NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

DOCTORAL THESIS

Scalable Semi-Supervised Structure Learning for Event Recognition

Author:

Evangelos MICHELIOUDAKIS

A thesis submitted in fulfilment of the requirements

for the degree of Doctoral of Philoshophy in the

National and Kapodistrian University of Athens

Department of Informatics and Telecommunications

in collaboration with the

National Centre for Scientific Research "Demokritos"

Institute of Informatics and Telecommunications

Software and Knowledge Engineering Laboratory

December 2023

ADVISORY COMMITEE

Georgios Paliouras, Researcher A, NCSR "Demokritos"

Alexander Artikis, Associate Professor, Univ. of Piraeus

Sergios Theodoridis, Professor, NKUA

EXAMINATION COMMITEE

Sergios Theodoridis
Professor, NKUA

Georgios Paliouras
Researcher A, NCSR "Demokritos"

Alexander Artikis
Associate Professor, Univ. of Piraeus

Panagiotis Rontogiannis
Professor, NKUA

Panagiotis Stamatopoulos
Assistant Professor, NKUA

Manolis Koumparakis
Professor, NKUA

George Vouros
Professor, Univ. of Piraeus

EXAMINATION DATE: December 11, 2023

“The question of whether computers can think
is like the question of whether submarines can swim.”
— Edsger W. Dijkstra

Abstract

Symbolic event recognition systems often rely on knowledge bases of event definitions, expressed in first-order logic, to detect event occurrences over time. Logical frameworks for representing and reasoning about events provide robust temporal reasoning and enable the automated discovery of event rules via Inductive Logic Programming (ILP). Although existing structure learning approaches ease the discovery of such rules in noisy data streams, they assume the existence of fully-labelled training sequences, which is unrealistic for most real-life applications. In this thesis we address the issue of scalable semi-supervised learning for event recognition. We propose two novel techniques for inferring the missing supervision on training sequences and enable learning event rules in the Event Calculus. First, we propose SPLICE, a framework that employs a graph-based method to derive labels for unlabelled data, based on their distance to their labelled counterparts. In order to adapt the graph-based method to first-order logic, we use a suitable structural distance for measuring the distance between sets of logical atoms. The labelling process is achieved online (single-pass) by means of a caching mechanism and the Hoeffding bound for filtering contradicting examples. However, SPLICE labelling may be compromised since its structural measure is agnostic of the feature semantics. Moreover, there is no guarantee about the quality of the labelling found in the local graphs that are built as the data stream in. To that end, we also propose SPLICE$^+$, a second method that improves upon SPLICE by employing a hybrid measure combining an optimised structural distance, and a data-driven one. The former is guided by feature selection, while the latter is a mass-based dissimilarity. In addition, SPLICE$^+$ improves the graph construction process, by storing a synopsis of the past, in order to achieve more informed labelling on the local graphs. We evaluate our approach on the task of complex event recognition by using a benchmark dataset for human activity recognition, a dataset for maritime monitoring, as well as a dataset for fleet management.

