supervisor to relabel any incoming suspicious examples. Compared to previous approaches, CINCER identifies the reasons behind the model's skepticism and asks the supervisor to double-check them too. This is done by computing a training example that maximally supports the machine's suspicions. This enables the user to correct both incoming and old examples, cleaning inconsistencies in the data that confuse the model. Our experiments shows that, by removing inconsistencies in the data, CINCER enables acquiring better data and models than less informed alternatives.

Our work can be improved in several ways. CINCER can be straightforwardly extended to online active and skeptical learning, in which the label of incoming instances is acquired on the fly [110, 188]. CINCER can also be adapted to correcting multiple counter-examples as well as the reasons behind mislabeled counter-examples using “multi-round” label cleaning and group-wise measures of influence [97, 13, 66]. This more refined strategy is especially promising for dealing with systematic noise, in which counter-examples are likely affected by entire groups of mislabeled examples.

Potential negative impact. Like most interactive approaches, there is a risk that CINCER annoys the user by asking an excessive number of questions. This is currently mitigated by querying the user only when the model is confident enough in its own predictions (through the margin-based strategy) and by selecting influential counter-examples that have a high chance to improve the model upon relabeling, thus reducing the future chance of pointless queries. Moreover, the margin threshold τ allows to modulate the amount of interaction based on the user’s commitment. Another potential issue is that CINCER could give malicious annotators fine-grained control over the training data, possibly leading to poisoning attacks. This is however not an issue for our target applications, like interactive assistants, in which the user benefits from interacting with a high-quality predictor and is therefore motivated to provide non-adversarial labels.

5
Knowledge Drift

Contents

We introduce and study knowledge drift (KD), a special form of concept drift that occurs in hierarchical classification. Section 5.2 explains that under KD the vocabulary of concepts, their individual distributions, and the is-a relations between them can all change over time. The main challenge is that, since the ground-truth concept hierarchy is unobserved, it is hard to tell apart different forms of KD. For instance, the introduction of a new is- a relation between two concepts might be confused with changes to those individual concepts, but it is far from equivalent. Failure to identify the right kind of KD compromises the concept hierarchy used by the classifier, leading to systematic prediction errors. Our key observation is that in human-in-the-loop applications like smart personal assistants, the user knows what kind of drift occurred recently, if any. Motivated by this observation, we introduce TRCKD in Section 5.3, a novel approach that combines two automated stages—drift detection and adaptation—with a new interactive disambiguation stage in which the user is asked to refine the machine's understanding of recently detected KD. In addition, TRCKD implements a simple but effective knowledge-aware adaptation strategy. Our simulations (5.4) on three challenging data sets show that, when the structure of the concept hierarchy drifts, a handful of queries to the user are often enough to substantially improve prediction performance on both synthetic and realistic data. We conclude with Section 5.5 and we position our contribution with respect to the existing literature and conclude with some final remarks and a brief discussion of promising research directions.

Attribution This chapter includes material previously published as [21]. Fausto Giunchiglia, Stefano Teso and Andrea Passerini contributed significantly to the material presented in this chapter.

5.1 Introduction

We are concerned with human-in-the-loop applications of hierarchical classification. Our main interest lies in smart personal assistants (PAs) that must infer the location or social context of their user from sensor data (e.g., GPS, nearby Bluetooth devices) under the constraint that the hierarchy of relevant places and people changes over time [71]. In these applications, the concept hierarchy embedded into the predictor can become obsolete and has to be continually re-aligned [154].

We refer to this as *knowledge drift* (KD). KD is a complex phenomenon: concepts and *is-a* relations between them may appear, disappear, and change. The main challenge is distinguishing between different kinds of KD and especially between changes to the data distribution and to the concept hierarchy. Existing approaches to concept drift make no attempt at understanding whether shifts in the observed correlations between concepts are due to changes to the hierarchy (like adding an *is-a* relation) or not [63]. These leave similar footprints on the data, but confusing one for the other – and confusing different kinds of KD – entails acquiring spurious *is-a* relations or neglecting changes to the hierarchy, leading to systematic mis-predictions on future instances.

Our key observation is that, in our human-in-the-loop setting, *an expert user can identify with little effort what kind of drift occurred*, if any. Several examples are given below. Motivated by this observation, we design TRaCking Knowledge Drift (TRCKD) TRCKD (TRaCking Knowledge Drift), an approach that tackles KD by combining *automated* drift detection and adaptation with a novel, *interactive* drift disambiguation step. In particular, TRCKD maintains two windows of examples for each concept – one holds old data points and the other the most recent ones – and it detects KD by checking whether the distribution of current and past examples have diverged in distribution. The two empirical distributions are compared using the maximum mean discrepancy (MMD), a flexible and efficient kernel-based divergence [78]. Whenever it detects KD, TRCKD guesses what kind of KD occurred using a simple heuristic and presents this initial description to a knowledgeable human supervisor. The latter is then tasked with either confirming the machine's description or improving it, if necessary, according to her own understanding. Finally, in order to adapt the model to the different kinds of KD, TRCKD implements a simple but effective knowledge-aware adaptation strategy that we ground on top of kNN-based multi-label classifiers [151]. Our experiments show that, when changes to the structure of the concept hierarchy occur, interactive drift disambiguation and knowledge-aware adaptation are key for good performance under KD, and that asking a handful of queries to the user is often enough to achieve substantial performance improvements.

Contributions. Summarizing, we:

1. Identify *knowledge drift* as a special kind of concept drift;
2. Introduce the related issue of *drift disambiguation* and identify interaction with an expert user as a natural solution;
3. Design TRCKD, an approach for handling KD that combines automated detection and adaptation with interactive disambiguation, and instantiate it on top of kNN-based classifiers by implementing a knowledge-aware adaptation strategy;
4. Compare empirically TRCKD with state-of-the-art competitors on three representative data sets.

5.2 Hierarchical classification and knowledge drift

We consider learning tasks in which each instance $\mathbf{x}$ belongs to one or more concepts (classes) organized in a ground-truth hierarchy $H = (C, I)$, a directed acyclic graph in which nodes $C = \{1, \ldots, c\}$ index concepts and edges $I \subseteq C \times C$ encode *is-a* relations. Instances are labeled by indicator vectors $\mathbf{y} \in \{0, 1\}^c$, whose i-th element y^i is 1 if $\mathbf{x}$ belongs to the i-th concept in H and 0 otherwise.

During operation, the machine receives a stream of examples $\mathbf{z}_t = (\mathbf{x}_t, \mathbf{y}_t)$ drawn from a ground-truth distribution $P_t(\mathbf{X}, \mathbf{Y})$ that is *consistent* with a corresponding ground-truth hierarchy H_t. In other words, if H_t asserts that concept j *is-a* specialization of concept i, then the probability $P_t(\mathbf{X}, \mathbf{y})$ of all labels $\mathbf{y}$ that violate this relation (i.e., $y^j = 1$ and $y^i = 0$), is zero. The goal is to learn a predictor $\hat{P}_t$ (as well as a hierarchy $\hat{H}_t$ consistent with it) that outputs high-quality predictions on future instances.[1] It goes without saying that, in order to avoid systematic prediction mistakes, the acquired hierarchy $\hat{H}_t$ must closely resemble the ground-truth H_t.

Example 3 *Ann's PA receives observations* $\mathbf{x}$ *(like GPS coordinates, nearby Bluetooth devices) and uses them to predict that "Ann is studying at the library with Bob": "Studying", "Library", and "Bob" are concepts and the hierarchy states, among other things, that "Bob"* is-a *"Friend" of Ann's and a "Person".*

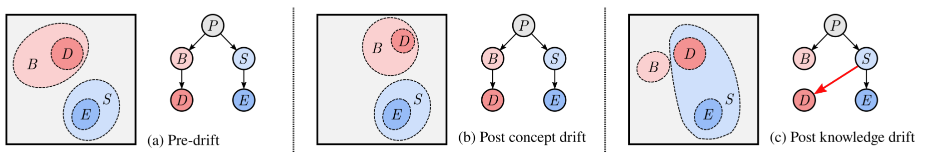


Figure 5.1: Left: Decision surface and hierarchy of a classifier for Ann's social context and five concepts: "Person", "Boss", "Subordinate", "Dave", and "Earl". **Middle**: Individual Concept Drift: Dave moves to a different office, so the decision surface changes but the hierarchy remains the same. **Right**: Knowledge Drift: Ann is promoted, "Dave" is now her subordinate. If the classifier knows that the hierarchy changed, it can transfer examples from "Dave" to "Subordinate", quickly improving its performance.

What makes our setting challenging is that the (unobserved) ground-truth concept hierarchy H_t and data distribution P_t can both unexpectedly and frequently change over time $t = 1, 2, \ldots$.

Approaches for dealing with concept drift do not capture this scenario. Indeed, regular concept drift is restricted to *distribution shift*, in which the prior distribution $P_t(\mathbf{X})$ changes, and *individual* or *multi-label concept drift*, in which the conditional distribution $P_t(Y_i \mid \mathbf{X})$ of one or more concepts changes [63, 197], but the concepts themselves are not mutually constrained by a ground-truth hierarchy.

Example 4 *During the semester, Ann spends most of her time studying at the library. Once the finals are over, Ann stops going to the library as often, and when she goes there she is less likely to be studying. This affects both the distribution of GPS coordinates and the conditional distribution of activities given GPS coordinates.*

In stark contrast, changes to the hierarchy H_t give rise to *knowledge drift* (KD), a special form of drift in which changes to the distribution are specifically due to changes to the hierarchy, cf. Figure 5.1. KD combines four types of atomic changes: *concept addition* and *removal*, which refer to the appearance of new concepts and the phasing out of obsolete concepts in C_t, respectively, and *relation addition* and *removal*, which refer to changes in the edges I_t themselves.

Example 5 *While concepts like "Friend" are immutable, the specific friends that matter to Ann (which are also concepts) change over time, e.g., if Ann moves abroad. For instance, if Ann buys a vacation home, a new concept "Ann's vacation home" appears in the ground-truth hierarchy (concept addition). Conversely, if Ann's vacation home is sold, the corresponding concept*

[1]We assume $\mathbf{y}$ to be given for ease of exposition. In practical applications where $\mathbf{y}$ is not given, it can be acquired using an active learning step.

is no longer meaningful and disappears (concept removal). If Ann receives a promotion and her old boss Dave becomes her subordinate, then "Dave" moves from being a child of "Boss" (relation removal) to being a child of "Subordinate" (relation addition).

Now, handling regular concept drift requires to detect changes in the data and adapt the model accordingly [63]. KD, on the other hand, requires an additional step, namely to understand what concepts and relations in the hierarchy H_t were affected, if any. This *drift disambiguation* step is crucial for preventing the estimated hierarchy $\hat{H}_t$ from getting misaligned, which could in turn lead to systematic, cascading prediction errors. It is also very challenging.

To see this, consider relation addition (RA). Like all forms of KD, RA can only be identified by its effects on the data, and specifically from the correlation between the concepts that it entails. However, correlations can exist and vary independently of the hierarchy. For instance, if Ann is visiting a branch of her company in another city, she could be taken out for lunch and dinner by Mary, the branch manager. Data could thus suggest a correlation between the concepts "Mary" and "Friend". Once Ann gets back home, she no longer hangs out with Mary, and the correlation drops dramatically. Since the machine has no sure way of telling apart RA from regular concept drift, it might wrongly add a spurious relation to its concept hierarchy – a mistake that takes plenty of examples to correct.

Similarly, concept removal (CR) implies not only that a concept i cannot occur (and should not be predicted) ever again, but also that its children are not longer attached to it. Treating CR as concept drift means that i) the conditional distribution $P_t(Y_i \mid \mathbf{X})$ is not constrained to zero, and that ii) the deleted concept might be predicted when one of its children is.[2] Naturally, multiple atomic changes can occur simultaneously, further complicating drift disambiguation.

5.3 Handling knowledge drift with TRCKD

Our key observation is that in many human-in-the-loop scenarios *the user can naturally disambiguate between different types of KD*. In our running example, for instance, Ann is perfectly aware that her vacation home has been recently sold, that Dave is now her subordinate and that Mary is a colleague and not a friend. It is therefore sensible to partially offload drift disambiguation to the user.

Motivated by this insight, we introduce TRCKD, a kNN-based approach for human-in-the-loop hierarchical classification under KD that combines *au-*

[2]The only "easy" case is concept addition, which is straightforward in our fully labeled setting and will not be considered further.

Algorithm 3 The TRCKD algorithm. Inputs: initial data set S_1 and hierarchy H_1, $s := |S_1|$, window size w, threshold τ, empirical estimator $\widehat{\mathrm{MMD}}$; $\mathbf{z}_t^i := (\mathbf{x}_t, y_t^i)$.

```
1: Fit initial classifier on $S_1$ and $H_1$, $\hat{H}_1 \leftarrow H_1$
2: for every concept $i$ in $\hat{H}_1$ do
3:     $W_{\mathrm{old}}^i \leftarrow \{\mathbf{z}_s^i, \ldots, \mathbf{z}_{s-w}^i\}$
4: for $t = 1, 2, \ldots$ do
5:     Receive new example $\mathbf{z}_t$
6:     for every concept $i$ in $\hat{H}_t$ do
7:         $W_{\mathrm{cur}}^i \leftarrow \{\mathbf{z}_{s+t}^i, \mathbf{z}_{s+t-1}^i, \ldots, \mathbf{z}_{s+t-w}^i\}$
8:     if $\exists i : \widehat{\mathrm{MMD}}(W_{\mathrm{cur}}^i, W_{\mathrm{old}}^i) \geq \tau$ then
9:         Illustrate detected KD to the user
10:        Adapt based on user's KD description
```

tomated detection and adaptation with a new, *interactive* drift disambiguation stage. Owing to their flexibility, kNN-based approaches are a popular choice for learning under drift [63] and achieve considerable performance in non-trivial learning tasks [135].[3] Our work builds on MW-kNN [151], an kNN-based approach to multi-class classification under drift that adapts to change by passively forgetting old, potentially obsolete examples. TRCKD upgrades MW-kNN from multi-class to hierarchical classification and additionally integrates a sliding-window approach [92] for drift detection. Moreover, TRCKD introduces a simple but effective *knowledge-aware* adaptation strategy specifically tailored for kNN-based classifiers (and that generalizes to other instance-based predictors).

The pseudo-code of TRCKD is listed in Algorithm 3. The algorithm takes a data set S_1 and a concept hierarchy H_1 consistent with it and uses them to train an initial classifier. Then, in each iteration $t = 1, 2, \ldots$ the machine receives a new example $\mathbf{z}_t = (\mathbf{x}_t, \mathbf{y}_t)$ and performs three steps: 1) It detects whether KD occurred, 2) It cooperates with the user to determine what concepts and relations were affected by KD, and 3) It adapts the classifier and the machine's hierarchy accordingly. We discuss these three steps in turn.

5.3.1 Step 1: Detection

For every concept i in $\hat{H}_t$, TRCKD maintains two windows of examples: W_{cur}^i holds the w most recent examples and is updated in each iteration, while W_{old}^i

[3]While the ideas behind TRCKD do carry over to other models, a proper assessment using non-kNN architectures is outside of scope for this chapter and left to future work.

holds w reference (past) examples.[4] Predictions for concept i are obtained by applying kNN to W^i_{cur}.

TRCKD detects drift by looking for changes in distribution between the recent and past windows of each concept. To this end, it employs the *maximum mean discrepancy* (MMD), a discrepancy employed in hypothesis testing [78] and domain adaptation [191]. Let P and Q be distributions over some space $\mathcal{X}$ and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a user-defined kernel. Then the MMD between P and Q relative to k is given by

$$\text{MMD}(P, Q)^2 = \mathbb{E}[k(\mathbf{a}, \mathbf{a}')] - 2\mathbb{E}[k(\mathbf{a}, \mathbf{b})] + \mathbb{E}[k(\mathbf{b}, \mathbf{b}')], \tag{5.1}$$

where $\mathbf{a}$, $\mathbf{a}'$ are drawn i.i.d. from P and $\mathbf{b}$, $\mathbf{b}'$ from Q. Estimating the MMD between W^i_{old} and W^i_{cur} requires to define a kernel between examples $\mathbf{z}$. TRCKD achieves this by defining two separate kernels over instances and labels k_X and k_Y and then taking their tensor product, i.e.,

$$k(\mathbf{z}, \mathbf{z}') = k((\mathbf{x}, y), (\mathbf{x}', y')) = k_X(\mathbf{x}, \mathbf{x}') \cdot k_Y(y, y'). \tag{5.2}$$

In our experiments, we employ a Gaussian kernel k_X for the instances and a delta kernel $k_Y(y, y') = \mathbb{1}\{y = y'\}$ for the labels.

The MMD is well-behaved, in the sense that if $P \equiv Q$ then $\text{MMD}(P, Q) = 0$ and if k is characteristic, the converse also holds [158]. However, unlike other well-behaved alternatives – for instance the total variation distance, the $\mathcal{A}$-distance [92], and the mutual information [124] – the MMD can be estimated efficiently (in linear time) even for high-dimensional data [78]. In addition, the MMD allows to select concrete examples (witness points) that illustrate the difference between the two distributions, simplifying the interaction with the user [106]. It also performs well empirically: in our experiments, the MMD achieved a better false detection rate than ME [89], a state-of-the-art discrepancy with better discrimination power on paper.

5.3.2 Step 2: Disambiguation

Upon detecting drift, TRCKD initiates interaction with the user by presenting a visualization of the detected KD and asking the user to verify and potentially improve a description of the detected KD.

If the KD detected by the machine identifies the right concepts – which should typically be if the drift detector is tuned well – then the user's jobs is to tell the machine whether the *relations* between the highlighted concepts have undergone drift and how. To this end, the user can modify the visualization by selecting or deselecting highlighted concepts and *is-a* edges. As long as the user is expert enough, she is likely to improve the machine's guess. A sufficiently motivated and knowledgeable user has also the option of editing

[4]TRCKD reserves 1/3 of each window to positive examples to account for class imbalance [151].

any drifting concepts or relations that were not detected by the machine, providing even more guidance. Notice that even *partial* improvements to the detected KD are likely to improve future predictive performance and are in any case better than no adaptation and fully automated disambiguation.[5]

Extra context can be supplied to the user by presenting a handful of examples that summarize how the concepts affected by KD have changed. Such examples can be selected from the past and current windows of those concepts using MMD witness functions [106].

5.3.3 Step 3: Adaptation

Once it receives the user's drift description, TRCKD adapts the machine's hierarchy and the windows accordingly. Here we present a simple knowledge-aware adaptation strategy for kNN-based approaches. In particular:

- For every instance of *individual concept drift* in the description, it transfers the contents of the current window of the affected concept to the past window and keeps only the u most recent examples in the former, i.e.:

  $$W^i_{\text{old}} \leftarrow W^i_{\text{cur}}, \qquad W^i_{\text{cur}} \leftarrow \{\mathbf{z}^i_{s+t}, \mathbf{z}^i_{s+t-1}, \dots, \mathbf{z}^i_{s+t-u}\} \tag{5.3}$$

  where u is a user-provided hyperparameter. All examples are not longer used for the classification task except for the most recent ones, which are likely drawn from the post-drift distribution and thus kept to facilitate recovery.

- For *concept removal*, the past and current windows of the affected concept are deleted, that is:

  $$W^i_{\text{old}} \leftarrow \varnothing, \qquad W^i_{\text{cur}} \leftarrow \varnothing \tag{5.4}$$

  Furthermore, all *is-a* relations between the removed concept are deleted and the children of the latter are attached to its parent. The concept will not occur and thus should not be predicted.[6] Notice that concept removal cannot be handled as concept drift, otherwise the deleted concepts would end up being predicted whenever one of its children is. The child concepts are no longer attached to it. The parent's window is reduced in size by w elements since it has to accommodate the examples of fewer child classes.

[5]The assumption is that the user is expert enough and therefore does not inject a lot of noise into the loop. This is a reasonable assumption to make applications like smart personal assistants.

[6]Notice that, by definition, a removed concept cannot recur unless it is added again to the hierarchy. This is handled separated via concept addition.

- For *relation addition*, the positive examples belonging to the child concept are copied to the ancestors' windows and the former are increased in size by w in order to take the examples of the new child. Given the child concept r:

$$W^i_{\text{old}} \leftarrow W^i_{\text{cur}}, \qquad W^i_{\text{cur}} \leftarrow W^i_{\text{cur}} \cup W^r_{\text{cur}} \tag{5.5}$$

  for each ancestor i. This adaptation ensures that the ancestors are predicted whenever the children is.

- For *relation removal*, the positive examples belonging to the child concept are removed from the parent's window and the latter is shrank accordingly. The child concept is also linked directly to its grand-parent. Given i and r the parent and the child respectively, then:

$$W^i_{\text{old}} \leftarrow W^i_{\text{cur}};\ W^i_{\text{cur}} = W^i_{\text{cur}} \setminus W^r_{\text{cur}} \tag{5.6}$$

Our experiments show that, despite its simplicity, our knowledge-aware adaptation strategy outperforms the knowledge-oblivious strategies of state-of-the-art kNN classifiers [151, 135]. It is reasonable to expect the benefits of knowledge-aware adaptation to carry over to other classifiers, e.g., neural networks [27].

5.4 Experiments

We empirically address the following research questions:

Q1 Is knowledge-aware adaptation useful for handling knowledge drift?

Q2 Does interaction with an expert user help adaptation?

Q3 Does TRCKD work in realistic, multi-drift settings?

The code of TRCKD as well as the complete experimental setup are available at: `https://gitlab.com/abonte/handling-knowledge-drift`.