Περίληψη

Η συμβολική αναγνώριση γεγονότων συχνά βασίζεται σε μια βάση γνώσης, η οποία περιέχει κανόνες εκφρασμένους σε λογική πρώτης τάξης, και χρησιμοποιείται για την αναγνώριση γεγονότων σε ροές δεδομένων. Τέτοια λογικά συστήματα αναγνώρισης γεγονότων προσφέρουν εύρωστο χρονικό συμπερασμό και επιτρέπουν την αυτόματη κατασκευή κανόνων με τη χρήση Επαγωγικού Λογικού Προγραμματισμού. Παρότι οι υπάρχουσες μέθοδοι για την εκμάθηση σχεσιακών δομών διευκολύνουν την εύρεση τέτοιων κανόνων σε θορυβώδεις ροές δεδομένων, υποθέτουν ότι τα δεδομένα εκμάθησης είναι πλήρως επισημειωμένα, πράγμα το οποίο είναι μη ρεαλιστικό σε πραγματικές εφαρμογές. Σε αυτή τη διατριβή επιχειρούμε να αντιμετωπίσουμε το πρόβλημα της κλιμακούμενης ημί-επιπλεπόμενης μάθησης για αναγνώριση γεγονότων. Προτείνουμε δύο καινούργιες τεχνικές για να συμπεραίνουμε τις απούσες επισημειώσεις στα δεδομένα εκπαίδευσης και να μαθαίνουμε κανόνες εκφρασμένους σε Λογισμό Γεγονότων. Το SPLICE είναι ένα σύστημα το οποίο χρησιμοποιεί γράφους για να εξάγει επισημειώσεις για τα μη-επισημειωμένα δεδομένα με βάση τις αποστάσεις τους από τα αντίστοιχα επισημειωμένα. Για να εφαρμόσουμε τη μέθοδο αυτή σε λογική πρώτης τάξης, χρησιμοποιούμε μια απόσταση για σχεσιακές δομές η οποία μετράει την απόσταση μεταξύ λογικών ατόμων. Η διαδικασία της επισημείωσης των δεδομένων γίνεται σε ένα πέρασμα με χρήση ενός μηχανισμού μνήμης και του Hoeffding bound για να φιλτράρουμε αντικρουόμενα παραδείγματα. Παρόλα αυτά το SPLICE, βασίζεται σημαντικά στη μετρική που υπολογίζει τις αποστάσεις μεταξύ των λογικών ατόμων. Επιπλέον, δεν υπάρχει καμία εγγύηση για την ποιότητα των λύσεων στους τοπικούς γράφους που κατασκευάζονται από τη ροή δεδομένων. Συνεπώς, προτείνουμε μια δεύτερη μέθοδο, το SPLICE$^+$, η οποία χρησιμοποιεί μια υβριδική απόσταση που συνδυάζει μια βελτιστοποιημένη απόσταση σχεσιακών δομών και μια απόσταση που βασίζεται στη μάζα των δεδομένων. Η πρώτη καθοδηγείται από μια επιλογή χαρακτηριστικών, ενώ η δεύτερη στην εκτίμηση της μάζα των δεδομένων. Επιπλέον, το SPLICE$^+$ βελτιώνει την κατασκευή του γράφου με το να αποθηκεύει μια σύνοψη του παρελθόντος ώστε να πετύχει καλύτερες επισημειώσεις στους τοπικούς γράφους. Αξιολογούμε τις μεθόδους μας σε εφαρμογές σύνθετης αναγνώρισης γεγονότων χρησιμοποιώντας ένα σύνολο δεδομένων για αναγνώριση ανθρώπινων δραστηριοτήτων, ένα για αναγνώριση ναυτιλιακών συμβάντων και ένα για τη διαχείριση εμπορικών οχημάτων.

Acknowledgements

Pursuing a PhD is a challenging endeavour. You need devotion and hard work to succeed, since the journey is long and involves a lot of failures. One must learn to be patient and thorough in order to tackle the research problems that often seem hopeless of solving. Part of the road also involves learning how to communicate your thoughts and ideas to other people, understanding the significance of answering why and managing disappointment. However, the most important lesson I received is to be humble. The more I learn, the more I realise that our knowledge is far less than we assume, or to quote the words of the Greek philosopher Socrates "έν οίδα, ότι ουδέν οίδα".

The moments of my doctoral journey have been shared with colleagues, friends and family to which I am deeply grateful for their support. First and foremost, I would like to thank my supervisors Dr. Georgios Paliouras and Dr. Alexander Artikis from NCSR "Demokritos", for their guidance and limitless encouragement during the course of my research. I really appreciate all the valuable discussions, the ideas, the willingness to share their knowledge, and the patience they always show in my mistakes. Their help has proved indispensable. I am also very thankful to Prof. Sergios Theodoridis from NKUA that made this doctoral thesis possible.

During my PhD I had the privilege of working on several research projects, get acquainted with interesting ideas and collaborate with respectable researchers. Nikos Katzouris and Elias Alevizos, my closer colleagues from the Complex Event Recognition group, deserve special credit for sharing ideas, discussions and many funny moments. I would also like to thank my friend and colleague Anastatios Skarlatidis. Although he is no longer part of the group and he had no involvement in my PhD, he is a person that greatly inspired me and taught me a lot of things. I surely own the quality of my work to all of them.

I would also like to thank all my friends for being there, providing emotional support and time well-wasted. I am lucky enough, since many of them are computer engineers themselves and were happy to listen, discuss the difficulties and offer advice.

I am very lucky to have the parents that I do. I owe everything that I have accomplished to them and this cannot be justified by words. I also wish to thank my brother Nikos and my sister Chrysanthi for they are both a never-ending source of love and encouragement.

Last, but by no means least, I am deeply thankful to my beloved Aspasia for her endless affection and patience. Since I met her, she has been by my side, constantly supporting my failures and celebrating my successes. We have shared together my doctoral journey that she made so much easier. I believe that she deserves credit for this work, so it seems only natural that I dedicate my PhD to her.

Evangelos Michelioudakis
Athens, December 2023

Dedicated to my beloved Aspasia.

Contents

List of Figures

List of Tables

1 Introduction

"The only reason for time is so that everything doesn't happen at once."
— Albert Einstein

Today's information systems collect, share and process a significant amount of data that stream-in from a plethora of sources. Probes and sensors are deployed everywhere, from large-scale industrial infrastructures to hand-held physical devices, capture all kinds of information from the environment and monitor for desirable conditions. For instance, transport vehicles moving on-land and vessels sailing at sea report information about their location and operational status, using accelerometers, gyroscopes, GPS devices, etc. Sensitive areas employ surveillance cameras to record video footage for security purposes. More than ever, daily human activities generate huge amounts of media content, emails, social media posts, and even physiological data[1]. As time evolves, all these data flowing quietly across information systems, can be aggregated and correlated in order to become a source of significant knowledge that may unveil important insights. For example, in video surveillance, the recorded video frames may indicate that an accident may have happened, or in a computer network an abrupt increase in the volume of incoming network packets may point to a possible distributed denial of service attack. Most of these data, accompanied by their temporal occurrences, can be represented by *events*.

An event usually refers to something that happens and constitutes a fundamental concept for representing temporal pieces of information. It can be anything, from a simple sensor reading, like a temperature measurement, or a GPS coordinate to structured content like a financial transaction, a video frame and even complex things like an act of piracy occurring at sea. Events can be instantaneous or durative, persisting over time, and are often related to other events in several ways, e.g., temporally, spatially, causally, etc. More importantly, these related

[1] It is estimated that 1.7 MiB of information is created every second by every human being on the planet.

events tend to define useful patterns. For instance, in a sequence of video frames, the events that collectively represent some people standing still at the same time and in a close distance to each other, may altogether capture a situation that those particular persons are meeting in a public area.

The automatic detection of such event patterns in data streams has been facilitated by *complex event processing* [Etzion and Niblett, 2010], a set of methodologies and techniques designed for analysing, filtering and aggregating multiple independent sources of information, in order to discover and report these interesting events in real-time.

1.1 Symbolic Complex Event Recognition

Symbolic *complex event recognition*, or "*event pattern-matching*" [Luckham, 2002], is a tool of complex event processing that aims to detect event patterns in temporal data streams. These methods have received increasing attention in a variety of applications, including activity recognition [Rodríguez et al., 2013; Turaga et al., 2008] from video [Brendel et al., 2011; Prapas et al., 2018] or sensor readings [Gayathri et al., 2017], computer network attacks [Dousson and Maigat, 2007], distributed diagnosis of web-services [Guillou et al., 2008], traffic and transport management [Artikis et al., 2012; Tsilionis et al., 2019], fraud detection in online transactions [Schultz-Møller et al., 2009], business process management [Montali et al., 2013], medical applications, such as recognition of cardiac arrhythmias [Callens et al., 2008], epidemic spread [Chaudet, 2006], patient monitoring [Falcionelli et al., 2019; Kafali et al., 2017], maritime monitoring [Patroumpas et al., 2015; Pitsikalis et al., 2019], daily assisted living [Kim et al., 2019; Loreti et al., 2019; Storf et al., 2009], and the Internet of Things [Wang et al., 2013b].