Competitors. We compared TRCKD against several alternatives:

- **PAW-kNN**: punitive adaptive window kNN, a *state-of-the-art* multi-label approach that employs a single sliding window for all concepts to address gradual drift, and that adapts by discarding examples responsible for prediction mistakes in case of abrupt drift [135];

- **MW-kNN**: the multi-window kNN approach of [151] designed specifically for multi-label problems that TRCKD builds on;

- ***k*NN 1-window**: kNN with a single sliding window for all concepts that simply forgets old examples;

- ***k*NN**: regular kNN with no adaptation.

Data sets. We ran experiments on three data sets from different domains:

- **H-STAGGER**: a hierarchical version of STAGGER, a widely used synthetic data set of two-dimensional objects with three categorical attributes (shape, color, size) and labeled by drifting random formulas like "small and (green or red)" [140]. H-STAGGER has 3 attributes with 4 values each and labels instances using 5 different drifting random formulas chosen to have a reasonable pos./neg. ratio. The hierarchy is created by selecting two concepts as part of a third one that acts as parent concept.

- **H-EMNIST**: a data set of 28×28 handwritten digits and (uppercase, lowercase) letters [36]. The data set was converted to hierarchical classification by grouping different characters into higher level concepts, e.g., even numbers and vowels. The digits and letters are grouped in 5 concepts and the hierarchy is created as for H-STAGGER. Instances were embedded using a variational autoencoder [95].

- **H-20NG**: a data set of newsgroup posts categorized in twenty topics.[7] The data set was converted to hierarchical classification by grouping different classes into super-topics (e.g., "religion' grouping alt.atheism, soc.religion.christian and talk.religion.misc). The documents were embedded using a pre-trained Sentence-BERT model [131] and compressed to 100 features using PCA.

All instances always belong to the *root* concept in the concept hierarchy. Each data sets is converted into a streams by sampling a sequence of examples at random. Table 5.1 reports, for each data sets, the number of attributes d, the number of concepts c, as well as the following three measures of annotation density taken from [192]: the average number of positive labels per example LC, the empirical probability that a label is positive LD, and how many distinct combinations of positive categories (out of 2^c) are annotated in the data DL.

5.4.1 Experimental details

All experiments were run on a machine with eight 2.8 GHz CPUs and 32 GiB RAM. Each experiment was run ten times by randomizing the choice of examples in the stream: 2 runs were used for hyperparameter selection and 8 for evaluation. The plots report the average and the standard error over the 8 runs. All methods received exactly the same sequences of examples.

The results for concept drift are independent of our knowledge-handling strategy, so our evaluation focuses on concept deletion, relation addition, and relation removal. In these three cases, drift is injected into the stream

[7]From https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

Name	$\lvert S \rvert$	d	c	LC	LD	DL
H-STAGGER	570	3	6	4.42 ± 0.16	0.63 ± 0.02	34.63 ± 11.89
H-EMNIST	570	10	9	3.50 ± 0.04	0.44 ± 0.00	55.38 ± 5.57
H-20NG	570	100	6	2.19 ± 0.00	0.31 ± 0.00	8.5 ± 0.5

Table 5.1: Data sets statistics (mean ± std. dev.) averaged over the runs of the sequential drift experiment. $|S|$: number of instances, d: number of attributes, c: number of labels, LC: label cardinality, LD: label density, DL: distinct label set. The metrics are averaged on 8 runs and refer to the experiment for Q3.

by removing a random concept from the available ones, adding a random relation, and removing a random relation, respectively. KD starts after all competitors approximately reach their peak performance, namely after 100 iterations for H-STAGGER and H-EMNIST, and 170 for H-20NG.

Performance was measured in terms of micro F_1 score on a hold-out test set (of size 64 for H-STAGGER and 200 for H-EMNIST and H-20NG) randomly selected before each run and shared by all competitors. The micro F_1 score we use considers the sparsity of positive annotations that is characteristic of hierarchical classification; that is, most concepts are negative most of the time. Letting $\{(\mathbf{x}_i, \mathbf{y}_i) : i = 1, \ldots, n\}$ be the examples in the *test set*, $\hat{\mathbf{y}}_i$ their predictions and y_i^j the jth element of y_i (which is 1 if $\mathbf{x}_i$ belong to the jth concept and 0 otherwise), the micro F_1 for multi-label classification is defined as follows [149]:

$$F_{1\text{-micro}} = \frac{2 \sum_{j=1}^{c} \sum_{i=1}^{n} y_i^j \hat{y}_i^j}{\sum_{j=1}^{c} \sum_{i=1}^{n} y_i^j + \sum_{j=1}^{c} \sum_{i=1}^{n} \hat{y}_i^j}$$

where j iterates over concepts and i over examples.

User replies were simulated by an oracle that always answers correctly to disambiguation queries. More specifically, the user confirms that a drift occurred if and only if the concept detected as drifting by MMD has actually undergone drift, i.e., the concept has been removed, has drifted or is the child or parent of an added/removed relation.

5.4.2 Hyperparameters

The window size of all window-based methods was set to $w = 200$. This value enables these approaches to achieve the same performance (micro F_1) as kNN when *no* drift is present. To speed up detection, in TRCKD the MMD of each concept i is computed on the 70 most recent examples in W_{cur}^i only. This choice does not sacrifice reliability of detection.

The MMD threshold τ used by TRCKD, the number of neighbors k used by all competitors are selected in two independent runs by optimizing the

RQ	**Drift**	τ	**TRCKD** k	**PAW-kNN** k
		H-STAGGER		
Q1 - Q2	CD	0.04	3	3
	CR	0.04	11	3
	RA	0.04	11	3
	RR	0.04	11	3
Q3	All	0.04	11	3
		H-EMNIST		
Q1 - Q2	CD	0.04	5	3
	CR	0.05	3	3
	RA	0.05	3	3
	RR	0.04	3	3
Q3	All	0.05	3	3
		H-20NG		
Q1 - Q2	CD	0.05	3	3
	CR	0.05	3	3
	RA	0.04	3	3
	RR	0.04	3	3
Q3	All	0.05	3	3

Table 5.2: Hyperparameter values. Abbreviations: CD is concept drift, CR concept removal, RA relation addition, and RR relation removal.

micro F_1. τ is selected from $\{0.4, 0.5\}$ as these values were frequently observed to indicate drift in preliminary experiments. k was selected from $\{3, 5, 11\}$ and it was chosen independently for TRCKD plus its variants (including MW-kNN) and for PAW-kNN. Table 5.2 reports the values used in each experiment of these hyperparamters. All other hyperparameters were kept fixed across experiments. The penalty ratio of PAW-kNN was set to $p = 1$ as suggested in [135], while the minimum and maximum window sizes were set to $m_{min} = 50$ for H-STAGGER and H-EMNIST and 80 for H-20NG data set, and $m_{max} = 200$ respectively. The number of examples retained in W_{cur} in case of concept drift adaptation was set to $u = 10$. The distribution ratio parameter of TRCKD and MW-kNN was set to $r = 2/3$ as in [151].

5.4.3 Q1: Knowledge-aware adaptation improves performance

To evaluate the impact of our adaptation strategy, we compare TRCKD against MW-kNN, PAW-kNN, kNN 1-window, regular kNN in a setting where all approaches are told exactly when KD occurs. In this setting, TR-

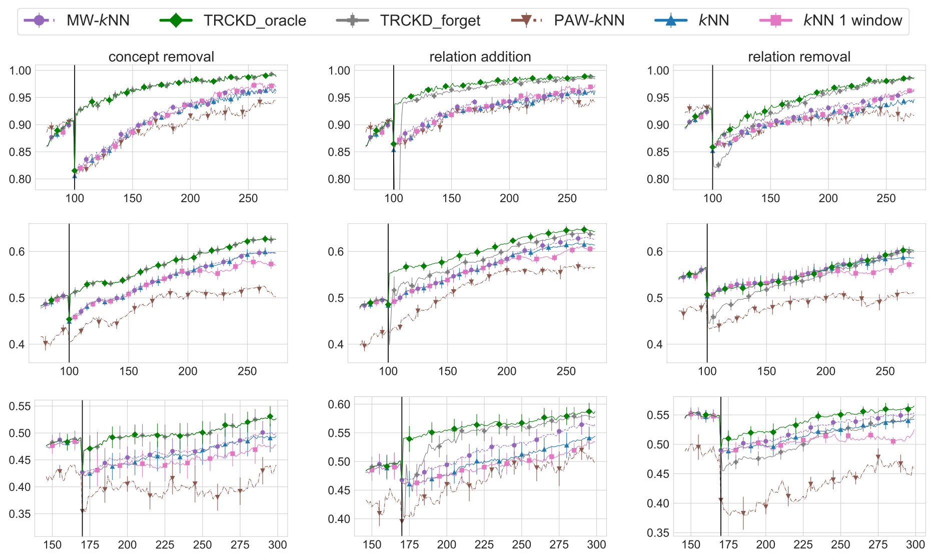


Figure 5.2: Comparison in terms of micro F_1 between TRCKD and standard forgetting strategies for kNN-based methods. Top to bottom: results for H-STAGGER, H-EMNIST and H-20NG. Left to right: concept removal, relation addition, and relation removal. Error bars indicate std. error.

CKD is denoted TRCKD$_{oracle}$, as knowledge-aware disambiguation combined with exact detection implies a perfect knowledge of the kind of drift that occurred. We also compare to a simple knowledge-unaware variant of TR-CKD, named TRCKD$_{forget}$, that adapts to all types of KD by forgetting old examples.

The results can be viewed in Figure 5.2. The plots show that TRCKD$_{oracle}$ is the best performing method on all data sets and for all forms of KD. Note how the performance difference between TRCKD$_{oracle}$ and the alternatives is larger than standard errors in most cases, especially when considering the first iterations after the drift. Most often, the runner up is TRCKD$_{forget}$: while it performs similarly to TRCKD$_{oracle}$ for concept removal (because our adaptation strategy boils down to forgetting in this simple setup), it does lag behind for relation addition and removal, showing a sizeable advantage for knowledge-aware adaptation. MW-kNN works reasonably well but suffers from relying on passive adaptation and does not always perform better than the two kNN baselines. PAW-kNN tends to underperform on H-EMNIST and H-20NG, especially when KD affects the relations. These results validate knowledge-aware adaptation on all data sets and allow us to answer **Q1** in the affirmative. For this reason, we will focus on knowledge-aware adaptation in the following experiments.

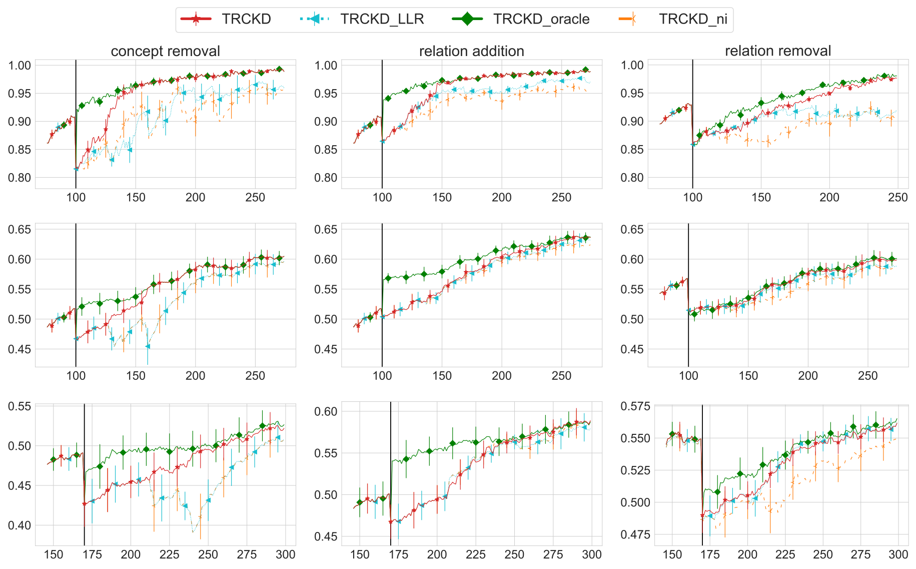


Figure 5.3: Comparison in terms of micro F_1 between TRCKD and less interactive variants. Top to bottom: results for H-STAGGER, H-EMNIST and H-20NG. Left to right: concept drift, concept removal, relation addition, and relation removal. Error bars indicate std. error.

5.4.4 Q2: Interaction is beneficial

To measure the impact of interaction, we compare four variants of TRCKD that differ in what information they elicit from the supervisor, namely: TRCKD, TRCKD$_{oracle}$, TRCKD$_{LLR}$ and TRCKD$_{ni}$.

TRCKD$_{LLR}$ is a fully-automated version of TRCKD that follows up MMD detection by performing drift disambiguation with a likelihood ratio test. This test detects a relation y^j *is-a* y^i iff

$$\frac{P(y^j \mid y^i)}{P(y^j \mid \neg y^i)} \geq \beta \tag{5.7}$$

with $\beta = \infty$. (This is the best possible value for β as the ground-truth data is assumed to be always consistent with the ground-truth hierarchy.) If the test detects relation addition/removal, TRCKD$_{LLR}$ applies the corresponding knowledge-aware adaptation strategy, otherwise it defaults to emptying the current window of the detected concept(s).

TRCKD$_{ni}$ is like TRCKD except that instead of interacting with the user it assumes that all concepts detected as drifting by MMD have undergone individual concept drift and adapts by purging their current window.

The results in Figure 5.3 are quite intuitive: TRCKD$_{oracle}$ substantially outperforms all alternatives in all cases except for relation removal in H-20NG. This shows that, if drift is detected correctly and timely, interactive

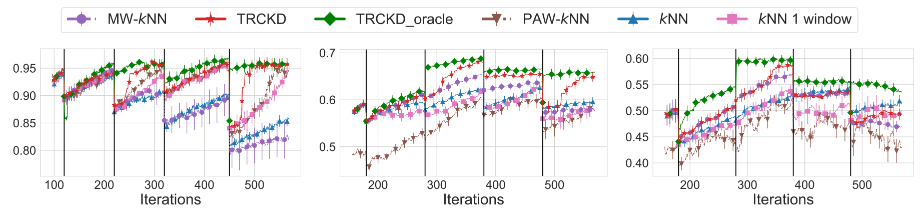


Figure 5.4: TRCKD versus competitors on sequential KD in terms of micro F_1. Left: H-STAGGER. Middle: H-EMNIST. Right: H-20NG. Error bars indicate std. error.

disambiguation is extremely useful for guiding knowledge-aware adaptation and quickly aligning the model to the ground-truth. TRCKD tends to perform substantially better than the no-interaction baselines TRCKD$_{LLR}$ and TRCKD$_{ni}$ and if MMD detection works well it quickly reaches the performance of the oracle. This allows us to answer **Q2** in the affirmative. If MMD underperforms (as in H-EMNIST), TRCKD does not get a chance to quickly interact with the user and shows no improvement. This could be fixed by better optimizing the choice of kernel and threshold used by MMD, perhaps by turning them into per-concept parameters. This is left to future work. The complexity of the task and the impact of the drift vary across the data sets. Thus, the difference between TRCKD and the competitor is smaller in some cases. Importantly, TRCKD interacts with the user 1.54 ± 0.78 times per run on average, showing that few interaction rounds are often enough to achieve a noticeable performance boost.

5.4.5 Q3: TRCKD works well in multi-drift settings

We consider a realistic scenario with four sequential KD events: concept drift, relation addition, relation removal, and concept removal. The results in Figure 5.4 show that TRCKD tends to outperform all competitors except the oracle. The advantage is quite marked whenever the KD affects the concept hierarchy itself, up to +10% F_1 for H-STAGGER and +5% for H-EMNIST. The plots mirror the advantages shown by TRCKD in the previous experiments and highlight that the benefits of knowledge-aware adaptation and interaction carry over to more realistic settings. This allows us to answer **Q3** in the affirmative. The lack of reactive adaptation penalizes MW-kNN and PAW-kNN, the latter especially on H-EMNIST.

5.4.6 Additional comparisons

TRCKD outperforms structure learning. Given the similarity between drift disambiguation and structure learning for probabilistic graphical models [98], it is natural to ask whether structure learning techniques could

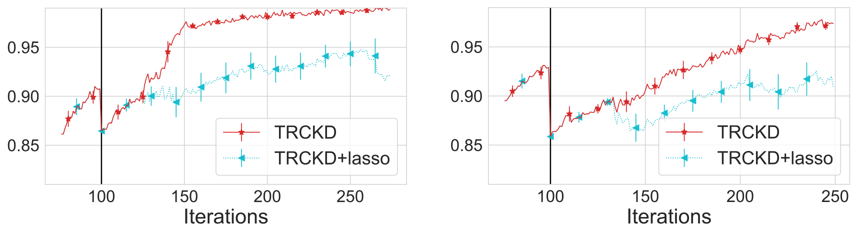


Figure 5.5: Micro F_1 results on H-STAGGER, automatic drift type identification with Graphical Lasso and MMD for detection versus TRCKD. Left: relation addition. Right: relation removal.

be used for handling KD in a fully automatical manner.

To answer this question, we evaluated a variant of TRCKD that uses graphical lasso to reconstruct the structure of the hierarchy from the most recent examples [61]. In particular, in each iteration a data set is built by combining the 70 most recent examples (analogously to what is done for MMD) for all concepts in the machine's hierarchy. This data set set is fed to graphical lasso, which spits out an (sparse) undirected graph based on the empirical correlation between all the concepts. The directions of individual edges are set so to maximize the likelihood of the child implying the parent and edges are treated as *is-a* relations. The resulting directed graph replaces the machine's concept hierarchy. The difference between the previous and current concept hierarchy is used to perform knowledge-aware adaptation.

A comparison between TRCKD +lasso and TRCKD is reported in Figure 5.5. It turns out that for relation addition and removal, graphical lasso often fails to estimate the ground-truth concept hierarchy, leading to *systematic prediction errors*. Furthermore, it is quite unstable and often detects spurious changes to the hierarchy. The main issue is that – like other fully automated approaches for structure learning – graphical lasso does require substantial amounts of data to perform reliably, and this is simply not the case in our non-stationary setting. This makes structured learning-based approaches unsuitable for this setup.

MMD outperforms mean embeddings. Another reasonable question is whether recent hypothesis tests outperform MMD for drift detection. Motivated by this, we replace MMD in TRCKD with Mean Embeddings (ME), a state-of-the-art kernel-based discrepancy between distributions [89]. Importantly, the discrimination power of ME can be optimized for the task at hand by performing gradient ascent on an independent training set. In our experiment, we carried out this optimization on the pre-drift examples (i.e., on the first 100 examples in the stream for H-STAGGER and H-EMNIST and on the first 170 for H-20NG). We also used the very same tensor product kernel for both MMD and ME, and for ME we tuned the width of the

Gaussian kernel k_X over instances along with the discrimination power of ME.

Despite being more powerful than MMD on paper, ME did not perform as well in our tests. In particular, ME turned out to be overly sensitive and had severe false detection issues. In practice, ME tends to detect three to four times as many drifts as MMD. For instance, the average number of changepoints detected by TRCKD +ME for the case of H-EMNIST with sequential drifts is 15.25 ± 1.0 compared to 4.25 ± 0.5 of TRCKD +MMD. The ME final hyperparameters used for this experiment are the number of witness points $J = 15$ and $\alpha = 0.01$. This overly sensitivity makes it inadequate for interacting with the user: indeed, querying the user too frequently is likely to rapidly make her lose interest in the interaction in practice.

5.5 Related work

There is an enormous amount of work on concept drift, most of which focuses on single-label [63] and – to a lesser extend – multi-label [197] classification. Surprisingly, drift in hierarchical classification, in which concepts are explicitly mutually constrained, has been so far neglected. In addition, we are not aware of any work on concept drift affecting the concept hierarchy nor on knowledge-aware adaptation strategies. As shown by our experiments, this setup is special enough that standard strategies struggle when applied or adapted to our setting.

The disambiguation step introduced with TRCKD is conceptually related to the problem of drift understanding [107], however works on this topic are unconcerned with hierarchical classification and, therefore, with drift in the background knowledge. To the best of our knowledge, this is the first work that tackles knowledge drift and drift disambiguation and to leverage interaction with a human supervisor to do so.

TRCKD makes ample use of well-known strategies. In particular, it combines ideas from MW-kNN [151], a multi-label kNN approach that adapts passively by discarding old examples, with a proactive detection strategy based on sliding windows. Actively detecting and reacting to drift is key for enabling interaction with the user. Importantly, sliding windows offer distribution-free guarantees on detection accuracy under mild assumptions on the model class and on drifting frequency and speed [92].

5.5.1 Maximum mean discrepancy.

MMD has been applied extensively in domain adaptation [191, 130]. The ME criterion [89], a more recent alternative, underperformed in our experiments but can act as a drop-in replacement for the MMD in applications where it performs better.

The idea of using concrete examples in the windows to explain drift was discussed in [92], although not for MMD. It is true, however, that MMD lends itself to this task. For instance, Kim et al. [93] propose a subset selection procedure for identifying both prototypes (examples that are representative of a particular learned concept) and criticism (examples that, conversely, are not representative), built around a submodular objective defined using MMD. These ideas can be used directly for illustrate drifting concepts to the user and could be generalized to explain the effects of knowledge drift [44].

5.5.2 Drift over graph data.

Drift detection has been studied in the context of graph classification. In this setting, the input data streams encode a sequence of knowledge graphs [123, 179, 185] or an ontology stream [32], and drift affects the distribution of said graphs. In contrast, in our target applications, the machine receives a sequence of examples consisting of a set of subsymbolic observations and of concept annotations, while the knowledge graph controlling the relationship between concepts is completely unobserved. Since KD can only be inferred indirectly by monitoring the examples, this makes drift detection (and disambiguation) much harder.

5.5.3 Open world recognition.

Our work is related to open world recognition, a streaming classification setting in which unanticipated classes appear over time [23]. Open world recognition matches our setting in the case of concept addition, but it is unconcerned with other forms of KD. Furthermore, even though the overall goal is to achieve low error rate on the new, unknown classes [139], most algorithms for open word recognition achieve this by refusing to output any predictions for incoming instances that belong to the unknown classes [138, 136, 23]. In stark contrast, and compatibly with other approaches to concept drift, TRaCking Knowledge Drift aims to adapt the model to new classes rather than reject challenging data points. Other recent work on interactive classification under concept addition [22] focuses on handling noisy labels rather than on adapting to drift.

5.5.4 Novelty and anomaly detection

Our approach shares some similarities with recent work on interactive anomaly detection for structured data [47]. In this work, anomalies are first identified by a machine and then double-checked by a human supervisor. The key idea of leveraging interaction with a supervisor has similar motivations as in our task. More generally, concept drift is related to novelty, anomaly, and out-of-distribution detection, which tackle the problem of identifying unexpected or anomalous datapoints [126]. Concept drift and novelty detection have also been combined [150, 113]. One key difference is that in these settings, the set of concepts is typically fixed. Furthermore, and more importantly, KD – and more generally concept drift – involves updating the model to track changes in the world, whereas no adaptation is typically necessary in novelty detection.

5.5.5 Other topics.

In continual learning, a machine acquires new concepts or tasks over time, and the challenge is to prevent the learned model – typically a neural network – from forgetting previously acquired knowledge [122, 59]. In stark contrast, in our setting the goal is to intentionally forget obsolete information whenever necessary. Moreover, continual learning is unconcerned with forms of knowledge drift other than concept addition, whereas we tackle all forms of KD.

Another related but separate topic is active learning of graphical models, see for instance [164], where the aim is to acquire a multivariate distribution (with a non-trivial factorization) by querying a human-in-the-loop. Specifically, the machine asks for the downstream value of certain variables upon intervention (i.e., manipulating a variable to a chosen value). Individually, these queries are not very informative and it may take tens of queries to acquire a model with a handful of variables. In contrast, TRaCking Knowledge Drift is only interested in obtaining a description of *change* to the concept *hierarchy* – not a whole model, not a distribution over concepts – and to this end it presents the user with an initial guess as guidance and then elicits a one-shot description. These more expressive queries are tailored to the KD use case and lead to more efficient interaction.

Prototypical networks use prototypes representations for each class, which are used to classify a test point [146]. Adding new classes cause catastrophic forgetting, which deteriorates the classifier performance. To avoid this effect, a set of training points that approximate the class mean are kept [129]. This solution is then robust to data representations changes. These two works do not address the concept drift challenge. By estimating the drift in the previous task, it can be compensated in the new task without storing exemplar of previous tasks [182]. The main limitation of these approaches is that they support one prototype for each class (e.g., different libraries are mapped to the same prototype). The solution does not provide a way to erase concepts that do not hold anymore.

5.6 Conclusion

We introduced the problem of knowledge drift in hierarchical classification and proposed to partially offload drift disambiguation to a user. We also proposed TRCKD, an approach for learning under KD that combines *automated* drift detection and adaptation, upgraded to hierarchical classifiers, with *interactive* drift disambiguation. Our empirical results indicate that TRCKD outperforms fully automated approaches by asking just a few questions to the user, even when detection performance is not ideal, showcasing the importance of interaction for handling knowledge drift.

Our work can be improved and extended in several directions. First of all, in practical applications the learner often receives no labels during its normal operation and must acquire any necessary supervision from the user during the interaction. Dealing with this setting requires to integrate a knowledge-aware active learning component into TRCKD. On the user interaction side, we plan to improve the interpretability of our interaction protocol (beyond using examples to illustrate the machine's behaviour to the user) by adapting ideas from explainable AI [44] and explainable interactive learning [141]. Another promising direction is to extend TRCKD to sequential learning methods beyond kNN, especially deep neural networks. The challenge here is to develop knowledge-aware adaptation strategies appropriate for this class of models. It is relatively straightforward to extend our approach to instance-based neural models, see for instance [146]. Knowledge-aware adaptation for other kinds of neural networks could be approached by leveraging recent developments in machine unlearning – so to force the model to forget obsolete concepts and relations – cf. [27] and follow-ups. These extensions, however, are highly non-trivial and left to future work.

Part II

From the lab to the wild

6 Context Recognition Architecture: From Perception Data to Knowledge

Contents

Context recognition in the wild must ensure an alignment between the human's and machine's representation of the world in order to perform well. After presenting the context definition and representation in Chapter 2, we presented the issues that undermine the recognition task, i.e., noisy supervision and knowledge drift (Chapters 3 to 5). The deployment in the real world of the system that performs PCR requires combining and integrating these solutions. To this end, Section 6.2 outlines the reference architecture and describes each component, whereas Section 6.3 lists the procedures that allow the integration and communication of the different components from the collection of the sensor data to the update of the acquired knowledge about the user and the world.

6.1 Introduction

Smart grids are electrical grids in which demand and offer of energy must always be balanced. This task is especially complicated when the grids contain renewable energy sources characterized by discontinuous energy production. Individual consumers can contribute by communicating to the energy provider in which time slots their houses can be disconnected in case of energy shortage in the grid and by allowing the provider to increase or reduce the energy supply based on the person's presence in the house building. The tailored energy supply requires that the system knows the user's behaviour routines. Indeed, the system can learn that the user is not at home between

8 am and 4 pm because he or she is working in the office and that every Monday evening, he or she cooks for her or his friends. This task is achieved by knowing the *situational context* of the user at any moment in time. Other applications like smart personal assistants, smart environments and health monitoring can leverage contextual information to provide their service.

Contextual information is not accessible to the system and needs to be inferred from perception data, such as GPS coordinates, nearby Bluetooth devices and inertial sensors coming from personal devices. Moreover, the semantic of the contextual information is highly subjective. For instance, the same person is a father from the point of view of his daughter and, at the same time, is seen as a professor by his students. However, the world and/or the user unpredictably change over time. Thus, the contextual knowledge acquired by the system over time becomes obsolete, incorrect and inconsistent. For instance, the professor moves to a new university and meets different students. The alignment between the machine's knowledge and the current world and user state is achieved through the bidirectional interaction between humans and machines, in which the continuous exchange of information constitutes the building block to the symbiosis between the two. From the user to the machine, the system leverages the techniques developed in knowledge representation and machine learning to learn to recognize personal context recognition and to handle wrong information provided by the user (Chapters 3 and 5). At the same time, in the opposite direction, the machine provides explanations about its decision (also known as explainable AI) and possibly receives corrections from the user ().

The personal situational context recognition (PCR) task described above can be tackled by combining several components, each one addressing a specific problem: *(i)* machine learning algorithms are trained to recognize the user's context from a stream of sensor data; *(ii)* knowledge representation formalizes the machine's internal representation of the world and user [80], *(iii)* remembering and forgetting policies are implemented to decide which information the system needs to focus on in any moment in time, *(iv)* smartphones and smartwatches are the means for interaction with the user and the perception of the world through sensor data collection.

Contributions. Summarizing, we:

- design a comprehensive architecture that integrates the reasoning and interaction with the user to address the context recognition task. The architecture flexibility allows the adaptation to the specific designer's requirements and thus to several application domains;
- presents procedures and policies that integrate ISGP and TRCKD in a single solution to make the recognition robust to changes in the world.

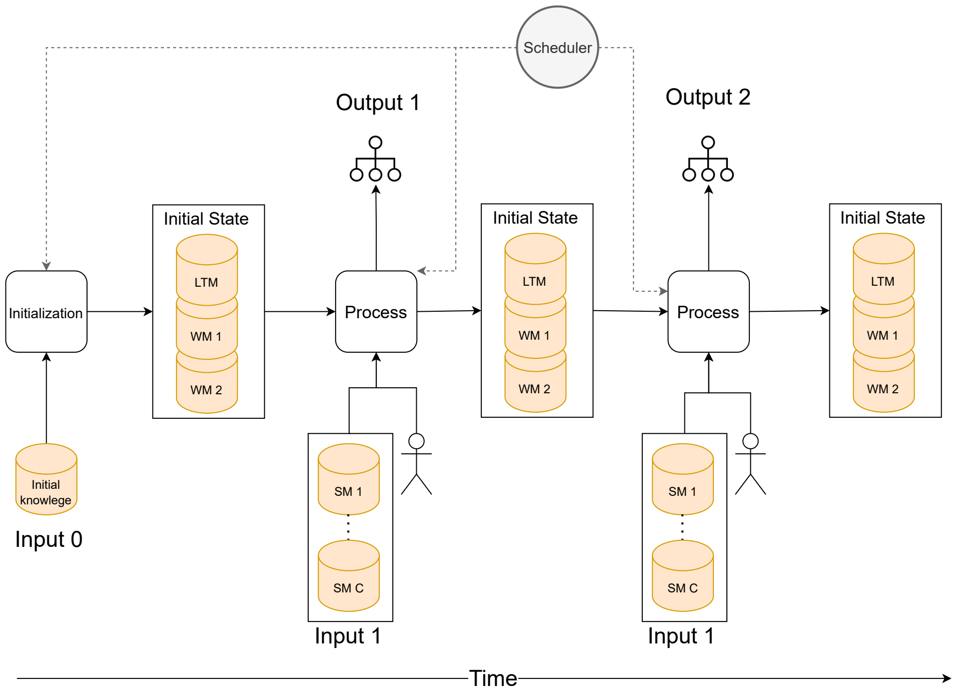


Figure 6.1: For each new input, the output is the function of the previous input's state, which is composed of the LTM and WM, and the current input. The output is a knowledge graph encoding the context of the user when the input was collected.

6.2 Personal context recognition architecture

Given the above introduction of the situational personal context, the context recognition task and its challenges, we present the architecture that supports PCR. The section highlights the statefulness of PCR by making an analogy with recurrent neural networks, in which the current output is influenced by the current input and the previous system state. Then, we describe the four main components: memory, reasoning, interaction and policy.

6.2.1 A recurrent network

Figure 6.1 presents an overview of the execution over time. The flow takes inspiration from what happens in recurrent neural networks: each output is computed from the current input and the "memory" generated by the previous input. However, the difference is that our architecture does not deal with only subsymbolic data like pixels but also considers high-level symbolic data. In our proposal, after the initialization of the memories, at regular intervals, the process outputs the current user's personal context according to the input and possibly after interacting with the user. The current state of

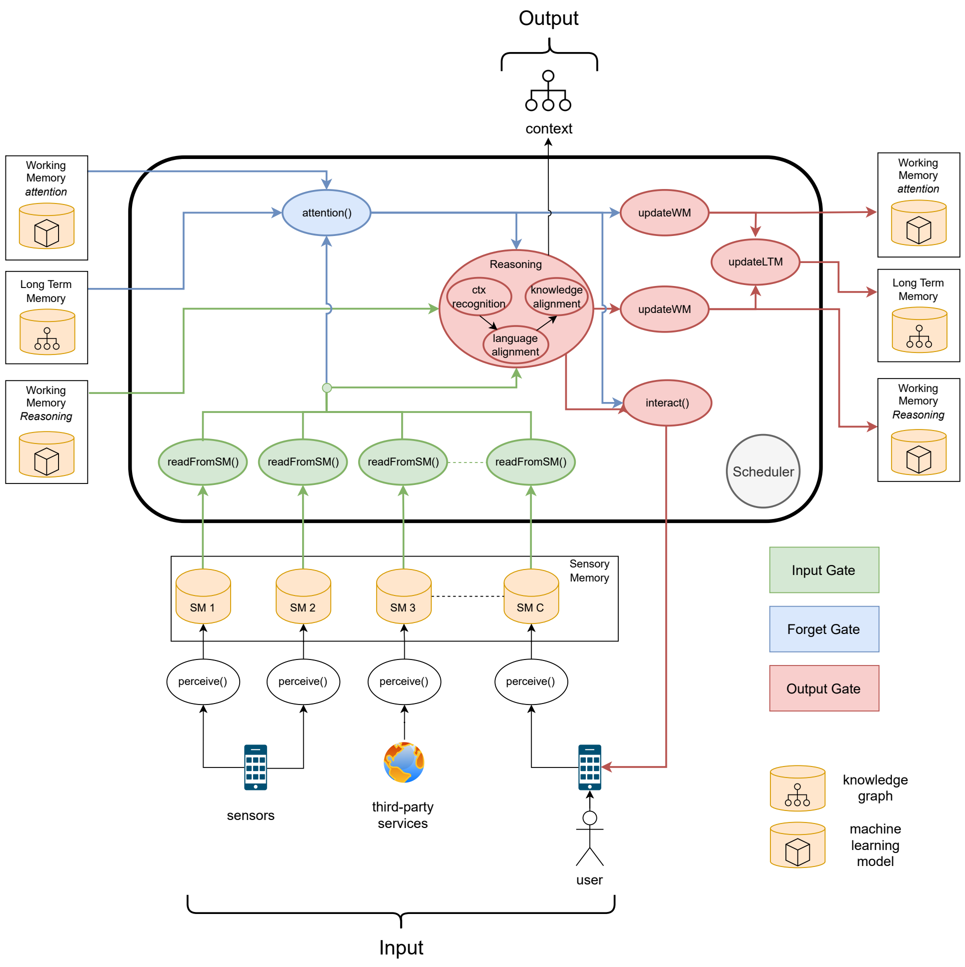


Figure 6.2: Architecture. The components are grouped into *input gate*, *forget gate* and *output gate*. *Scheduler* triggers the execution of each gate.

the system becomes then the input for the next prediction round. Note that the scheduler is responsible for deciding when to move to the next iteration and coordinates the components inside the process.

Figure 6.2 expands the process cell by showing how the different components interact and how the data flows among them. Drawing inspiration from the recurrent neural networks, we divide the components into three types: *Input gate* collects and preprocesses the sensor data; *Forget gate* retrieves the relevant knowledge from the previous state. *Output gate* outputs the context by processing the input data and the retrieved knowledge and, if needed, by interacting with the user. After the computation, the knowledge stored in the memories is updated.

6.2.2 Memory

The system needs to store the knowledge acquired over time and retrieve it for context recognition. Thus, the knowledge is stored in memories, drawn as orange cylinders in Figure 6.2, and their organization takes inspiration from human memory . The memories are the following:

- **sensory memory** (SM) stores the sensed data, i.e., the raw data coming from the user's devices and external sources like OpenStreetMap and open data. The perception of the surrounding environment is represented by the data contained in this memory. Moreover, it stores the user-machine interactions in terms of questions and answers. Each input sense has its own memory, i.e., $\text{SM} = \{\text{SM}^1, \ldots, \text{SM}^{|\mathcal{C}|}\}$, where $\mid \mathcal{C} \mid$ is the total number of input channels. Each SM is a stream indexed by the collection timestamp. The PERCEIVE function (see Figure 6.2) may collect the data at different rates and calls the API of the operating system or external services to retrieve devices sensor data and third-party data. In general, the memory keeps the data until it is processed by the cell process. However, alternative strategies (e.g., first-in-first-out) can be implemented, e.g., by considering the memory size and expiration time constraints. Each input memory can implement different pre-processing strategies, which are defined in the READFROMSM functions, one for each input channel. The function reads the data from the SM and prepares them, for instance, by performing feature engineering and submitting the relevant features used for context recognition to the downstream components.

- **long term memory** (LTM) maintains the built-in knowledge and acquired knowledge. It is structured in three levels, as shown in Figure 6.3:

  - *semantic memory* stores both information about the language and the acquired knowledge about the user and the world. It structures in language, knowledge and data, as proposed in [75]. The language graph (upper part of Figure 6.3) is provided by external services, such as the Universal Knowledge Core (UKC) [70], a multilingual machine-readable lexical resource. It is a hierarchy of words and how they are related to each other. The semantic memory also keeps the entity types, which are typically part of the built-in knowledge at the beginning and is the schema encoding the concepts that model the world and their relations (middle part of Figure 6.3). The lower part stores the instances of the concepts, e.g., the specific friend or the specific office. Over time, both graphs are updated as new information becomes available, e.g., the user's home GPS coordinates or a new colleague. See

  Section 2.2 for the description of entity and entity types. Each entity is linked to the corresponding word in the language section to ensure a mutual understanding between machine and user. The semantic memory stores facts necessary to recognize the context in future moments.
  - *episodic memory*: a sequence of graphs $\{G_0, G_1, G_2, \dots G_t\}$, from the oldest to the current one at time t, that represents the user's personal context at any moment in time as a knowledge graph pointing to the elements in the semantic memory (`occurenceOf` in Figure 6.3). This connection allows the system to retrieve all past occurrences of an entity. The memory also stores the input instance $\mathbf{x}$ used to compute the output context. In this way, $\mathbf{x}$ is accessible by other modules, even in future iterations. For instance, the reasoning component can detect possible data distribution changes (see 6.2.3). In contrast to semantic memory, which contains knowledge used for future context recognition, episodic memory focuses on remembering information about the past.

  The other components access this memory through the ATTENTION function, which retrieves the relevant knowledge using a custom strategy. The other components can update the LTM by calling UPDATELTM: the recognized context is added to the episodic memory and the semantic memory is updated if changes in the knowledge are detected by the reasoning component.

- **working memory** (WM) is the reserved memory of each component to perform its computations and the structure is component dependent. In general, each component can have either a shared memory with others or its own memory. For instance, the reasoning component in Figure 6.2 stores the machine learning model in the working memory. As will be explained in Section 6.3, the ATTENTION function employs a machine learning model, which thus is stored in the reserved WM. Each WM implements a different updating strategy, encoded in UPDATEWM function.

6.2.3 Reasoning

The reasoning component outputs the current personal situational context. As described in Section 2.4, the classifier needs to be robust to annotation noise and changes to the world and the user. The component performs the following steps.

Recognizing the context. Given the knowledge retrieved from LTM and the pre-processed data from SM, the context recognition module outputs the

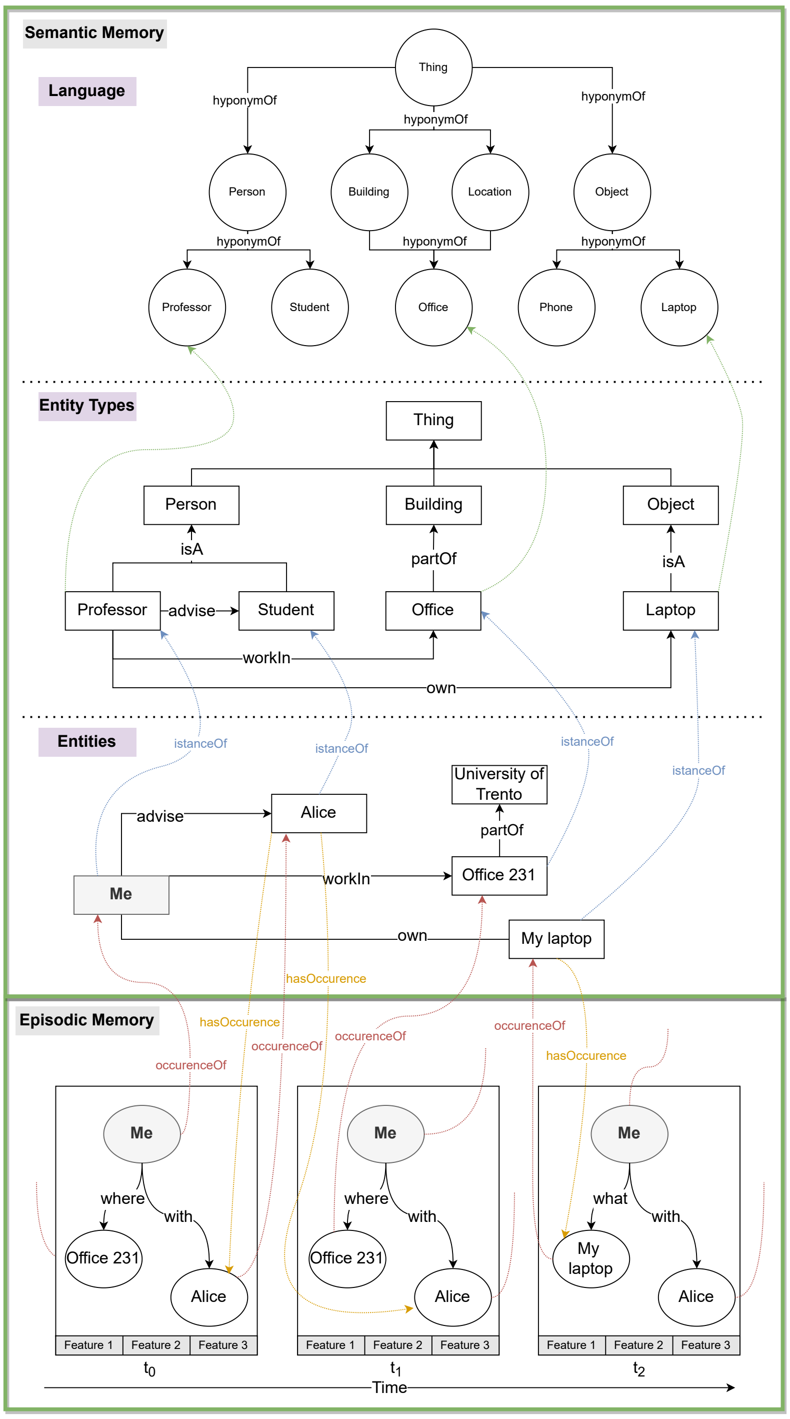


Figure 6.3: Example of the long term memory (LTM). It is composed of semantic memory and episodic memory.

knowledge graph representing the current user's context. External service can exploit the recognized context.

Language alignment. If the machine is not confident about its prediction, it asks the user to answer a few questions about his or her current context in order to improve the predictor performance (see the arrow from REASONING to INTERACT function). A human-machine alignment of the world representation and language is crucial to ensure the mutual understanding between machine and user. The possible inconsistencies between the user and machine labels are: *(i)* different but related words, e.g., they are synonyms or are more/less general meaning; *(ii)* labels have multiple meanings and thus need to be disambiguate; *(iii)* the user label is unknown to the machine; *(iv)* the machine's predicted label is wrong due to either irreducible and generalization error or outdated knowledge (see knowledge drift in Chapter 5); *(v)* the user provides wrong answers due to, for instance, inattention, memory bias or misunderstanding. The annotation mistakes badly impact the PCR predictor performance.