Symbolic complex event recognition (CER) systems [Cugola and Margara, 2012], as depicted in Figure 1.1, consume input sequences of *simple derived events* (SDEs), match them against a knowledge base of high-level event patterns – defined on some formal language [Grez et al., 2019, 2020] – and recognise *complex events* (CEs) of interest. Each SDE usually corresponds to a single time-stamped observation and their occurrence is assumed not to depend on other events. Consider, for instance, a video tracking system detecting that a person is walking, by processing the raw video frames, and producing a walking SDE per frame. On the contrary, CEs can only happen when other related events (SDE or CE) have happened under some specific constraints (e.g., temporal and spatial relations). For example, a number of people are moving together. Therefore, CE recognition is associated with the occurrence of various SDEs and/or other CEs, involving

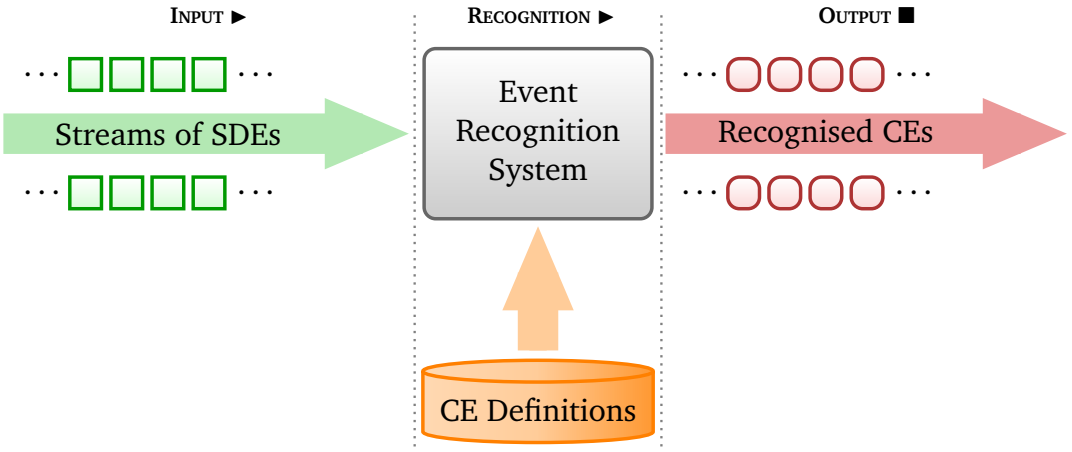

FIGURE 1.1: An illustration of the event recognition process. The CER system constantly monitors the input stream of SDEs for matches of the underlying CE patterns. The output of the system is a stream of recognised CEs (as per Skarlatidis [2014]).

multiple entities, e.g., people, vehicles or other objects, etc. CEs, are therefore, relational structures over other sub-events, either CE or SDE. The knowledge base of the CE patterns is either predefined by domain experts or learned from data and captures knowledge of significant CE for the target application.

1.2 Motivation

Event recognition systems that adopt a logic-based approach [Artikis et al., 2012], can naturally and compactly represent relational CE structures. They employ first-order temporal logical formalisms, such as the Event Calculus [Kowalski and Sergot, 1986], to model the effects and duration of event occurrences and rely on logical inference to perform the event recognition. They utilise formal and declarative semantics, in contrast to other CER systems that exhibit informal, procedural semantics, which is crucial in order to trace and validate the origins and effects of the recognised complex events [Paschke and Kozlenkov, 2009]. Moreover, they enable reasoning over complex relations among entities and can exploit background knowledge provided by domain experts, contrary to non-logic-based approaches. Examples of such CER systems include SAGE [Broda et al., 2009], ETALIS [Anicic et al., 2012], and RTEC [Artikis et al., 2015].

Typically, in an event recognition system, the event specification patterns are manually curated by human domain experts. Since CEs are defined as relational structures over actors and objects involved in an event, their manual derivation can be an expensive, time-consuming and error prone task [Artikis et al., 2010]. In addition, event recognition applications usually operate in noisy data streams of significant volume and velocity, see [Giatrakos et al., 2020] for an overview,

which further renders the synthesis of such relational dependencies unrealistic. To that end, machine learning methods for automatically constructing the structure of the CEs in a single pass over a data stream are essential [Dries and Raedt, 2009; Gama, 2010; Srinivasan and Bain, 2017].

Although learning logical theories for CER remains a challenging task, a couple of online relational learners have been proposed for the automated curation of CE patterns under uncertainty [Katzouris et al., 2016, 2018; Michelioudakis et al., 2016b], in the form of Event Calculus theories [Kowalski and Sergot, 1986; Mueller, 2008a] or probabilistic variants [Skarlatidis et al., 2015a,b]. These approaches stem from Inductive Logic Programming (ILP) [Fürnkranz et al., 2012; Muggleton and Raedt, 1994; Raedt, 2008] and Statistical Relational Learning (SRL) [Getoor and Taskar, 2007], fields that employ machine learning for logic programming and graphical models in order to provide tools and algorithms for learning logical theories from relational noisy data.