Skeptical learning (SKL), presented in Chapter 3, is an interactive learning protocol that asks the user to revise the annotation when the machine is suspicious about the incoming example $(\mathbf{x}, \mathbf{y})$. Challenging the user is often enough to fix errors due to inattention. Hence, the ground-truth label is retrieved from the user. However, the machine can leverage SKL to fix the inconsistencies described above, and not only unreliable user supervision. The user can disambiguate among the possible inconsistencies that may arise in the language. The machine's suspicion can be estimated using a Bayesian (as in Chapter 3) or frequentist [161] approach. In addition to cleaning incoming data, also past data that elude the skeptical check or are included in the bootstrap data set can be cleaned (as in Chapter 4).

Knowledge alignment. The world and the user change over time, and these changes also impact machine knowledge (e.g., the user relocates to a new country and meets new friends). The changes affect the sensor data distribution (e.g., different home GPS coordinates) and personal context observations. Moreover, it affects the knowledge stored in the LTM, which the machine must align with the user's knowledge. Given that the ground-truth knowledge is unobserved, the challenge is to distinguish among the different types of drift and update the KG and classifier accordingly. In Chapter 5, we presented an approach that combines the drift detection phase, and classifier and graph adaptation with an interactive disambiguation phase. The user knows what drift occurred, and thus, he or she is asked to describe the change. If this is the case, the semantic memory is adapted to match the detected drift: adding and removing entities and properties between them.

6.2.4 Interaction

The personal context is ego-centric. Hence, the human-machine alignment involves continuous interaction with the user. As described above, in case of language divergence, the user is asked to fix it. In knowledge alignment, the only way to disambiguate among different types of knowledge drift, which can have the same footprint on the data distribution, is to interact with the user herself. In these scenarios, the user satisfies the needs of the machine to fix potential inconsistencies between what the machine has acquired and what the user knows. However, the machine cannot bother the user excessively, who may leave the system. Thus, the machine needs to learn when to send the questions in order to maximize the quality and the number of responses, possibly by aggregating several kinds of questions that the machine asks over time. The INTERACT function in Figure 6.2 is the proxy between the user and machine through which the components send the questions. The interaction occurs through close-ended questions with a fixed number of options, free-text input or chat-bot. Regardless of the input type, the language alignment presented above leverages the semantic memory to ensure that the components and the user communicate using common concepts. For instance, the machine knows that “my office” and “office n. 231” refer to the same user’s office.

6.2.5 Execution policy

The execution of the components happens according to the policy implemented in SCHEDULER in Figure 6.2. The scheduler establishes the conditional sequence of the operations based on the output of other components (e.g., the answer of the user triggers the update of the machine learning model). This architecture allows the designer to implement custom policies and adapt them to the available sensors and algorithms. For instance, the policy can define an asynchronous execution. Potentially, the policy can be changed at runtime.

6.2.6 Design principles

Given the explanation above about the components, the principles that guided the architecture design are:

- *evolutionary*: the knowledge stored in the memories can be expanded to remember new information. The machine operates in a lifelong setting, which means that inputs, namely sensor data and user interactions, arrive over time, partially and unordered. Each component needs to evolve over time based on the new data. For instance, the context recognition component needs to implement an incremental training step to incorporate as soon as possible the new available sensor data and labels. The knowledge stored in the LTM is promptly updated after detecting obsolete statements.

- *adaptability*: the components should be robust to noisy data and knowledge change and thus adapt to reduce future misprediction. The user is continuously involved in helping the machine to fix and adapt.
- *compositionality*: the architecture considers the variability in the kinds of data, algorithms and composition. The parametrization of each component helps to define custom policies to tackle new problems and customization to other application domains. New components can be added easily. For instance, a new SM and pre-processing procedure are instantiated if a new sensor becomes available.
- *modularity*: each component can be easily replaced with another, implementing a different computation. Indeed, context recognition, language and knowledge alignment can be implemented using different ML algorithms.

6.3 Personal context procedures

In the previous section, we presented the overall architecture of the context recognition task, and we described the purpose of each component. This section's aim is to provide a possible architecture implementation The system designer can choose different procedures according to the task at hand. In the rest of this section, we detail, in a top-down approach, the pseudo-code of all components.

6.3.1 Scheduler

The scheduler defines the sequence of operations, and Algorithm 4 presents a sequential scheduler. The first step is the initialization of working memory, long-term memory and sensory memory with a given a-priori knowledge graph G_0 and machine learning parameters θ_0. Next, READFROMSM retrieves the raw input data from SM, given the set of input channel $\mathcal{C}$ and the functions $\mathcal{F}$ to pre-process the loaded data. For each new input $\mathbf{x}$, the ATTENTION function computes the relevant subgraph H of the LTM given the pre-processed input data $\mathbf{x}$ (line 5). This computation is based on an ML algorithm, which model is stored in the working memory (line 6). The data retrieved from SM and LTM are the input of the context recognition machine learning algorithm (line 7), which outputs the personal context knowledge graph $\hat{\mathbf{y}}_t$. The language alignment fixes language inconsistencies between user and machine, outputs the updated SKL model θ^r and revised context $\mathbf{y}'_t$. These outputs encode the changes to apply to WM ^{r} and LTM (lines 9

Algorithm 4 Sequential scheduler.

```
Inputs: set of input channels $\mathcal{C}$,
        input data aggregation time frequency per channel $\mathbf{k} \in \mathcal{R}^{|\mathcal{C}|}$,
        set of pre-processing functions applied to each input channel $\mathcal{F}$,
        a-priori knowledge graph $H_0$ and initial model $\theta_0$,
        time interval between each loop execution $j$

 1: $\mathrm{LTM}, \mathrm{WM}^r, \mathrm{WM}^m, \mathrm{SM} \leftarrow$ INITIALIZATION$(H_0, \theta_0, \mathcal{C})$
 2: for $i = 1, 2, \ldots$ do
 3:     $\mathbf{X} \leftarrow$ READFROMSM$(\mathcal{C}, \mathcal{F}, \mathbf{k}, \mathrm{SM})$
 4:     for every new example $\mathbf{x}_t$ in $\mathbf{X}$ do
 5:         $H, \theta^m \leftarrow$ ATTENTION $(\mathbf{x}_t, \mathrm{WM}^m, \mathrm{LTM})$
 6:         $\mathrm{WM}^m \leftarrow$ UPDATEWM$(\mathrm{WM}^m, \theta^m)$        ▷ update attention model
 7:         $\hat{\mathbf{y}}_t \leftarrow$ CTXRECOGNITION$(\mathbf{x}_t, \mathrm{WM}^r, H)$        ▷ context prediction
 8:         $\mathbf{y}'_t, \theta^r \leftarrow$ LANGUAGEALIGNMENT$(\mathrm{WM}^r, H, \mathbf{x}_t, \hat{\mathbf{y}}_t)$ ▷ SKL (Ch. 3)
 9:         $\mathrm{WM}^r \leftarrow$ UPDATEWM$(\mathrm{WM}^r, \theta^r)$
10:         $\mathrm{LTM} \leftarrow$ UPDATELTM$(\mathrm{WM}^r, \mathrm{LTM})$
11:     $\hat{H}, \theta^r \leftarrow$ KNOWLEDGEALIGNMENT$(\mathrm{LTM}, \mathrm{WM}^r)$      ▷ TRCKD (Ch. 5)
12:     $\mathrm{WM}^r_i \leftarrow$ UPDATEWM$(\mathrm{WM}^r, \theta^r)$
13:     $\mathrm{LTM} \leftarrow$ UPDATELTM$(\mathrm{WM}^r_i, \mathrm{LTM})$
14:     SLEEP$(j)$
```

and 10). After processing all new input $\mathbf{x}$ and before starting a new iteration, the next step is to detect possible inconsistencies between the stored knowledge and the one acquired with the new inputs (line 11). Finally, the working memory and the long-term memory are updated and the process restarts.

6.3.2 Initialization

Algorithm 5 initializes the SM, LTM and WM before starting the execution the first time. The LTM is initialized with a given a-priori knowledge graph H_0, which, e.g., contains a large multilingual resource and initial knowledge assertions about the user derived from an initial questionnaire. For each input channel, a sensory memory is created. A WM is initialized for each component that needs stateful computation. In our scenario, both ATTENTION and REASONING store their ML models in a WM, which are initialized with the initial model's parameters.

Algorithm 5 The INITIALIZATION procedure. The initial knowledge graph G_0 contains prior knowledge about the user and the world, θ_0 are the initial model parameters, $\mathcal{C}$ is the set of input channels.

```
Input: $H_0, \theta_0, \mathcal{C}$
Output: $\mathrm{LTM}, \mathrm{WM}^r, \mathrm{WM}^m \mathrm{SM}$
1: $\mathrm{LTM} \leftarrow \{H_0\}$                    ▷ long term memory
2:
3: for $c$ in $\mathcal{C}$ do
4:     $\mathrm{SM}^c \leftarrow \varnothing$                    ▷ sensory memory
5:
6: $\mathrm{WM}^r \leftarrow \theta_0$                    ▷ working memory of REASONING
7: $\mathrm{WM}^m \leftarrow \theta_0$                    ▷ working memory of ATTENTION
```

Algorithm 6 The PERCEIVE procedure reads data from the user's devices and external sources and stores them in the sensory memory. The maximum size of the memory is defined by $\mathbf{m} \in \mathcal{N}^{|\mathcal{C}|}$. If the memory reaches the maximum size, the oldest data is deleted to accommodate the newest.

```
Input: $\mathrm{SM}, \mathcal{C}$, $\mathbf{m}$
Output: SM
1: $\mathcal{C} = \textsc{chooseChannel}(\mathcal{C})$
2: for $c$ in $\mathcal{C}$ do
3:     $S^c \leftarrow \textsc{readNewValuesFromSensors}()$
4:     $\mathrm{SM}^c \leftarrow \textsc{slide}(\mathrm{SM}^c, S^c, m^c)$
```

6.3.3 Perceive the world and the user

The PERCEIVE function loads the raw data from device-specific API or third-party services and stores it into the SM, ready to be processed. Algorithm 6 describes the steps. First, a subset of the input channels is chosen according to the designer's choice or dynamically, e.g., based on which data are more informative for the prediction task at hand. Then, for each channel, new sensor data are stored in SM. If the maximum memory size is reached, the oldest examples are removed to accommodate new data (line 4). The algorithm execution is not triggered by the scheduler but by data availability in the external input devices. For instance, the sensor data transmission from mobile devices and the server is conditioned by the availability of WiFi networks. At the same time, devices with low memory capacity send data more frequently to avoid memory saturation. Given the reasons above, whenever new data are available, they are loaded into the sensory memory. Thus, this task is performed in parallel with the scheduler algorithm (Algorithm 4).

Algorithm 7 The READFROMSM procedure retrieves the data from the sensory memory, generates the features and returns the matrix containing the computed features, which will be stored in the working memory. A subset of channels and feature functions are chosen. $\mathcal{F} = \{f^1, \dots, f^{|\mathcal{C}|}\}$ is the set of feature engineering functions.

```
Inputs: $\mathcal{C}, \mathcal{F}, \mathbf{k}, \mathrm{SM}$
Output: $\mathbf{X}$
1: $\mathcal{C}_t$ = CHOOSECHANNEL($\mathcal{C}$)
2: for $c$ in $\mathcal{C}$ do
3:     $\mathbf{X}^c \leftarrow f^c(\mathrm{SM}^c, k^c)$
4: $\mathbf{X} = [\mathbf{X}^1, \dots, \mathbf{X}^{|\mathcal{C}|}]$
```

Algorithm 8 The ATTENTION procedure extracts from the LTM the most relevant parts according to the current input data.

```
Input: $\mathbf{x}, \mathrm{LTM}, \mathrm{WM}$
Output: $H, \theta$
1: $\theta \leftarrow$ GETMODEL(WM)                  ▷ retrieve the model
2: $L \leftarrow$ GETLOCATIONS(LTM)   ▷ retrieve spatial information from LTM
3: $\hat{y} \leftarrow$ PREDICT($\theta, L, \mathbf{x}$)          ▷ user's location prediction
4: if uncertain about $\hat{y}$ then
5:     $\tilde{y} \leftarrow$ INTERACT($\mathbf{x}, \hat{y}$)                 ▷ request label
6:     $\theta \leftarrow$ TRAINMODEL($\theta, \mathbf{x}, \tilde{y}$)
7: else
8:     $\tilde{y} \leftarrow \hat{y}$
9: $H \leftarrow$ SCOPE($\tilde{y}$, LTM)
```

6.3.4 Relevance computation: load data from the memories

Data are loaded from SM and LTM before the reasoning step. The READFROMSM function, presented in Algorithm 7, retrieves new sensor readings. Since the input data are structured as an infinite stream, the data are discretized in time windows of size k^i by the aggregating function f^i, where i is the i-th channel. For instance, every 30 minutes, the average latitude and longitude of the user in the last 30 minutes are computed. Please note that each channel can have a different aggregation frequency and pre-processing function.

The ATTENTION function, detailed in Algorithm 8, selects the most relevant part H of LTM given the current user's location predicted from the sensory data. The user's location is a strong prior over the set of persons and activities that can be recognized at a certain moment [35]. First, GETMODEL retrieves the machine learning model stored in WM. The GETLOCATIONS extract spatial information from LTM, namely entities, entity types and relations referring to locations, which are the target classes of the PREDICT function. PREDICT outputs the most probable location $\hat{y}$. Then, the machine has to decide whether to request the location label of $\mathbf{x}$. Intuitively, if the machine is uncertain of its prediction, it queries the user. The query decision can be implemented as a randomized choice (e.g., Equation (3.8) of ISGP) or by fixing a threshold on the probability of $\hat{y}$. The implementation highly depends on the underlying ML algorithm used to recognize the location. If the user is queried, then the model is incrementally trained with the requested label. In alternative to the user interaction, the location can be retrieved from third-party geolocalization services using GPS coordinates.

Given the location of the user, the SCOPE function retrieves the set of entities and relations in the LTM (language and entities) that were true at least once in a certain location by navigating the sequence of past personal contexts in the episodic memory. These entities and relations are the target classes of the context recognition model, and this means that the probability assigned to the other entities and relations is zero. These classes refer to the social context and activities. For instance, all persons who have been at the user's home at least once are the target of the predictor when the user is at home. This approach reduces the number of possible target classes of the context prediction task, given that the number of entities and relations in our lifelong setting increases over time. If the set of retrieved target classes given the location is outdated (e.g., user's *roommate* concept is not retrieved when her location is *home*), then, during the knowledge alignment phase, the LTM will be updated by adding the missing or outdated entities and relations to the current location. In addition to the entities and relations, their past occurrences, i.e., the input instances $\mathbf{x}$, are also retrieved and made available to the reasoning component (see Section 6.2.2). For instance, in the example above, SCOPE retrieves all $\mathbf{x}$ training examples collected when the user was at home. Context recognition and language alignment train the machine learning models on these instances.

6.3.5 Reasoning

The reasoning component is the core of the architecture. It performs context recognition and ensures user-machine language and knowledge alignment. The input of CTXRECOGNITION is the relevant part of the LTM H and the current input $\mathbf{x}_t$ and the output is the personal context knowledge graph, i.e., location, activity and social context.

The LANGUAGEALIGNMENT function takes in input the models stored in WM^r, the relevant graph H, the current instance $\mathbf{x}_t$ and machine's labels $\hat{\mathbf{y}}_t$. This function implements the skeptical learning algorithm (refer to Chapter 3), which asks the user to provide labels about the context when the machine is uncertain about its prediction. If the machine is suspicious about the answer, the user is contradicted and asked to revise his or her feedback.

Algorithm 9 UPDATEWM

```
Input: WM, $\hat{\theta}$
Output WM
1: WM $\leftarrow$ WM $\cup \{\hat{\theta}\}$
```

Algorithm 10 UPDATELTM

```
Input: WM, LTM
Output: LTM
1: $G \leftarrow$ COMPUTEDIFFERENCE(LTM, WM)
2: LTM = LTM $\cup \{G\}$
```

The language alignment ensures that the machine and user understand each other, and it is performed on each new input instance in $\mathbf{X}$. The output is the revised context $\mathbf{y}'_t$ and the updated model θ^r.

After having aligned the language of all the current new input instances in $\mathbf{X}$, KNOWLEDGEALIGNMENT detects possible knowledge drifts and, in case, interacts with the user to adapt the knowledge $\hat{H}$ (*cfr.* Chapter 5). The input is the current knowledge structure, which is stored in LTM, and compares the data distribution of past instances, stored in the LTM and the most recent instances, stored in reasoning working memory $\mathbf{m}^r$. If the two data distributions diverge, the user is asked to modify the knowledge graph and adjust it if some entities or relations are no longer valid.

6.3.6 Memories update

Algorithm 9 describes UPDATEWM, which updates WM with the new model parameters $\hat{\theta}$. Algorithm 10 details the UPDATELTM function. First, COMPUTEDIFFERENCE function computes the difference between the knowledge graph modified by the reasoning component and stored in WM, and the graph in the LTM. The graph with the differences is used to update the semantic memory by propagating possible knowledge drift and the episodic memory by appending a new personal situational context.

6.4 Related works

Zhang et al. [195] proposes an architecture to integrate SKL with an existing backend infrastructure for mobile sensor data collection [186], also used in our experimentation in the next chapter. Our architecture also deals with knowledge drift and memory management, combined with procedures to let the various components communicate among them.

Over the years, many works have presented frameworks to support the development of context-aware applications [5], context-aware middleware on mobile devices [184], platforms to collect context information [58] or simply sensor data or administering questionnaires [34, 29, 153, 169, 115]. Coutaz et al. [41] propose a framework that combines a direct state graph that models the transitions between contexts and a general-purpose architecture for the context-aware application. The latter is structured as multiple levels of abstraction: a sensing layer dealing with numeric observations, a perception layer handling symbolic observations, a situation and context identification layer that identifies the context and the changes in the context, and an exploitation layer which allows applications to exploit the context information. Here, the shifts are considered in terms of moving from one context to another, causing changes in the set of entities and surrounding environment. However, they do not consider the context recognition task in the wild and potentially, our architecture can be seen as a plugin to enhance these systems. Moreover, they do not handle the knowledge, and their use cases are not the lifelong user-machine alignment.

Context recognition can suffer from the increasing number of possible target classes of the context dimensions (e.g., the increasing number of activities learned over time). This is especially problematic when the machine needs to discriminate among the possible activities that look similar in terms of sensor data [35]. For this reason, we introduce the relevance computation to perform the recognition only considering the relevant knowledge (Section 6.3.4). In [132], the recognition system uses ontological reasoning based on the user's location to restrict the possible activities identified by the activity recognition module. Out architecture can support different selection strategies with the aim of recognizing all context dimensions and constructing an egocentric knowledge graph. In addition to restricting the target space, the contextual information can be used to select the sensors relevant at a certain point in time [17].

6.5 Conclusion

We presented a modular architecture to support context recognition in the wild by ensuring language alignment through ISGP and knowledge alignment through TRCKD. The architecture design takes inspiration from the recurrent neural network in order to highlight the sequentiality of the context recognition task. Indeed, at any moment in time, the prediction is inferred using only the relevant part of the acquired knowledge and the current sensor data. The former is crucial in a lifelong user-machine symbiosis, given that the information and knowledge grow over time.

Limitations. One drawback of this architecture is that it might be difficult to test the whole execution flow. The interaction with the user happens through question-answers iterations. In this work, we do not make any assumption about the answer format, which can be a label out of a fixed list of options, free text or audio recording. Additional effort is needed to convert the input into a format understandable by the machine. This work does not detail the memory structure because is part of another research direction. Future work needs to define possible forgetting policies of LTM, such as removing entities in the semantic memory that are no longer used in the context recognition task (e.g., the girl I met only once in a bar 10 years ago).

7 Skeptical Learning evaluation in the wild

Contents

Skeptical learning (SKL) addresses the problems of learning in the wild, i.e., noisy supervision and task shift. In Chapter 3, we presented ISGP by evaluating it on synthetic and real-world data. However, this evaluation has been run in a controlled environment. Namely, we were able to measure the method performance in isolation, removing external factors like missing data and supervision, and by having available an oracle. In this chapter, we move the evaluation outside the lab by making the real users interact with the machine. We test the algorithm in a social science research scenario, described in Section 7.2. In this setting, ISGP contributes by reducing the number of questions to the user and ensuring a high-quality answer. To this end, we introduce in Section 7.3 a simplified and working prototype of the architecture presented in the previous chapter. This implementation integrates ISGP into the existing data collection infrastructure allowing it to interact with the experiment participants and to obtain the personal device sensor data. The evaluation involved university students who were asked to install an application on their devices and to answer questionnaires over a period of four weeks. We describe the experiment design in Section 7.4, whereas Section 7.5 analyzes the collected data and the results.

7.1 Introduction

Personal assistants support and give suggestions to their users based on past observations. Researchers in social sciences observe the participants in longitudinal studies for a period of time by sending questionnaires through their personal devices. In these seemingly distant scenarios, the observations are information about the user context collected from the person (see 2.1). The context allows, in the former case, to train a model that provides the user with the correct and timely suggestion and, in the latter, to infer social practices and routines. They also have in common the trade-off between the number of questions and the granularity of the collected information. Too many questions cause the "respondent burden", whereas an insufficient number of questions does not allow the personal assistant and the social researcher to collect information with needed granularity. Moreover, as outlined in Chapter 3, the answers can be noisy and thus affect the quality of the data.

To solve the quality problem, we introduced ISGP in Chapter 3 to detect possible suspicious answers and to revise them. The work on ISGP has been evaluated on synthetic and real-world data sets, and there, an oracle simulates the user annotator by providing the answers to the questions and contradiction of the machine. This setting introduces assumptions about the user and the data necessary to evaluate the algorithm w.r.t. state-of-the-art competitors in isolation from external influences. For instance, the examples arrive in an ordered sequence, and the user promptly answers the questions without considering the user may leave the experiment or some sensor data is not received because the sensor is disabled by the user (e.g., GPS coordinates) or not available on the device. The issues impact the predictive performance of the model. However, a solution that relies on user interaction cannot be exempted from an evaluation with real users in the wild. Most of the studies on interactive machine learning miss an evaluation with real uses because it is time-consuming, expensive and requires a multidisciplinary approach (e.g., sociologist, experimental psychologist, expert in human-computer interaction) .

To evaluate ISGP in a real-world scenario, we apply ISGP to the use case of longitudinal studies in social sciences run through smartphones. The expected benefits of ISGP in this use case are to reduce the effort of the user in filing the questions and an improvement in the answer quality. ISGP learns to answer the questions that are sent at regular intervals, asking about the participant's context and contradicting his or her when skeptical about the answer. We designed and ran an experiment with students of the University of Trento to evaluate ISGP in recognizing the location of the participants. [1] The ultimate goal of ISGP is to reduce the number of questions sent to the user by asking the model to answer the questions about the user's position.

Contributions. Summarizing, in this chapter, we:

- instantiate ISGP part of the reference architecture into a real-working

[1]This experiment involves human subjects and has been approved by the Research Ethics Committee of the University of Trento (protocol n. 2023-006).

prototype and describe the technical solutions;

- introduce a social science use case in which ISGP can contribute improving the data collection in terms of duration of the experiment and data quality;
- design the experiment and run it with university students for one month.

7.2 Use case

In social sciences, researchers are interested in observing human social behaviour in daily life. Using personal mobile devices is a promising direction allowing a continuous data collection that combines the traditional experience sampling method (ESM) [102] with passive sensor data [171]. ESM is a longitudinal study in which the participants are asked to report their feelings, environment or other behaviours multiple times per day. Researchers send questionnaires over time to groups of people to survey routines and social practices by collecting contextual information of the person. The context can be defined by the location of the user, the performed activity and the persons with whom he or she interacts. Originally, these surveys were paper-based and given one or more times a day. Nowadays, as mobile devices become more widespread, surveys are delivered through smartphones and smartwatches, which allow to send questions more frequently over the days and to collect also sensor data. This new collection approach allows for gathering more granular data [25].

However, the increase in the number of questions causes the so called "respondent burden". The participants may stop answering the questions or provide inaccurate answers. In addition, participants might make inadvertent or systematic errors when reporting their activities. This is caused by inattention or, in the more complex cases, by active choices or biases that induce the participants to alter the answer [38]. For instance, the participants modify their behaviour due to the awareness of being observed (Hawthorne effect [114]), do not understand the question, have memory biases [165], are unwilling to report [39], or they report behaviours that are socially desirable, i.e., conditioning.

The benefit of ISGP in this scenario is twofold. First, the machine learning model can answer these questions autonomously reducing the effort of the participants. Second, the quality of the answers can be improved both to obtain better data for the researchers and to train a more accurate model.

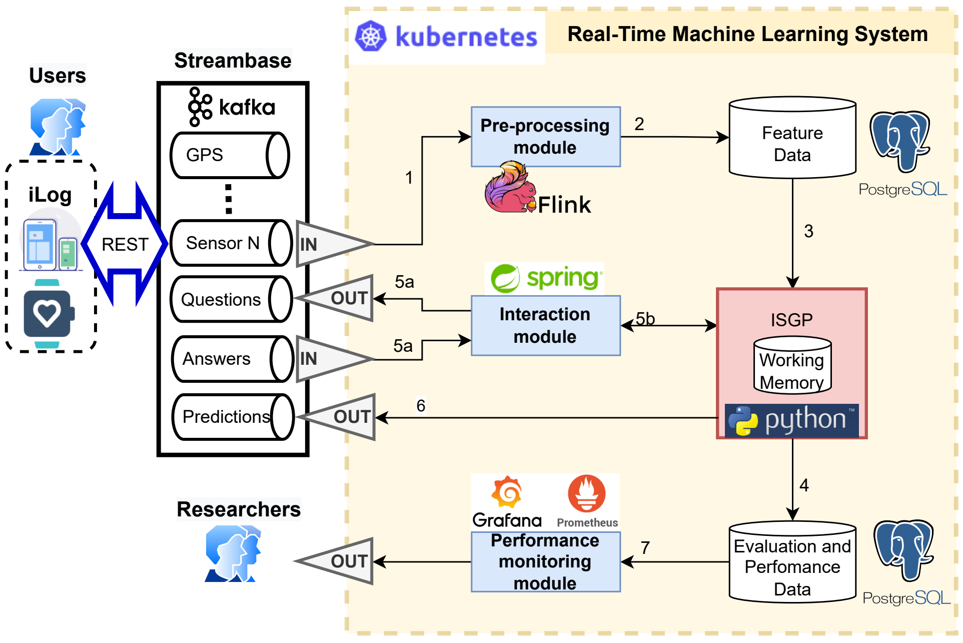


Figure 7.1: Experiment architecture: the yellow dashed box contains the implemented components and their technology. Red box is the learning algorithm, whereas blue boxes are the components that provide support services to it.

7.3 Machine learning architecture

This section outlines the architecture and describes the modules that allow the integration of ISGP with the existing data collection infrastructure (Section 7.3.1). This architecture is the backend system used in the experiment with real-user described in the next section Section 7.4. This system also enables researchers to plug in their algorithms to evaluate them by reusing the basic building blocks presented in this work. Figure 7.1 shows the overall architecture. The yellow area highlights the contribution of this work and the technological stack based on open-source tools. Essentially, the learning algorithms retrieve the sensor readings from the iLog system and, after a pre-processing phase, the data is used to train and recognize the context of the user (i.e., location, activity and social context). The interactive learning algorithm interacts with the user through iLog, and the predictions of the learner are stored locally and sent back to iLog to be reused and shared with external components.

Researchers who want to successfully validate in the real world their ML algorithms need to deploy them in a production environment. To this end, MLOps is the process that automates, operationalizes and maintains machine learning solutions in production [100]. The deployment of the model is only part of the process. Indeed, a lot of effort is required to keep the machine learning system in execution [142]. In traditional batch machine learning, the model is trained offline on a batch of data and then deployed in production. Given the model degradation and concept drift, the model needs to be retrained on new data and redeployed. In our scenario, however, the learning happens in an online fashion, i.e., new data arrives as a stream, and thus, the model continues to make predictions and is updated dynamically. This setting poses additional challenges to the operationalization of the models like the lack of ground-truth labels, which become available with a certain delay, and the risk of introducing performance degradation or errors [9, 10]. The architecture must support real-time training and predictions and thus require additional strategies to support requirements like reproducibility and monitoring [10]. The prototype follows the microservices architectural style [49], in which each module is an independent service, and these services communicate among themselves to provide the expected functionality. Each service is a package, *aka* container, containing the code and all its dependencies and can be on different computing environments. The management and orchestration of these containers are delegated to Kubernetes[2], which "is an open-source system for automating deployment, scaling, and management of containerized applications".

7.3.1 iLog infrastructure

iLog is a system that collects personal information and generates streams of data coming from smartphones and wearable devices [187]. The system is designed to be easily integrated into the daily life of the users by running in the background to avoid affecting their routines. The app supports multiple smartphone models, allowing it to not hinder the participants from joining the experiments. The users install the mobile application on their smartphones and smartwatches and authorize the collection of the sensor values. Through the mobile application, the user answers questions using different formats like text or pictures. In our scenario, we are interested in acquiring information about the context dimensions (see Section 2.1). The application collects hardware sensors such as accelerometer, gyroscope and rotations, and software sensors such as phone calls, running application and battery level. The system allows the data collection designer to configure several aspects such as the collection frequency of each sensor and the questions text and answer options. The data are sent to the backend server, called Streambase [186] in Figure 7.1, which stores the data streams, sends questions to the users and retrieves the answers. Streambase is a collection of input streams to the learning infrastructure and output streams recording the outputs of the learning methods. These output streams allow other ex-

[2]Kubernetes website https://kubernetes.io/

ternal components to leverage the predictions made by the machine learning algorithms. The streams are implemented using Apache Kafka[3], an “open-source distributed event streaming platform”. Each continuous flow of data is called *topic*, which stores the data of either one sensor, questions, answers or the output of the machine learning system. The components writing data on a topic are called producers, whereas the components reading and processing the data are called consumers. The topics contain a stream of data that arrives over time and is processed incrementally by the machine learning algorithms.

7.3.2 Pre-processing of the sensor data

The pre-processing module performs cleaning steps on the raw sensor data and generates the features, which are sent to the machine learning algorithm. The user’s devices send the sensor data to the Kafka topics in Streambase (*arrow 1* in Figure 7.1), and then the module aggregates the data into time windows, e.g., of 30 minutes, and stores these feature vectors in the feature store (*arrow 2*). In this work, we use PostgreSQL[4]. The feature vector contains statistics about the sensors, e.g., the number of calls and the centroid of the GPS coordinates. Moreover, it validates and cleans the data by fixing errors or missing values. Given the streaming nature of the sensor data, i.e., the data are continuously generated and processed, we leverage Apache Flink, which “is a framework and distributed processing engine for stateful computations over unbounded and bounded data streams”.[5]. This solution is considered as data engineering best practice [10]. Researchers need to define which and how the features are created since they have an impact on the performance of the ML algorithm. The throughput of the module is crucial since the amount of data generated by each user is high, and thus, the computational and memory demands must be tuned accordingly.

7.3.3 Learning module

Multiple ML algorithms can be deployed in parallel to address different tasks. For this experiment, we focus only on ISGP. The goal of machine learning modules is to recognize the user context from a continuous stream of feature vectors. Thus, the learners implement an incremental learning strategy to update the learned model over time efficiently. Since the user’s context is egocentric, i.e., it encodes the user’s point of view, each user has its own model, and the data of one user is not used to train the model of another. Also, the hyperparameters of the ML algorithms are tuned on a per-user basis. Each ML algorithm stores the learned model in internal storage called working

[3]Apache Kafka https://kafka.apache.org/
[4]PostgreSQL website https://www.postgresql.org/
[5]Apache Flink website https://flink.apache.org/

memory. Each time the algorithm performs a training or prediction step for a specific user, it loads the corresponding model from the internal storage and requests feature vectors from the database (*arrow 3*). The stream of predictions can be used by other modules or sent back to the user (*arrow 3*). If we would add TRCKD, then the detected knowledge drifts would represent a new stream.

7.3.4 User-machine interaction

The interactive machine learning algorithm sends questions or messages to the user. In our use case, ISGP sends requests to label incoming examples and contradiction questions to ask the users to revise their label when the machine detects a possible mistake in the answer. The communication with the users occurs through the interaction module (*arrows 5b*) that is responsible for delivering content to Streambase (*arrow 5a*) and then forwarded to the users' devices. The answers follow back the same path. All learning algorithms need to minimize the number of interactions with the users and avoid bothering them too often. This aspect is especially challenging when more algorithms are run in parallel and send multiple questions. This condition may generate a high number of questions, and the user may stop answering them.

7.3.5 Performance monitoring

The researchers need to monitor the learning algorithms over time. To this end, they define the set of relevant metrics to monitor the model performances and these metrics are then stored in the evaluation and performance database (*arrows 4*) and visible to the researchers. They have access to a dashboard that loads and processes these metrics through the performance monitoring module (*arrow 7*). The metrics measure both the model performance, such as F1 score, recall and accuracy, and the infrastructure performance, such as throughput and predictions per second. Evaluating stream learning algorithms is challenging [62, 18]. Moreover, the ground-truth class may not be available or can be delayed [79]. The ML algorithm handles a stream of data, and the training and prediction phases are continuously executed over time as new data becomes available. The monitoring is even more important in this streaming setting because of the dynamic nature and continuous evolution of the model. Indeed, the model is incrementally updated to integrate the new data coming from the user's device. The monitoring module collects and stores all metrics and measurements in Prometheus[6] and visualize them through the monitoring dashboards of Grafana[7].

[6]Prometheus website `https://prometheus.io/`

[7]Grafana website `https://grafana.com/`

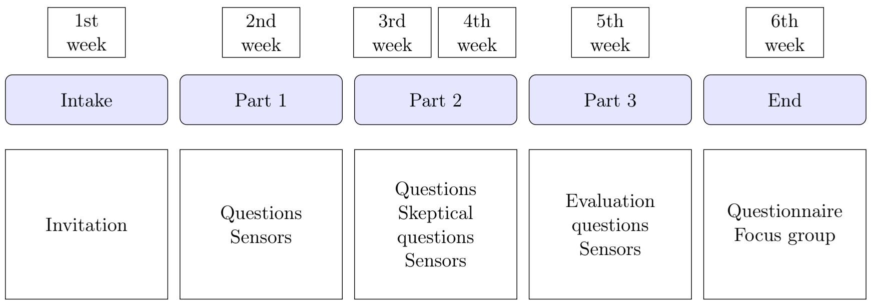


Figure 7.2: Research protocol.

7.4 Experiment design

This section presents the research protocol of the ISGP evaluation experiment developed in a multi-disciplinary team composed of a sociologist and software engineers and involved in previous data collection [72]. This experiment evaluates ISGP on the user case in Section 7.2 to recognize the spatial dimension of the context and thus to answer this question. The experiment focuses only on the spatial dimension for the following reasons:

- the location dimension is easier to recognize by the machine with respect to the other dimension of the context;
- taking into account all context dimensions would require the user to answer questions for each dimension, increasing the user effort and making more difficult the evaluation of ISGP;
- it is possible to compute the ground-truth position from the GPS coordinates of the University of Trento and of the main home. The ground-truth labels can then be compared with the labels provided by the user.

The benefits of applying ISGP in this setting are the reduction of the effort of the participants and the improvement of the quality. This experiment makes it possible to evaluate ISGP in an out-of-the-lab environment.

7.4.1 Research protocol

Figure 7.2 outlines the phases of the research protocol. The overall length of the experiment is six weeks, and four of them are allocated to the data collection

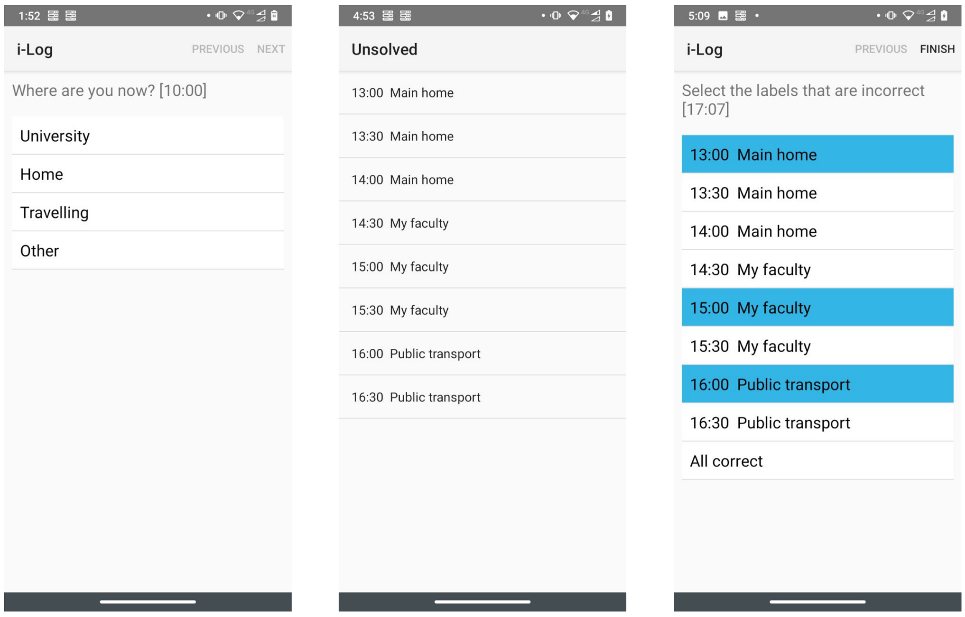
Intake. The students of the Department of Information Engineering and Computer Science of the University of Trento have been contacted through email to present the research project and to provide instructions on how to join the experiment by installing the iLog app on Android devices. The installation on the personal device allows the collection of data that faithfully describes their daily life and does not alter their daily routines. The participating students must read and accept the privacy information through the app and authorize the data collection for each sensor individually. The incentive strategy includes bonuses as follows: €30 to all participants with at least 75% completed questions, prizes of €100 to three randomly selected most active participants and an additional €10 to those attending the focus group.

Part 1. In this first data collection phase, the app starts collecting data from the sensors, and the phase lasts one week. Moreover, every 30 minutes, the app asks the participant's location, which is the context dimension we investigate (Figure 7.3a shows a screenshot of the question page in the app). See the answer options in Table 7.2 with code Q1. ISGP algorithm bootstraps the learning on this data before transitioning to the next phase.

Part 2. For two weeks, the participant continues to answer the time diaries, and in addition, the model starts challenging the participant on suspicious labels. A label is suspicious if it does not equal the predicted labels and the model is sufficiently confident that its label is correct (*cfr.* Chapter 3). The participant can confirm the predicted labels, and in this case, the model was able to detect that the label provided as the answer of the time diary was not correct. If this is not the case, a new answer is provided. The model is refined by considering this participant feedback. The model sends the contradictions every evening at 7 pm all at once to concentrate the answering effort in single and specific periods of time. See how the contradiction question is visualized on the smartphone Figure 7.3b.

Part 3. The goal of the last week of the data collection is the evaluation of the model predictions and, thus, the machine-participant alignment. The participant needs to select the incorrect location labels from a list of predicted labels (e.g., Figure 7.3c). This question is the only interaction with the participant in this phase, and it occurs at 7 p.m.

Conclusion. In the last phase of the research protocol, the participant is expected to fill out a questionnaire to collect socio-demographic information. Among the participants who interact the most with the app by answering the time diaries, ten are selected to participate in a short focus group. This activity investigates general aspects like the participants' interaction with machine learning models and more specific aspects like the interaction with ISGP, how they felt being challenged by the machine and the evaluation of the iLog app.


1:52

i-Log PREVIOUS NEXT

Where are you now? [10:00]

University

Home

Travelling

Other

4:53

Unsolved

13:00 Main home

13:30 Main home

14:00 Main home

14:30 My faculty

15:00 My faculty

15:30 My faculty

16:00 Public transport

16:30 Public transport

5:09

i-Log PREVIOUS FINISH

Select the labels that are incorrect [17:07]

13:00 Main home

13:30 Main home

14:00 Main home

14:30 My faculty

15:00 My faculty

15:30 My faculty

16:00 Public transport

16:30 Public transport

All correct

(a) Time diary **(b)** Skeptical **(c)** Evaluation

Figure 7.3: The three types of questions shown on the iLog app.

7.4.2 Experiment setup

Sensors. The sensor values are continuously collected during the four weeks of data collection. The full list of the collected sensors is presented in Table A.1, which also reports their collection frequency. In this experiment, we use a subset of the 30 sensors supported by the application to select those sensors that are more informative in predicting the location and avoid drying the battery excessively. The streams are temporarily stored on the device and updated on the server periodically.

Features. The raw sensor data are aggregated in time windows of 30 minutes, which is the time between two consecutive annotations, by generating feature vectors to input into the model. The features are described in Table A.2.

Questions. Table 7.1 lists the questions. Time diaries are sent every 30 minutes, and the user is asked to indicate his or her location. The list of location options (see Table 7.2) is derived from the guidelines for time use surveys [56]. To reduce the user effort, the options are aggregated into main categories. Skeptical questions are sent once a day, one for each suspicious answer. One evaluation question is sent every day in the last phase and reports the list of predicted labels. Then, the user has to select the incorrect

	Timing/ Condition	**Question**	**Answer options**
Q1	1st week: every 30 minutes	Where are you now?	see Table 7.2
Q2	2nd and 3rd week at 7:00 pm	Is <*time*> <*predicted label*> correct?	1. Yes 2. No
Q3	if Q2 = No	Where are you at <*time*>?	go to Q1
Q4	4th week at 7:00 pm	Select the labels that are incorrect	1. <*time*> <*predicted label*> 2. <*time*> <*predicted label*> 3. . . . 4. All correct

Table 7.1: Structure of the questions. Q1 is the time diary, Q2 and Q3 are the skeptical questions and Q4 is the evaluation question.

ones. If time diaries and questions are not answered within 8 and 12 hours, respectively, then they expire and cannot be answered.

Hyperparamters. We employ the ISGP presented in Chapter 3, and we use the same hyperparamters of the evaluation in Section 3.5.2.

7.5 Results

This section presents the main results and statistics about the sensor data, interaction with the participants and performance of ISGP. The number of participants that downloaded and installed the iLog application on their devices is 77, 58 uploaded sensor data and answers. During the data collection, we sent a questionnaire to collect demographics. We obtained the data from 42 participants, of which 90% consider themselves male and 10% female. All the participants belong to the Department of Information Engineering and Computer Science of the University of Trento. Most of the participants are pursuing a bachelor's degree (74%), and the remaining a master's degree (26%).

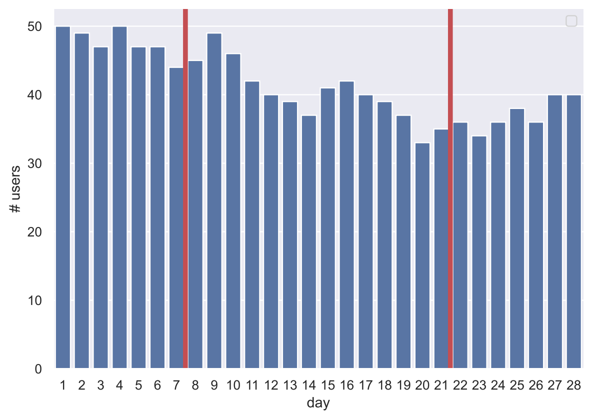


Figure 7.4: Number of users who uploaded sensor data by day of the experiment. Red lines denote the three parts of the data collection.