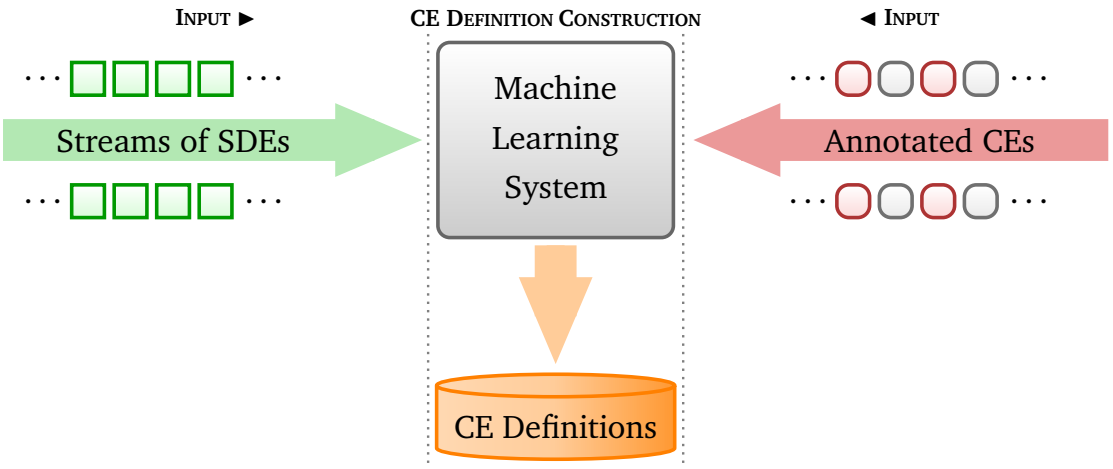

FIGURE 1.2: The online semi-supervised learning process. The machine learning system consumes the semi-supervised streams of SDEs (denoted by the grey-out annotation batches), in order to enhance and update the knowledge base of the CE patterns.

Although these techniques have facilitated the automated discovery of multi-relational dependencies in noisy environments, nonetheless, all of them assume that a fully labelled training sequence arrives for processing, which of course is an unrealistic assumption. Usually, either sparse, infrequent, labels become available on-stream or, more commonly, they are provided at the beginning of learning in the form of historical data. Figure 1.2 depicts the online semi-supervised structure learning process. The machine learning system consumes the streams of SDEs, as well as, the incomplete (semi-supervised) streams of CEs, usually in small, time-consecutive batches of data, called *micro-batches*, in order to construct and update the knowledge base of the CE patterns.

However, learning Event Calculus theories, or even simpler logical rules, using semi-supervised approaches and ILP remains a challenging and unexplored task. Existing work on semi-supervised learning assumes numerical data that are available at once during learning [van Engelen and Hoos, 2020]. On the other hand, only a couple of approaches have attempted to combine variations of co-training to ILP systems [Li and Guo, 2011; Soonthornphisaj and Kijsirikul, 2004]. These systems are only able to learn from small datasets and do not scale to the volumes of data collected in event recognition.

Therefore, the motivation behind this thesis is the development of scalable algorithms for the semi-supervised learning of CE patterns, in the form of Event Calculus theories, from temporal data. Throughout the thesis we focus on learning CE patterns that are represented as first-order logic rules, thus, we henceforth refer to them as CE rules.

1.3 Thesis Contribution

In this thesis, we focus on scalable semi-supervised learning methods for the automated construction of event rules in the form of Event Calculus theories, and present two graph-based techniques SPLICE and SPLICE$^+$ for inferring the missing supervision. To demonstrate the benefits of our proposed approaches, we present an experimental study on various aspects of these approaches using real-life event recognition datasets.

1.3.1 Semi-Supervised Online Learning for Complex Event Recognition

In order to address the issue of incomplete supervision, we propose SPLICE, a novel method for inferring the missing labels, using graph-based techniques [Zhu et al., 2009] and a distance function for first-order logic. Graph-based methods to semi-supervised learning essentially derive labels for unlabelled data, by computing their distance to their labelled counterparts. To that end, we adapt the *label propagation* approach proposed by [Zhu et al., 2003] to first-order logic, in order to operate over logical structures instead of numerical data. To do so, we utilise a structural measure [Nienhuys-Cheng, 1997], designed to compute the distance between logical atoms, and modify it using the Kuhn-Munkres algorithm [Kuhn, 1955], in order to accurately compute the distance over sets of logical atoms (Herbrand interpretations) that represent training examples.