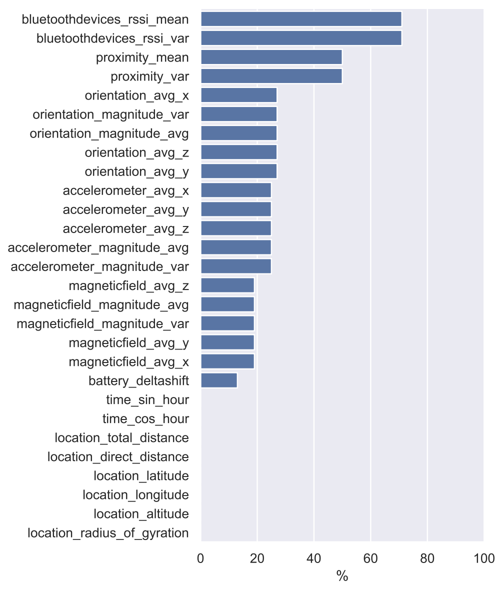


Figure 7.5: Percentage of missing values for each numeric feature.

Main category	**Subcategory**
University	1. My faculty 2. Other faculty (UniTn) 3. Other
Home	1. Main home 2. Weekend home or holiday apartment 3. Other people's home
Travelling	1. Foot 2. Bicycle 3. Moped, motorcycle or motorboat 4. Passenger car 5. Other private transport mode 6. Public transport
Other	1. Restaurant, cafe, or pub 2. Shopping centers, malls, market, other shops 3. Hotel, guesthouse, camping site 4. Street, square, city park 5. Sports center 6. Other

Table 7.2: List of answer options to the time diary question "Where are you now?".

7.5.1 Sensor data

The attrition effects are clearly visible in Figure 7.4 and caused the participants to leave the experiment. During the first week, the server received sensor data from 48 participants, whereas it decreased to 37 in the last week. A second common problem in real-life datasets is the missing values. In this experiment, the percentage of missing values for every numeric feature varies considerably, as shown in Figure 7.5. The reasons for missing values can be unsupported mobile devices or participants actively disabling one or more sensors. For instance, the incompatibility of some Android versions caused a large number of missing values for the features derived from Bluetooth.

7.5.2 Time diaries

The mobile application sends time diaries every 30 minutes. Figure 7.6 shows the distribution of the answer over hours of the day and compares weekdays and weekend days. As expected, the dataset is highly unbalanced given the

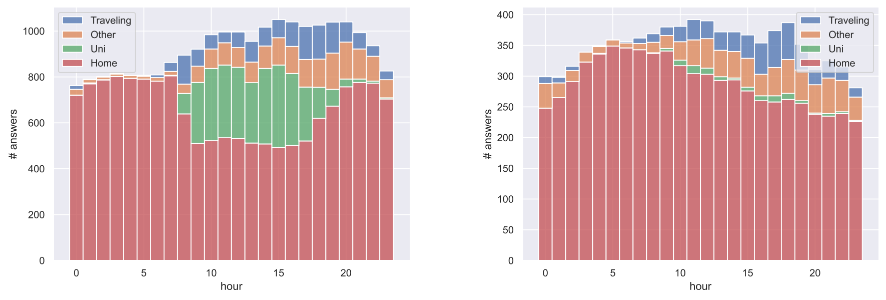


Figure 7.6: Number of time diary answers by hour of the day, divided by weekdays (left) and weekends (right). The answers are aggregated by main categories.

high number of labels related to home (main home, weekend home or other's home). The labels related to university are the ones that vary the most between weekdays and weekends.

Figure 7.7 shows the time diary answers for each user over the first three weeks. Each cell of the heatmap is an interval of 30 minutes, and the color denotes the main category of the answer, i.e., university, home, travelling and other. Blue cells are unanswered questions. Note how the answering pattern varies across users. Top rows represent users who regularly provide answers, whereas bottom users were less active and, at a certain point, left the experiment. Users in the middle alternate days with answers and periods where the question expired.

7.5.3 Skeptical questions

The time diary answers are used to train the ISGP model, one for each user. The model learns the mapping between sensor data and location labels. As described in Chapter 3, mistaken labels badly affect the machine's performance. In phase two of the experiment (second and third week of the experiment), the machine sends skeptical questions to contradict the user whenever it is suspicious about the label. The suspicion is based on the fact that the predicted label and user label are not compatible and the machine is sufficiently confident of its prediction (cf. Section 3.4). Figure 7.8(left) shows the total number of contradictions sent to the user split between answered and not answered. The fraction of missing answers is high, namely more than 50% for most of the days. The main causes are that the question is not delivered because the phone was not connected to the Internet or users did not respond in time. In the 25% of the answers, the machine prediction was selected as correct, whereas in the rest of the answers, the user provided a new label. The 80% of these labels of the latter case matches the label provided before being contradicted.Figure 7.8(right) plots the number of times

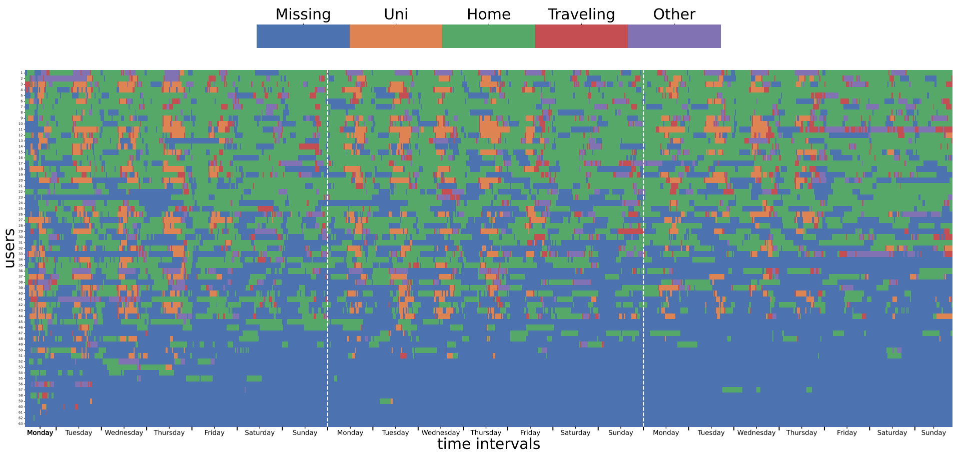


Figure 7.7: The main category of the time diary answers over the first three weeks of the experiment. Rows: all users of the experiment. Columns: time interval of 30 minutes (i.e., annotation). White vertical lines denote each week.

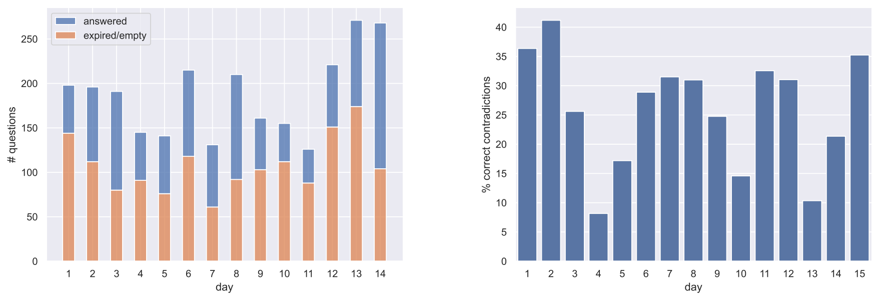


Figure 7.8: Overall statistics about skeptical questions during the two weeks of part 2 of the experiment. **Left**: number of questions with (blue) and without (orange) an answer. **Right**: percentage of skeptical contradiction in which the machine label is confirmed as correct by the user.

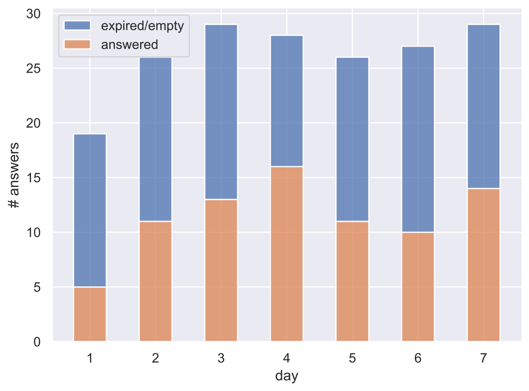


Figure 7.9: Total number of evaluation questions sent to the user for each day of the last week of the experiment. Orange: number of received answers. Blue: number of questions without answer.

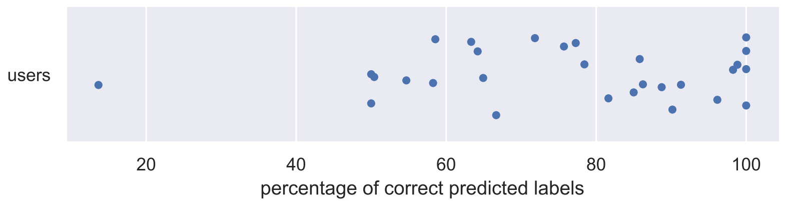


Figure 7.10: Percentage of predicted labels that are evaluated as correct by the user. Each point is a user.

the machine was right.

7.5.4 Evaluation question

In the third and last phase, each day, the participant is asked to evaluate the prediction of the machine. The questions, sent at 7 pm, report the list of the locations labels predicted in the last 24 hours. Then, the user selects those labels that she or he considers wrong. Figure 7.9 shows the number of questions sent to the user for each day of the last week of the experiment. The fraction of unanswered questions is more than 50% (blue bar). In 25% of the received answers, the participants evaluated the prediction of that day as all correct. Each day, participants have to evaluate 30 labels on average. Figure 7.10 details, for each user, the percentage of the correct labels, which average is 76%.

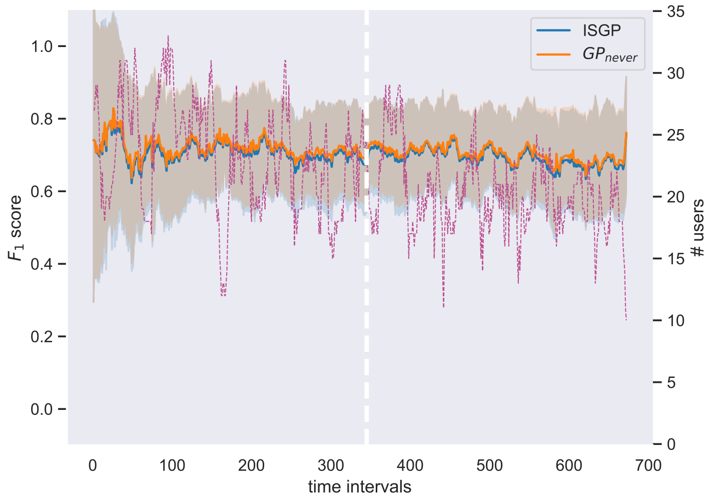


Figure 7.11: Progressive $F1$-score averaged over all users. Time intervals are the 30-minute windows of two weeks. Shaded area is the standard deviation. Violet dashed line represents the number of users. Blue line is the ISGP, as presented in Chapter 3. Orange line: ISGP variant in which the user is never contradicted.

7.5.5 ISGP performance

We compare the effectiveness of ISGP with a variant that never contradicts the user. Figure 7.11 reports the experimental results on these two methods averaged over all users. The plot shows the progressive F_1over the third and fourth week for every 30-minute timeslot. Since the user label, which is used as ground truth, or the input data are not always available, the number of users varies over the time intervals (violet dashed line). The performance of ISGP and GP_{never} are overlapping, which shows that, on average, there was no advantage in being sceptical. Indeed, in only 26% of the contradiction, the machine was correctly skeptical about the participant supervision (see Figure 7.8). The motivations are that the participants were attentive and thus provided correct answers.

7.6 Conclusion

In this section, we designed and executed an experiment with real users in the wild to evaluate ISGP. The experiment was designed around a social science use case. Specifically, social researchers run longitudinal studies in which the participants provide information about their daily life multiple times a day. In this scenario, ISGP is employed during the data collection to reduce the respondent burden by allowing the machine to automatically answer the questions whenever it is confident of its prediction. Here, ISGP ensures that the target labels, which are collected from the user and used to train the model, are correct. The number of questions can thus be reduced, allowing the researcher to increase the duration of data collection and mitigating the drop-out problem. In this experiment, we focused on the redesign presented in Chapter 3. To run the experiment, we proposed and implemented an architecture that integrates ISGP learning methods in an existing data collection platform, namely the iLog infrastructure.

Limitations. The results of this experiment cannot be generalised because the participant's sample is small and includes only students from one university department. In this experiment, ISGP has some difference with respect to the version presented in Chapter 3. First, we removed the active queries, and thus, during the first three weeks, time diaries were sent all the time to ask for the label on all examples without considering the machine prediction confidence. This was necessary because the sensor data are not sent in real-time to the model on the server. This is also needed to collect a sufficient number of labels to use as ground truth for the model evaluation. Second, skeptical questions were sent at a fixed time because it was not possible to send them close to the time diary they referred to. The reasons are, as above, that the sensor data are received when the participant uploads them and to avoid further disturbing the participant during the day. Of course, this can introduce some memory biases, like forgetting the location where the participant was at a specific time. The evaluation questions were sent in the last week of the experiment and are meant to allow the user to select wrongly predicted labels. However, participants may select only a subset of the mistakes to avoid going through the full list of predictions. Regarding skeptical questions, the participants may feel frustrated by the contradiction and thus always reject the machine label. Future works should investigate these psychological and behavioural implications that arise from interacting with a machine that learns about your routines and contradicts your answers.

Discussion. The results show that in this experiment, the participants were mostly consistent in the provided supervision. This might be explained by the reduced impact of the respondent burden due to the short experiment period and the fact there was only one question, which investigated the location. In this scenario, being skeptical about the labels does not have an impact on the model performance. However, the key point is that the participants rated as correct the majority of the labels predicted in the evaluation week, confirming the benefits of employing the model in longitudinal studies. The benefits of sceptical learning arise when running longer data collections with multiple questions. In addition, we observed a low fraction of answered questions. Future experiment design should consider the factors that impact the answering of the questions [19] and have better scheduling of the questions [196]. The model performance can be improved by implementing a per-user hyperparameter tuning, which adapts the model to the specific user. To this end, recent works proposed solutions for hyperparameter tuning on data streams [172, 28] and AutoML for online learning [30].

8 Conclusion

Contents

8.1 Summary

In the first part of this work, we presented three algorithms to fix inconsistencies in the data with the goal of performing personal context recognition in the wild. Namely, the methods are robust to language and knowledge changes that occur over the lifelong interaction between user and machine.

In Chapter 3, we proposed a solution to two challenges of deploying interactive learners like personal assistants in the wild, i.e., noisy supervision and the number of classes increases over time. To address these challenges, we proposed a redesign of Skeptical Learning named ISGP. The method builds on Gaussian Processes to estimate the uncertainty used to ask the annotator to reconsider her feedback when confident that an example is mislabeled. ISGP improves over the previous methods by *(i)* labeling examples far from known training examples, *(ii)* avoiding to become overconfident and thus continuously contradicting the user, *(iii)* being a simple approach that requires find-tuning or difficult hyper-parameter setting and *(iv)* learning incrementally to incorporate new classes and examples. Our experimental evaluation on synthetic data and real-world context data showed that ISGP works well in terms of query budget allocation, prediction quality and efficiency on different levels of noise and as new classes are observed.

In Chapter 4, we extended the work in the previous chapter by allowing the annotator to fix the noisy examples that eluded the cleaning step, e.g., during the initial bootstrap phase. Having noisy training examples impacts the predictive performance and also the ability to spot future noisy supervision of examples that fall in regions affected by the noise. Thus, we proposed CINCER that, whenever it detects a suspicious example, select a set of counter-examples in the training set that are maximally incompatible, from the point of view of the model, with the suspicious example. In this case, the user can provide relabel both or either the examples to resolve the inconsistency. The counter-examples serve as an explanation of the model suspicions and allow the user to fix the reasons for the suspicion. Leveraging influence functions, CINCER retrieves counter-examples that explain why the example looks spacious, improve the model as much as possible if fixed and are easier to interpret by the annotator. The method has been evaluated on diverse data sets. The work focuses on visual inputs, which makes it straightforward for the annotator to detect the inconsistency. In the case of sensor data, the examples should be processed to make them understandable. For instance, the GPS coordinates can be displayed on a map. This aspect is left as future work.

In Chapter 5, we introduced and focused on knowledge drift, a special form of concept drift that occurs in hierarchical classification. Here, the hierarchy represents the knowledge of the machine about the world and the user, and it is encoded as a DAG where nodes are the different concepts and the edge are the *is-a* relation among them. Under KD, the set of concepts, their individual distributions and the relations change over time. To perform classification in this setting, we detailed TRCKD, an approach built on kNN combining three stages: drift detection, interactive disambiguation stage in which the user informs the machine which type of KD occurred, and adaptation of model and hierarchy. In PCR, the user knows if any KD occurred and which kind. The experiments highlight that the three steps above are necessary to achieve performance improvements. Failing to detect and adapt to drifts implies systematic mis-predictions on future examples. Future investigations will focus on other learning architectures like neural networks and how to handle other types of relations besides *is-a*, e.g., by converting them to DAG [73].

In the second part, how to move PCR from the controlled environment of the lab to the real world. In Chapter 6, we proposed a reference architecture that integrates language and knowledge alignment addressed by skeptical learning and knowledge drift methods, respectively. As described above, PCR is achieved by addressing multiple aspects, namely noisy labels, knowledge drift and interaction with the user. The compositionality and modularity of the architecture allow the implementation of diverse policies to load the necessary data from the memory, to pre-process the data coming from the sensor and to update the knowledge in the memory. This is key in a lifelong scenario where the recognition task evolves, and obsolete or irrelevant data should be pruned over time to keep only what is needed to solve the task at hand.

In Chapter 7, we instantiate the architecture above and ISGP in a concrete use case, namely social science data collection. The goal is to investigate the benefits of PCR in longitudinal studies. The benefits are twofold.

First, it can reduce the respondent burden by letting the machine answer the questions asking about the participants' contexts. Second, it improves the answers quality by asking the user to revise suspicious answers. To this end, we ran a real-world experiment with university students interacting with ISGP.

8.2 Discussion

8.2.1 Benefits and limitations

The ML approaches of this work, i.e., ISGP, TRCKD and CINCER, have been presented with the goal of improving context recognition in the wild. However, they can be applied in different domains characterized by human supervision, incremental learning and knowledge changes. The latter affects almost all tasks, but in some scenarios, the changes are gradual and thus can be neglected. Possible applications are quantified self [108], i.e., self-tracking personal activities and daily life, citizen science and data curation.

The central claim of this work is that detecting and adapting to changes in knowledge and language is crucial to build robust models. Recognizing the correct context is necessary to provide useful services to empower human intelligence with artificial intelligence. This robustness has been obtained by following a *model-centric* paradigm focusing on building new models and losses [120]. This work shifts to *data-centric* solutions [189] that fix mislabeled labels and update training examples labels in case of drifts. Thus, the ML model is used to perform data cleaning and error fixing [40] to improve the performance.

One limitation of the work is that the human evaluation has been performed only on Skeptical Learning. TRCKD and CINCER miss an evaluation with real users. Another future work direction is exploring how to visualize counter-examples selected by CINCER when working on tabular data and sensor data. When working on images, the example is shown directly to the annotator, but in the case of tabular data, each feature might not have immediate semantic meaning.

8.2.2 Ethics statement and societal impact

Following the approach applied by several conferences[1] in the machine learning field, we communicate possible and known societal impacts of this work. The approaches conceived in this work, namely ISGP, TRCKD and CINCER are interactive learning approaches that leverage human feedback to improve the reliability of the predictions. The risk is that the methods annoy the user by asking an excessive number of questions. In the case of ISGP,

[1]E.g., NeurIPS Code of Ethics https://web.archive.org/web/20230904142928/https://neurips.cc/public/EthicsGuidelines

this is mitigated by increasing the probability of contradicting the user only when the model is sufficiently confident in its own predictions (Section 3.4.2). Moreover, ISGP prevents situations in which the model fails to learn because refuses to request labels far from the training sets and continuously querying the user regardless of her past annotation correctness. Regarding TRCKD, the drift detection threshold τ can be fine-tuned to reduce the false positive detection rate and thus the number of disambiguation interactions, at the cost of adapting to the drift with increased delay. The proposed approaches are susceptible to adversarial attacks like providing corrupted supervisions to alter the model and drive its predictions malignant goals. For this reason, the model must be protected by restricting the access to the model by the authorized user only, as done in safety-critical applications.

The methods implement an egocentric view, namely the perspective of the user, to provide reliable and useful services to the user, who acts collaboratively to maximize the benefits of using them. The egocentric view is enabled by training the model on the sensor data of the user's devices and her or his answer to the questions. In addition, the interaction with the user to fix noisy annotations and disambiguate among the drift contributes to keep the machine aligned with the user. These approaches prevent possible bias against people of a certain gender, race, sexuality, or other protected characteristics. Since the models are user-specific and leveraged by the user herself, there are no discrimination issues, such as access to healthcare and education. Possible discrimination may arise from the point of view of the accessibility of the technology and services, issues addressed with careful interface design. On the other hand, egocentric data may be exploited to build a user profile, for surveillance reasons, to predict protected categories or to endanger individual well-being. For these reasons, the data must be stored on secure and privacy-preserving data storage that implements encryption and anonymization to reduce the risk of data leakage or theft. Solutions like Solid[2] enable the discovery and sharing of personal data, protecting the user's privacy. The users store their personal data in one or more decentralized personal online data stores, managed by trusted providers or self-hosted. the user grants access to the applications, which can access specific data by leveraging interoperable data formats and protocols. Hence, the user keeps the ownership and control of the data. In addition to the data, the machine learning models must be protected and thus stored and run on the user's machine rather than executing them on a central server.

The research detailed in Chapter 7 involved human participants interacting with a technical system, and thus, the work went through the approval process of the Research Ethics Committee of the University of Trento. The participants were at least 18 years old and provided consent prior to participating in the experiment. At the end of the data collection, the user iden-

[2]Solid (Social Linked Data) https://solidproject.org/

tifiers are anonymized, and the connectivity sensors (WiFi and Bluetooth) are hashed. The data are minimized by using techniques like summarization and aggregation, ensuring that the data can still be used to fulfil the research purpose. The processed data are then archived in protected repositories.

8.3 Future research development

Each chapter that describes the contributions reports the future work and potential improvement of the presented approaches. Here, we focus on a more general discussion of new research directions. Traditional interactive learning and active learning approaches investigate which examples interact with the annotator to retrieve or fix the label. One interesting direction is to find the best time of the day or context for interacting. This would have to consider both past behaviour and user preferences. For instance, the user may be more willing to answer before going to bed, and the machine should avoid sending questions when driving. In this work, the user was the source of the feedback, given the egocentric requirement of PCR. However, with the goal of reducing the respondent burden, third parties can be involved, such as friends or external services (e.g., using weather forecasting to refine the confidence of the machine on outdoor activity labels by decreasing their probability during rainy days). One limitation of learning from streams is the cold start problem. Indeed, at the beginning, a limited amount of data is available and acquiring enough data requires from days to weeks. Transfer learning can overcome this [170]. The initial model is taken from another person with a profile similar to the reference user. Then, the model is refined and adapted over time by leveraging the technique presented in this work. However, a few open questions arise concerning which part of the model and knowledge hierarchy to transfer and how to define the profile of the user. Further work should focus on how to deal with very unbalanced context labels (e.g., *home* is the most frequent answer to the time diaries in results of Section 7.5.2) and sparse data such as missing labels and disabled sensors. These issues impact the model performance. Regarding the explainable skepticism in Chapter 4, it can be further evaluated on images collected from egocentric cameras (e.g., [77]). In this case, also the smart glasses become a source of data to be used to identify contextual information.

Bibliography

[1] Zahraa S Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. “Activity recognition with evolving data streams: A review”. In: *ACM Computing Surveys (CSUR)* (2018) (cit. on p. 16).

[2] Zahraa S Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. “AnyNovel: Detection of novel concepts in evolving data streams: An application for activity recognition”. In: *Evolving Systems* (2016) (cit. on p. 17).

[3] Naman Agarwal, Brian Bullins, and Elad Hazan. “Second-order stochastic optimization for machine learning in linear time”. In: *The Journal of Machine Learning Research* (2017) (cit. on p. 46).

[4] Dana Angluin and Philip Laird. “Learning from noisy examples”. In: *Machine Learning* (1988) (cit. on pp. 31, 37).

[5] Gregory D. Abowd Anind K. Dey and Daniel Salber. “A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications”. In: *Human–Computer Interaction* (2001). DOI: `10.1207/S15327051HCI16234\_02`. eprint: `https://doi.org/10.1207/S15327051HCI16234_02`. URL: `https://doi.org/10.1207/S15327051HCI16234_02` (cit. on p. 91).

[6] Wageesha Bangamuarachchi, Anju Chamantha, et al. “Sensing Eating Events in Context: A Smartphone-Only Approach”. In: *IEEE Access* (2022) (cit. on p. 17).

[7] Allan de Barcelos Silva, Marcio Miguel Gomes, et al. “Intelligent personal assistants: A systematic literature review”. In: *Expert Systems with Applications* (2020) (cit. on p. 2).

[8] Gianni Barlacchi, Christos Perentis, et al. “Are you getting sick? Predicting influenza-like symptoms using human mobility behaviors”. In: *EPJ data science* (2017) (cit. on p. 17).

[9] Mariam Barry, Albert Bifet, and Jean-Luc Billy. “StreamAI: Dealing with Challenges of Continual Learning Systems for Serving AI in Production”. In: *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE. 2023 (cit. on p. 98).

[10] Mariam Barry, Jacob Montiel, et al. “StreamMLOps: Operationalizing Online Learning for Big Data Streaming & Real-Time Applications”. In: *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE. 2023 (cit. on pp. 98, 99).

[11] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. “RelatIF: Identifying Explanatory Training Examples via Relative Influence”. In: *The 23rd International Conference on Artificial Intelligence and Statistics*. 2020 (cit. on pp. 48, 49, 53).

[12] Samyadeep Basu, Philip Pope, and Soheil Feizi. “Influence functions in deep learning are fragile”. In: *arXiv preprint arXiv:2006.14651* (2020) (cit. on p. 47).

[13] Samyadeep Basu, Xuchen You, and Soheil Feizi. “Second-Order Group Influence Functions for Black-Box Predictions”. In: *arXiv preprint arXiv:1911.00418* (2019) (cit. on p. 55).

[14] Jonathan Baxter. “A model of inductive bias learning”. In: *JAIR* (2000) (cit. on p. 37).

[15] Abhijit Bendale and Terrance Boult. “Towards open world recognition”. In: *CVPR*. 2015 (cit. on p. 37).

[16] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. “Importance weighted active learning”. In: *ICML*. 2009 (cit. on p. 28).

[17] G. Biegel and V. Cahill. “A framework for developing mobile, context-aware applications”. In: *Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Proceedings of the*. 2004. DOI: `10.1109/PERCOM.2004.1276875` (cit. on p. 92).

[18] Albert Bifet, Gianmarco de Francisci Morales, et al. “Efficient online evaluation of big data stream classifiers”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015 (cit. on p. 100).

[19] Ivano Bison and Haonan Zhao. *Factors Impacting the Quality of User Answers on Smartphones*. 2023. arXiv: `2306.00627 [cs.HC]` (cit. on p. 112).

[20] Andrea Bontempelli, Marcelo Rodas Britez, et al. “Lifelong Personal Context Recognition”. In: *First international workshop on Human-Centered Design of Symbiotic Hybrid Intelligence co-located with The first International Conference on Hybrid Human-Artificial Intelligence (HHAI)* (2022) (cit. on p. 2).

[21] Andrea Bontempelli, Fausto Giunchiglia, Andrea Passerini, and Stefano Teso. “Human-in-the-loop handling of knowledge drift”. In: *Data Mining and Knowledge Discovery* (2022) (cit. on p. 57).

[22] Andrea Bontempelli, Stefano Teso, Fausto Giunchiglia, and Andrea Passerini. “Learning in the Wild with Incremental Skeptical Gaussian Processes”. In: *IJCAI*. 2020 (cit. on pp. 20, 41, 44, 51, 73).

[23] Terrance E Boult et al. "Learning and the Unknown: Surveying Steps toward Open World Recognition". In: *AAAI*. 2019 (cit. on pp. 15, 23, 36, 37, 73).

[24] Paolo Bouquet and Fausto Giunchiglia. "Reasoning about theory adequacy. a new solution to the qualification problem". In: *Fundamenta Informaticae* (1995) (cit. on p. 2).

[25] Matteo Busso. "The iLog methodology for fostering valid and reliable Big Thick Data". PhD thesis. University of Trento, 2024 (cit. on p. 96).

[26] Luca Canzian and Mirco Musolesi. "Trajectories of depression: unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis". In: *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*. 2015 (cit. on p. 139).

[27] Yinzhi Cao and Junfeng Yang. "Towards making systems forget with machine unlearning". In: *2015 IEEE Symposium on Security and Privacy*. 2015 (cit. on pp. 64, 75).

[28] Matthias Carnein, Heike Trautmann, Albert Bifet, and Bernhard Pfahringer. "confstream: Automated algorithm selection and configuration of stream clustering algorithms". In: *Learning and Intelligent Optimization: 14th International Conference, LION 14, Athens, Greece, May 24–28, 2020, Revised Selected Papers 14*. Springer. 2020 (cit. on p. 112).

[29] Scott Carter, Jennifer Mankoff, and Jeffrey Heer. "Momento: support for situated ubicomp experimentation". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2007 (cit. on p. 92).

[30] Bilge Celik, Prabhant Singh, and Joaquin Vanschoren. "Online automl: An adaptive automl framework for online learning". In: *Machine Learning* (2023) (cit. on p. 112).

[31] Nicolo Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. "Worst-case analysis of selective sampling for linear classification". In: *JMLR* (2006) (cit. on pp. 28, 29).

[32] Jiaoyan Chen, Freddy Lécué, Jeff Pan, and Huajun Chen. "Learning from ontology streams with semantic concept drift". In: *Twenty-Sixth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization. 2017 (cit. on p. 73).

[33] Kaixuan Chen, Dalin Zhang, et al. "Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities". In: *ACM Computing Surveys (CSUR)* (2021) (cit. on p. 2).

[34] Jonas Chromik, Kristina Kirsten, et al. "SensorHub: Multimodal sensing in real-life enables home-based studies". In: *Sensors* (2022) (cit. on p. 92).

[35] Gabriele Civitarese, Riccardo Presotto, and Claudio Bettini. "Context-driven active and incremental activity recognition". In: *arXiv preprint arXiv:1906.03033* (2019) (cit. on pp. 89, 92).

[36] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. "EMNIST: Extending MNIST to handwritten letters". In: *IJCNN*. 2017 (cit. on p. 65).

[37] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982 (cit. on pp. 41, 45).

[38] Piergiorgio Corbetta. *Social research: Theory, methods and techniques*. Sage, 2003 (cit. on p. 96).

[39] Louise Corti. "Using diaries in social research". In: (1993) (cit. on pp. 41, 96).

[40] Pierre-Olivier Côté, Amin Nikanjam, et al. "Data Cleaning and Machine Learning: A Systematic Literature Review". In: *arXiv preprint arXiv:2310.01765* (2023) (cit. on p. 115).

[41] Joëlle Coutaz, James L Crowley, Simon Dobson, and David Garlan. "Context is key". In: *Communications of the ACM* (2005) (cit. on pp. 3, 9, 12, 92).

[42] Rocco De Rosa, Thomas Mensink, and Barbara Caputo. "Online open world recognition". In: *arXiv preprint arXiv:1604.02275* (2016) (cit. on p. 37).

[43] Florenc Demrozi, Graziano Pravadelli, Azra Bihorac, and Parisa Rashidi. "Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey". In: *IEEE access* (2020) (cit. on p. 16).

[44] Jaka Demšar and Zoran Bosnić. "Detecting concept drift in data streams using model explanation". In: *Expert Systems with Applications* (2018) (cit. on pp. 73, 75).

[45] G Denevi, C Ciliberto, D Stamos, and M Pontil. "Incremental learning-to-learn with statistical guarantees". In: *UAI*. 2018 (cit. on p. 37).

[46] Thomas Dietterich. "Steps toward robust artificial intelligence". In: *AI Magazine* (2017) (cit. on pp. 3, 15, 21).

[47] Kaize Ding, Jundong Li, and Huan Liu. "Interactive anomaly detection on attributed networks". In: *Proceedings of the twelfth ACM international conference on web search and data mining*. 2019 (cit. on p. 73).

[48] Pinar Donmez and Jaime G Carbonell. “Proactive learning: cost-sensitive active learning with multiple imperfect oracles”. In: *Proceedings of the 17th ACM conference on Information and knowledge management*. 2008 (cit. on p. 37).

[49] Nicola Dragoni, Saverio Giallorenzo, et al. “Microservices: yesterday, today, and tomorrow”. In: *Present and ulterior software engineering* (2017) (cit. on p. 98).

[50] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml` (cit. on p. 50).

[51] Bruce Edmonds. “Context in social simulation: why it can’t be wished away”. In: *Computational and Mathematical Organization Theory* (2012) (cit. on p. 9).

[52] Luca Erculiani, Andrea Bontempelli, Andrea Passerini, and Fausto Giunchiglia. “Egocentric Hierarchical Visual Semantics”. In: *Proceedings of the Second International Conference on Hybrid Human-Machine Intelligence (HHAI 23)*. 2023 (cit. on p. 13).

[53] Luca Erculiani, Fausto Giunchiglia, and Andrea Passerini. “Continual Egocentric Object Recognition”. In: *ECAI 2020*. IOS Press, 2020 (cit. on p. 13).

[54] Miikka Ermes, Juha Pärkkä, Jani Mäntyjärvi, and Ilkka Korhonen. “Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions”. In: *IEEE transactions on information technology in biomedicine* (2008) (cit. on p. 16).

[55] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *KDD*. 1996 (cit. on p. 35).

[56] eurostat. *Harmonised European Time Use Surveys (HETUS)*. 2018. URL: `https://ec.europa.eu/eurostat/web/products-manuals-and-guidelines/-/ks-gq-20-011` (cit. on p. 103).

[57] Anna Ferrari, Daniela Micucci, Marco Mobilio, and Paolo Napoletano. “Deep learning and model personalization in sensor-based human activity recognition”. In: *Journal of Reliable Intelligent Environments* (2023) (cit. on p. 17).

[58] Denzil Ferreira, Vassilis Kostakos, and Anind K. Dey. “AWARE: Mobile Context Instrumentation Framework”. In: *Frontiers in ICT* (2015). DOI: `10.3389/fict.2015.00006`. URL: `https://www.frontiersin.org/articles/10.3389/fict.2015.00006` (cit. on p. 92).

[59] Timo Flesch, Jan Balaguer, et al. “Comparing continual task learning in minds and machines”. In: *Proceedings of the National Academy of Sciences* (2018) (cit. on p. 74).

[60] Benoît Frénay and Michel Verleysen. “Classification in the presence of label noise: a survey”. In: *IEEE Trans. Neural Netw. Learn. Syst* (2014) (cit. on pp. 23, 37, 41).

[61] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. “Sparse inverse covariance estimation with the graphical lasso”. In: *Biostatistics* (2008) (cit. on p. 71).

[62] Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. “Issues in evaluation of stream learning algorithms”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009 (cit. on p. 100).

[63] João Gama, Indrė Žliobaitė, et al. “A survey on concept drift adaptation”. In: *ACM Comput Surv* (2014) (cit. on pp. 3, 57, 59–61, 72).

[64] Nan Gao, Mohammad Saiedur Rahaman, Wei Shao, and Flora D Salim. “Investigating the reliability of self-report data in the wild: The quest for ground truth”. In: *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*. 2021 (cit. on p. 3).

[65] Xin Geng and Kate Smith-Miles. “Incremental Learning”. In: *Encyclopedia of Biometrics*. Boston, MA: Springer US, 2009. ISBN: 978-0-387-73003-5. DOI: `10.1007/978-0-387-73003-5_304`. URL: `https://doi.org/10.1007/978-0-387-73003-5_304` (cit. on p. 16).

[66] Amirata Ghorbani and James Zou. “Data shapley: Equitable valuation of data for machine learning”. In: *International Conference on Machine Learning*. PMLR. 2019 (cit. on p. 55).

[67] Aritra Ghosh, Naresh Manwani, and PS Sastry. “On the robustness of decision tree learning under label noise”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2017 (cit. on p. 37).

[68] Christophe Giraud-Carrier. “A note on the utility of incremental learning”. In: *Ai Communications* (2000) (cit. on p. 16).

[69] Fausto Giunchiglia. “Contextual reasoning”. In: *Epistemologia, special issue on ’I Linguaggi e le Macchine’* (1993) (cit. on pp. 2, 9).

[70] Fausto Giunchiglia, Khuyagbaatar Batsuren, and Gabor Bella. “Understanding and Exploiting Language Diversity”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*. 2017 (cit. on pp. 4, 81).

[71] Fausto Giunchiglia, Enrico Bignotti, and Mattia Zeni. “Personal context modelling and annotation”. In: *PerCom*. 2017 (cit. on pp. 10, 20, 57).

[72] Fausto Giunchiglia, Ivano Bison, et al. "A worldwide diversity pilot on daily routines and social practices (2020)". In: (2021) (cit. on p. 101).

[73] Fausto Giunchiglia, Marcelo Rodas Britez, Andrea Bontempelli, and Xiaoyue Li. "Streaming and learning the personal context". In: *arXiv preprint arXiv:2108.08234* (2021) (cit. on p. 114).

[74] Fausto Giunchiglia and Mattia Fumagalli. "Teleologies: Objects, actions and functions". In: *International conference on conceptual modeling*. Springer. 2017 (cit. on p. 10).

[75] Fausto Giunchiglia, Alessio Zamboni, Mayukh Bagchi, and Simone Bocca. "Stratified data integration". In: *arXiv preprint arXiv:2105.09432* (2021) (cit. on pp. 10, 81).

[76] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. "Understanding individual human mobility patterns". In: *nature* (2008) (cit. on p. 139).

[77] Kristen Grauman, Andrew Westbury, et al. "Ego4d: Around the world in 3,000 hours of egocentric video". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022 (cit. on p. 117).

[78] Arthur Gretton, Karsten M Borgwardt, et al. "A kernel two-sample test". In: *JMLR* (2012) (cit. on pp. 57, 62).

[79] Maciej Grzenda, Heitor Murilo Gomes, and Albert Bifet. "Delayed labelling evaluation for data streams". In: *Data Mining and Knowledge Discovery* (2020) (cit. on p. 100).

[80] Christophe Guerét. "Knowledge Graphs in support of Human-Machine intelligence". In: *The first International Conference on Hybrid Human-Artificial Intelligence (HHAI2022)* (2022). URL: https://www.hhai-conference.org/wp-content/uploads/2022/06/hhai-2022_paper_67.pdf (cit. on p. 78).

[81] Ramanathan Guha, Vineet Gupta, Vivek Raghunathan, and Ramakrishnan Srikant. "User modeling for a personal assistant". In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. 2015 (cit. on p. 2).

[82] Riccardo Guidotti, Anna Monreale, et al. "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* (Aug. 2018) (cit. on p. 4).

[83] Han Guo, Nazneen Fatema Rajani, et al. "FastIF: Scalable Influence Functions for Efficient Model Interpretation and Debugging". In: *arXiv preprint arXiv:2012.15781* (2020) (cit. on p. 49).

[84] Shengbo Guo, Scott Sanner, and Edwin V Bonilla. "Gaussian process preference elicitation". In: *NeurIPS*. 2010 (cit. on p. 25).

[85] James Hensman, Nicolò Fusi, and Neil D Lawrence. “Gaussian processes for Big data”. In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. 2013 (cit. on p. 30).

[86] Martin Heusel, Hubert Ramsauer, et al. “GANs trained by a two time-scale update rule converge to a local nash equilibrium”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017 (cit. on p. 49).

[87] Tommi S Jaakkola, David Haussler, et al. “Exploiting generative models in discriminative classifiers”. In: *Advances in neural information processing systems* (1999) (cit. on pp. 52, 54).

[88] Jeya Vikranth Jeyakumar, Joseph Noor, et al. “How Can I Explain This to You? An Empirical Study of Deep Neural Network Explanation Methods”. In: *Advances in Neural Information Processing Systems* (2020) (cit. on p. 54).

[89] Wittawat Jitkrittum, Zoltán Szabó, Kacper P Chwialkowski, and Arthur Gretton. “Interpretable distribution features with maximum testing power”. In: *NeurIPS*. 2016 (cit. on pp. 62, 71, 72).

[90] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. “Active learning with gaussian processes for object categorization”. In: *ICCM*. 2007 (cit. on pp. 25, 26).

[91] Rajiv Khanna, Been Kim, Joydeep Ghosh, and Sanmi Koyejo. “Interpreting Black Box Predictions using Fisher Kernels”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2019 (cit. on pp. 53, 54).

[92] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. “Detecting change in data streams”. In: *VLDB*. 2004 (cit. on pp. 61, 62, 72, 73).

[93] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. “Examples are not enough, learn to criticize! criticism for interpretability”. In: *Advances in neural information processing systems* (2016) (cit. on p. 73).

[94] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 51).

[95] Diederik P Kingma and Max Welling. “Auto-encoding Variational Bayes”. In: *ICLR’14*. 2014 (cit. on p. 65).

[96] Pang Wei Koh and Percy Liang. “Understanding black-box predictions via influence functions”. In: *Proceedings of the 34th International Conference on Machine Learning*. JMLR. org. 2017 (cit. on pp. 41, 45–47, 52–54).

[97] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. “On the accuracy of influence functions for measuring group effects”. In: *Advances in Neural Information Processing Systems*. 2019 (cit. on p. 55).

[98] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. 2009 (cit. on p. 70).

[99] Jan Kremer, Fei Sha, and Christian Igel. “Robust active label correction”. In: *International conference on artificial intelligence and statistics*. PMLR. 2018 (cit. on pp. 41, 43).

[100] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. “Machine learning operations (mlops): Overview, definition, and architecture”. In: *IEEE Access* (2023) (cit. on p. 97).

[101] F Kunstner, L Balles, and P Hennig. “Limitations of the Empirical Fisher Approximation for Natural Gradient Descent”. In: *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. Curran Associates, Inc. 2020 (cit. on pp. 48, 54).

[102] Peter Kuppens. “The open handbook of experience sampling methodology: A step-by-step guide to designing, conducting, and analyzing ESM studies”. In: (2021) (cit. on p. 96).

[103] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* (2010) (cit. on p. 50).

[104] Erich L Lehmann and George Casella. *Theory of point estimation*. Springer Science & Business Media, 2006 (cit. on p. 41).

[105] Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. “Human-in-the-loop Debugging Deep Text Classifiers”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020 (cit. on p. 54).

[106] James R Lloyd and Zoubin Ghahramani. “Statistical model criticism using kernel two sample tests”. In: *Advances in Neural Information Processing Systems* (2015) (cit. on pp. 62, 63).

[107] Jie Lu, Anjin Liu, et al. “Learning under concept drift: A review”. In: *IEEE T. Knowl. Data En.* (2018) (cit. on p. 72).

[108] Deborah Lupton. *The quantified self*. John Wiley & Sons, 2016 (cit. on p. 115).

[109] Alexander Lütz, Erik Rodner, and Joachim Denzler. “I Want To Know More – Efficient Multi-Class Incremental Learning Using Gaussian Processes”. In: *Pattern Recognition and Image Analysis* (2013) (cit. on pp. 21, 26, 27, 38).

[110] Luigi Malago, Nicolo Cesa-Bianchi, and J Renders. “Online active learning with strong and weak annotators”. In: *NIPS Workshop on Learning from the Wisdom of Crowds*. 2014 (cit. on p. 55).

[111] James Martens and Roger Grosse. “Optimizing neural networks with kronecker-factored approximate curvature”. In: *International conference on machine learning*. PMLR. 2015 (cit. on pp. 48, 54).

[112] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (cit. on p. 50).

[113] Mohammad Masud, Jing Gao, et al. “Classification and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints”. In: *IEEE Transactions on Knowledge and Data Engineering* (2011) (cit. on p. 74).

[114] Rob McCarney, James Warner, et al. “The Hawthorne Effect: a randomised, controlled trial”. In: *BMC medical research methodology* (2007) (cit. on p. 96).

[115] Merijn Mestdagh, Stijn Verdonck, et al. “m-Path: An easy-to-use and flexible platform for ecological momentary assessment and intervention in behavioral research and clinical practice”. In: (2022) (cit. on p. 92).

[116] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial Intelligence* (Aug. 2018). ISSN: 0004-3702. DOI: 10.1016/j.artint.2018.07.007 (cit. on pp. 4, 54).

[117] Tom M Mitchell, Rich Caruana, et al. “Experience with a learning personal assistant”. In: *Communications of the ACM* (1994) (cit. on p. 2).

[118] Karen Myers, Pauline Berry, et al. “An intelligent personal assistant for task and time management”. In: *AI Magazine* (2007) (cit. on p. 2).

[119] Nitika Nigam, Tanima Dutta, and Hari Prabhat Gupta. “Impact of noisy labels in learning techniques: a survey”. In: *Advances in data and information sciences*. Springer, 2020 (cit. on p. 37).

[120] Curtis Northcutt, Lu Jiang, and Isaac Chuang. “Confident learning: Estimating uncertainty in dataset labels”. In: *Journal of Artificial Intelligence Research* (2021) (cit. on pp. 37, 115).

[121] Luca Pappalardo, Salvatore Rinzivillo, et al. “Understanding the patterns of car travel”. In: *The European Physical Journal Special Topics* (2013) (cit. on p. 139).

[122] German I Parisi, Ronald Kemker, et al. “Continual lifelong learning with neural networks: A review”. In: *Neural Networks* (2019) (cit. on p. 74).

[123] Ramesh Paudel and William Eberle. “An Approach For Concept Drift Detection in a Graph Stream Using Discriminative Subgraphs”. In: *TKDD* (2020) (cit. on p. 73).

[124] Fernando Pérez-Cruz. “Estimation of information theoretic measures for continuous random variables”. In: *NeurIPS*. 2009 (cit. on p. 62).

[125] Gianluigi Pillonetto, Francesco Dinuzzo, and Giuseppe De Nicolao. “Bayesian online multitask learning of Gaussian processes”. In: *IEEE Trans. Pattern Anal. Mach. Intell* (2008) (cit. on p. 37).

[126] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. “A review of novelty detection”. In: *Signal Processing* (2014) (cit. on p. 73).

[127] Teodora Popordanoska, Mohit Kumar, and Stefano Teso. “Machine guides, human supervises: Interactive learning with global explanations”. In: *arXiv preprint arXiv:2009.09723* (2020) (cit. on p. 54).

[128] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. “A unifying view of sparse approximate Gaussian process regression”. In: *JMLR* (2005) (cit. on p. 30).

[129] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. “icarl: Incremental classifier and representation learning”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017 (cit. on p. 74).

[130] Ievgen Redko, Emilie Morvant, et al. “A survey on domain adaptation theory: learning bounds and theoretical guarantees”. In: *arXiv preprint arXiv:2004.11829* (2020) (cit. on p. 72).

[131] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *EMNLP-IJCNLP*. 2019 (cit. on pp. 50, 65).

[132] Daniele Riboni and Claudio Bettini. “COSAR: hybrid reasoning for context-aware activity recognition”. In: *Personal and Ubiquitous Computing* (2011) (cit. on p. 92).

[133] Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. “Gaussian process classification and active learning with multiple annotators”. In: *ICML*. 2014 (cit. on p. 25).

[134] Yannick Roos, Michael D Krämer, et al. “Does your smartphone “know” your social life? A methodological comparison of day reconstruction, experience sampling, and mobile sensing”. In: *Advances in Methods and Practices in Psychological Science* (2023) (cit. on p. 16).

[135] Martha Roseberry, Bartosz Krawczyk, and Alberto Cano. “Multi-label punitive kNN with self-adjusting memory for drifting data streams”. In: *TKDD* (2019) (cit. on pp. 61, 64, 67).

[136] Ethan M Rudd, Lalit P Jain, Walter J Scheirer, and Terrance E Boult. "The extreme value machine". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2017) (cit. on pp. 37, 73).

[137] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* (2019) (cit. on p. 49).

[138] Walter J Scheirer, Lalit P Jain, and Terrance E Boult. "Probability models for open set recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2014) (cit. on pp. 15, 37, 73).

[139] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. "Toward open set recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2013) (cit. on pp. 37, 73).

[140] Jeffrey C Schlimmer and Richard H Granger. "Incremental learning from noisy data". In: *Machine learning* (1986) (cit. on p. 65).

[141] Patrick Schramowski et al. "Making deep neural networks right for the right scientific reasons by interacting with their explanations". In: *Nat Mach Intell* (2020) (cit. on pp. 54, 75).

[142] David Sculley, Gary Holt, et al. "Hidden technical debt in machine learning systems". In: *Advances in neural information processing systems* (2015) (cit. on p. 98).

[143] Ramprasaath R Selvaraju, Stefan Lee, et al. "Taking a hint: Leveraging explanations to make vision and language models more grounded". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019 (cit. on p. 54).

[144] Grigorios Skolidis and Guido Sanguinetti. "Bayesian multitask classification with Gaussian process priors". In: *IEEE Trans. Neural Netw.* (2011) (cit. on p. 37).

[145] Arnold WM Smeulders, Marcel Worring, et al. "Content-based image retrieval at the end of the early years". In: *IEEE Transactions on pattern analysis and machine intelligence* (2000) (cit. on p. 13).

[146] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning". In: *NeurIPS*. 2017 (cit. on pp. 74, 75).

[147] Youngdoo Son and Seokho Kang. "Regression with re-labeling for noisy data". In: *Expert Systems with Applications* (2018) (cit. on p. 37).

[148] Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. "Learning from noisy labels with deep neural networks: A survey". In: *arXiv preprint arXiv:2007.08199* (2020) (cit. on pp. 37, 41).

[149] Mohammad S Sorower. “A literature survey on algorithms for multi-label learning”. In: *Oregon State University, Corvallis* (2010) (cit. on p. 66).

[150] Eduardo J Spinosa, André Ponce de Leon F. de Carvalho, and João Gama. “Olindda: A cluster-based approach for detecting novelty and concept drift in data streams”. In: *Proceedings of the 2007 ACM symposium on Applied computing*. 2007 (cit. on p. 74).

[151] Spyromitros-Xioufis et al. “Dealing with concept drift and class imbalance in multi-label stream classification”. In: *IJCAI*. 2011 (cit. on pp. 57, 61, 62, 64, 67, 72).

[152] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. “Information-theoretic regret bounds for gaussian process optimization in the bandit setting”. In: *IEEE Transactions on Information Theory* (2012) (cit. on p. 25).

[153] Sebastian Staacks, Simon Hütz, Heidrun Heinke, and Christoph Stampfer. “Advanced tools for smartphone-based experiments: phyphox”. In: *Physics education* (2018) (cit. on p. 92).

[154] Ljiljana Stojanovic, Alexander Maedche, Boris Motik, and Nenad Stojanovic. “User-driven ontology evolution management”. In: *ECAW*. 2002 (cit. on p. 57).

[155] Marija Stojchevska, Mathias De Brouwer, et al. “From Lab to Real World: Assessing the Effectiveness of Human Activity Recognition and Optimization through Personalization”. In: *Sensors* (2023) (cit. on p. 17).

[156] Thomas Strang and Claudia Linnhoff-Popien. “A context modeling survey”. In: *Workshop Proceedings*. 2004 (cit. on p. 9).

[157] William R Swartout, Benjamin D Nye, et al. “Designing a personal assistant for life-long learning (PAL3)”. In: *The Twenty-Ninth International Flairs Conference*. 2016 (cit. on p. 2).

[158] Zoltán Szabó and Bharath K Sriperumbudur. “Characteristic and universal tensor product kernels”. In: *JMLR* (2017) (cit. on p. 62).

[159] Stefano Teso, Andrea Bontempelli, Fausto Giunchiglia, and Andrea Passerini. “Interactive Label Cleaning with Example-based Explanations”. In: *arXiv preprint arXiv:2106.03922* (2021) (cit. on p. 40).

[160] Stefano Teso and Kristian Kersting. “Explanatory interactive machine learning”. In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 2019 (cit. on pp. 49, 54).

[161] Stefano Teso and Antonio Vergari. “Efficient and Reliable Probabilistic Interactive Learning with Structured Outputs”. In: *arXiv preprint arXiv:2202.08566* (2022) (cit. on p. 84).

[162] Sebastian Thrun. "Is learning the n-th thing any easier than learning the first?" In: *NeurIPS*. 1996 (cit. on p. 37).

[163] Daniel Ting and Eric Brochu. "Optimal subsampling with influence functions". In: *Advances in Neural Information Processing Systems*. 2018 (cit. on p. 48).

[164] Simon Tong and Daphne Koller. "Active learning for structure in Bayesian networks". In: *International joint conference on artificial intelligence*. Citeseer. 2001 (cit. on p. 74).

[165] Roger Tourangeau, Lance J Rips, and Kenneth Rasinski. *The psychology of survey response*. 2000 (cit. on pp. 14, 21, 41, 96).

[166] Alexey Tsymbal. "The problem of concept drift: definitions and related work". In: *Computer Science Department, Trinity College Dublin* (2004) (cit. on p. 3).

[167] Ruth Urner, Shai Ben-David, and Ohad Shamir. "Learning from weak teachers". In: *Artificial intelligence and statistics*. PMLR. 2012 (cit. on pp. 41, 43).

[168] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. "Recognizing detailed human context in the wild from smartphones and smartwatches". In: *IEEE pervasive computing* (2017) (cit. on pp. 1, 17).

[169] Yonatan Vaizman, Katherine Ellis, Gert Lanckriet, and Nadir Weibel. "ExtraSensory App: Data Collection In-the-Wild with Rich User Interface to Self-Report Behavior". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. New York, NY, USA: Association for Computing Machinery, 2018. ISBN: 9781450356206. DOI: 10.1145/3173574.3174128. URL: https://doi.org/10.1145/3173574.3174128 (cit. on p. 92).

[170] Yonatan Vaizman, Nadir Weibel, and Gert Lanckriet. "Context recognition in-the-wild: Unified model for multi-modal sensors and multi-label classification". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2018) (cit. on pp. 16, 117).

[171] Niels Van Berkel, Denzil Ferreira, and Vassilis Kostakos. "The experience sampling method on mobile devices". In: *ACM Computing Surveys (CSUR)* (2017) (cit. on p. 96).

[172] Bruno Veloso, João Gama, Benedita Malheiro, and João Vinagre. "Hyperparameter self-tuning for data streams". In: *Information Fusion* (2021) (cit. on p. 112).

[173] Rui Wang, Gabriella Harari, et al. "SmartGPA: how smartphones can assess and predict academic performance of college students". In: *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*. 2015 (cit. on p. 16).

[174] Brady T West and Jennifer Sinibaldi. “The quality of paradata: A literature review”. In: *Improving surveys with paradata* (2013) (cit. on pp. 21, 41).

[175] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. 2006 (cit. on pp. 21, 25).

[176] Mike Wojnowicz, Ben Cruz, et al. ““Influence sketching”: Finding influential samples in large-scale regressions”. In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016 (cit. on p. 53).

[177] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: `cs.LG/1708.07747 [cs.LG]` (cit. on p. 50).

[178] Xuhai Xu, Xin Liu, et al. “GLOBEM: Cross-Dataset Generalization of Longitudinal Human Behavior Modeling”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* (2023) (cit. on p. 16).

[179] Yibo Yao and Lawrence B Holder. “Detecting concept drift in classification over streaming graphs”. In: *Proc. KDD Workshop on Mining and Learning with Graphs*. 2016 (cit. on p. 73).

[180] Timothy Ye. *Example Based Explanation in Machine Learning*. `https://github.com/Timothy-Ye/example-based-explanation`. 2020 (cit. on p. 50).

[181] Chih-Kuan Yeh, Joon Sik Kim, Ian EH Yen, and Pradeep Ravikumar. “Representer point selection for explaining deep neural networks”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018 (cit. on p. 47).

[182] Lu Yu, Bartlomiej Twardowski, et al. “Semantic drift compensation for class-incremental learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020 (cit. on p. 74).

[183] Yang Yue, Tian Lan, Anthony GO Yeh, and Qing-Quan Li. “Zooming into individuals to understand the collective: A review of trajectory-based travel behaviour studies”. In: *Travel Behaviour and Society* (2014) (cit. on p. 139).

[184] Özgür Yürür, Chi Harold Liu, et al. “Context-awareness for mobile sensing: A survey and future directions”. In: *IEEE Communications Surveys & Tutorials* (2014) (cit. on p. 92).

[185] D. Zambon, C. Alippi, and L. Livi. “Concept Drift and Anomaly Detection in Graph Streams”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2018) (cit. on p. 73).

[186] Mattia Zeni. "Bridging sensor data streams and human knowledge". PhD thesis. Università degli studi di Trento, 2017 (cit. on pp. 91, 98).

[187] Mattia Zeni, Ilya Zaihrayeu, and Fausto Giunchiglia. "Multi-device activity logging". In: *UbiComp*. 2014 (cit. on pp. 35, 98).

[188] Mattia Zeni, Wanyi Zhang, et al. "Fixing Mislabeling by Human Annotators Leveraging Conflict Resolution and Prior Knowledge". In: *IMWUT* (2019) (cit. on pp. 3, 5, 14, 21, 23, 24, 29–31, 34–36, 38, 41, 43, 55).

[189] Daochen Zha, Zaid Pervaiz Bhat, et al. "Data-centric artificial intelligence: A survey". In: *arXiv preprint arXiv:2303.10158* (2023) (cit. on p. 115).

[190] Jing Zhang, Xindong Wu, and Victor S Sheng. "Learning from crowdsourced labeled data: a survey". In: *Artificial Intelligence Review* (2016) (cit. on pp. 37, 41).

[191] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. "Domain adaptation under target and conditional shift". In: *ICML*. 2013 (cit. on pp. 62, 72).

[192] Min-Ling Zhang and Kun Zhang. "Multi-label learning by exploiting label dependency". In: *ACM SIGKDD*. 2010 (cit. on p. 65).

[193] Wanyi Zhang, Andrea Passerini, and Fausto Giunchiglia. "Dealing with mislabeling via interactive machine learning". In: *KI-Künstliche Intelligenz* (2020) (cit. on p. 14).

[194] Wanyi Zhang, Qiang Shen, et al. "Putting human behavior predictability in context". In: *EPJ Data Science* (2021) (cit. on p. 2).

[195] Wanyi Zhang, Mattia Zeni, Andrea Passerini, and Fausto Giunchiglia. "Skeptical Learning—An Algorithm and a Platform for Dealing with Mislabeling in Personal Context Recognition". In: *Algorithms* (2022) (cit. on p. 91).

[196] Haonan Zhao and Fausto Giunchiglia. "Scheduling Real-Time Acquisition of Context Information". In: (2023) (cit. on p. 112).

[197] Xiulin Zheng, Peipei Li, Zhe Chu, and Xuegang Hu. "A survey on multi-label data stream classification". In: *IEEE Access* (2019) (cit. on pp. 59, 72).

[198] Zhi-Hua Zhou and Zhao-Qian Chen. "Hybrid decision tree". In: *Knowledge-based systems* (2002) (cit. on p. 16).

Appendices

A
iLog sensor list

Table A.1: List of iLog sensors collected during the four weeks of experiment explained in Chapter 7.

Sensor	**Description**
CONNECTIVITY	
Bluetooth normal, Bluetooth low energy	Returns the discovered Bluetooth normal or low energy devices by providing: • *name*: user-friendly name of the remote device; • *address*: hardware MAC Address of the device; • *bondstate*: whether the remote device is connected; • *rssi*: Received Signal Strength Indicator; • *class code* and*class tag*: Bluetooth class of the device (e.g., phone or computer), and the class describes the characteristics and capabilities of the device (e.g., audio and telephony).

WiFi Event	Returns information related to the WIFI network to which the phone is connected; if connected, it also reports the WIFI network ID. Additional features are: • *bssid*: (Basic Service Set Identifier): Special SSID used to define a wireless computer network configured to communicate directly with each other without an access point; • *isconnected*: return if the phone is connected to the WIFI; • *ssid*: (Service Set Identifier) ID or unique identifier of a digital network (Wi-Fi or WLAN).
WiFi Networks Event	Returns all WIFI networks detected by the smartphone. Additional features are: • *address*: is a unique identifier assigned to a network interface controller (NIC) for use as a network address in communications within a network segment; • *capabilities*: allows local area networks (LANs) to operate without cables and wiring; • *frequency*: the WiFi frequency band, that includes two frequency ranges within the wireless spectrum that are designated to carry WIFI: 2.4 GHz and 5 GHz; • *name*: the name assigned to the WiFi network • *rssi*: (Received Signal Strength Indicator) is an estimated measurement of how well a device can hear, detect, and receive signals from any wireless access point or Wi-Fi router. An RSSI closer to 0 is stronger, and closer to –100 is weaker.
Activity	
Accelerometer	Returns the acceleration of the device along the three coordinate axes.

Activities	Return the user's activity recognized by the Google Activity Recognition API. The recognized activities are *in vehicle*, *on bicycle*, *on foot*, *running*, *still*, *tilting*, *walking* and *unknown*. The sensor reports a confidence score between 0 and 100, which represents the likelihood that the user is performing the activity.
Step detector	An event is triggered each time the user takes a step.
Orientation	Returns the position of the device relative to the earth's magnetic north pole.
Location	
Location event	GPS coordinates (latitude, longitude and altitude)
Magnetic field	Reports the ambient magnetic field along the three sensor axes.
Proximity Event	Measures the distance between the user's head and the phone. Depending on the phone, it may be measured in centimetres (i.e., the absolute distance) or as labels (e.g., 'near', 'far')
Software	
Battery Charge Event	Returns whether the phone is on charge and the type of charger
Battery Monitoring Log	Returns the phone's battery level

Table A.2: List of features generated by aggregating raw sensor data in windows of 30 minutes.

Feature name	Type	Description
TIME		
`time_is_workday`	boolean	True for the days from Monday to Friday
`time_is_morning`	boolean	True for the hours between 6 am and 9 am
`time_is_noon`	boolean	True for the hours between 10 am and 1 pm
`time_is_afternoon`	boolean	True for the hours between 2 pm and 5 pm
`time_is_evening`	boolean	True for the hours between 6 pm and 9 pm
`time_is_night`	boolean	True for the hours between 10 pm and 5 am
`time_sin_hour`, `time_cos_hour`	float	Sine and cosine transformations of the hour to encode a stronger connection between two nearby hours[1]
CONNECTIVITY		
`bluetoothdevices_rssi_{mean,var}`	float	mean and variance of the Received Signal Strength Indicator (RSSI) of the detected Bluetooth devices
`bluetoothdevices_nunique`	integer	number of unique Bluetooth normal and low energy devices
`wifi_connection_count`	integer	number of times the device connected to a WiFi network
`wifi_is_connected`	boolean	True if the devices connected to a WiFi network at least once
`wifinetworks_nunique`	integer	number of unique networks detected
ACTIVITY		
`step_detection_count`	integer	number of step detection events
`activity_{invehicle,onbycicle,onfoot,running,still,unknown,walking}`	boolean	True if the Google activity recognition API has recognized the activity
`accelerometer_avg_{x,y,z}`	float	mean of all accelerometer values for each axes separately

[1] Techinal blog on cyclical feature encoding https://developer.nvidia.com/blog/three-approaches-to-encoding-time-information-as-features-for-ml-models/

accelerometer_magnitude_{avg,var}	float	mean and variance of the magnitude of each sensor reading
orientation_avg{x,y,z}	float	mean of all orientation values for each axes separately
orientation_magnitude_{avg,var}	float	mean and variance of the magnitude of each sensor reading
Location		
location_{altitude,longitude,latitude}	float	Averaged GPS coordinates
location_direct_distance	float	Distance between the first and last location point
location_total_distance	float	total distance covered [26]
location_radius_of_gyration	float	Deviation from the centroid of the GPS points [121, 76, 183]
magneticfield_avg{x,y,z}	float	mean of all magneticfield values for each axes separately
magneticfield_magnitude_{avg,var}	float	mean and variance of the magnitude of each sensor reading
proximity_{mean,var}	float	mean and variance of the proximity values
Software		
battery_deltashift	float	Battery level difference between the beginning and the end of the interval
battery_charge_count	integer	number of times the phone has been connected to a charging source during the interval