The proposed supervision completion method operates in an online fashion (single-pass), by means of a caching mechanism that stores previously seen labels for future usage. The *Hoeffding bound* [Hoeffding, 1963a], a statistical tool that enables approximate globally-optimal decisions from locally-optimal ones, is used to filter out contradicting labels that may compromise the labelling accuracy. The completed training data can be subsequently used by any supervised structure learner to learn Event Calculus theories. In summary, the contributions of SPLICE are the following:

- An online supervision completion method using a caching mechanism to store labelled examples for future usage, and the Hoeffding bound to filter-out contradicting examples that may compromise the overall accuracy.
- An adaptation of the label propagation technique to first-order logic, using a structural distance for comparing logical atoms, and the Kuhn-Munkres algorithm for improving the accuracy of the distance calculation.
- The first system for semi-supervised online structure learning, combining online supervision completion and state-of-the-art structure learners, in order to learn Event Calculus theories for CER applications.

1.3.2 Semi-Supervised Online Learning Combining Structure and Mass-based Predicate Similarity

Although SPLICE aids the automated discovery of complex event rules in the presence of incomplete supervision, its distance measure may be compromised by irrelevant features or imbalanced supervision, since it is agnostic of the feature information. Moreover, SPLICE does not provide any guarantee about the labelling computed per micro-batch, compared to the one that would have been obtained if all examples were available as a large graph. In fact, as the micro-batch size gets smaller the harmonic solution produces labels that tend to be less dependent on the unlabelled examples. It is interesting to note that, in the case of true streaming (one example per micro-batch), the optimisation reduces to k-nearest neighbour classification (Chapelle et al. 2006, Section 11.6).

To address these issues, we propose an improved hybrid distance measure that combines the structural measure of SPLICE with a mass-based dissimilarity [Ting et al., 2019], employing mass estimation theory [Ting et al., 2013] to quantify the distance between examples of logical atoms. We further enhance the structural distance by performing feature selection, optimised for k-nearest neighbour (kNN) classification. To that end, we adapt Large-Margin Nearest Neighbour

(LMNN) [Weinberger and Saul, 2009], a state-of-the-art approach to metric learning, for the selection of informative logical predicates. Finally, in order to provide guarantees about the online labelling, we use a technique proposed by Wagner et al. [2018] that retains a synopsis of the graph in order to achieve more informed labelling across the incoming micro-batches. Similar to SPLICE, the completed training data can be subsequently used by any supervised structure learner. The contributions of SPLICE$^+$ are summarised as follows:

- The SPLICE$^+$ online semi-supervised learning system that retains a graph synopsis of temporally adjacent examples, in order to operate on large training sequences, and learns explainable composite event rules in the Event Calculus.
- SPLICE$^+$ adapts a metric learning technique for feature selection over logical atoms. The adapted technique is used as an informed structural distance which accounts for irrelevant and noisy predicates (features) that may compromise accuracy.
- A hybrid distance measure that combines the informed structural distance with a mass-based dissimilarity. The hybrid measure exploits both the labelled and unlabelled data to quantify the distance between examples of logical atoms. The resulting measure is a semi-supervised metric learning technique.

1.3.3 Publications

In the course of this doctoral thesis the following articles have been published:

Central Publications:

- Michelioudakis E., Artikis A. and Paliouras G. (2023)
  "Online Semi-Supervised Learning of Composite Event Rules by Combining Structure and Mass-Based Predicate Similarity"
  Machine Learning (*accepted*)
- Michelioudakis E., Artikis A. and Paliouras G. (2019)
  "Semi-Supervised Online Structure Learning for Composite Event Recognition",
  Machine Learning, 108(7), pp. 1085–1110

Peripheral Publications:

- Michelioudakis E. et al. (2022)
  "Parallel Model Exploration for Tumor Treatment Simulations",
  Computational Intelligence, 38(4), pp. 1379–1401
- Stavropoulos V., Michelioudakis E., Akasiadis C., Artikis A. (2022)
  "Resource-Effective Exploration of Tumor Treatments with Multi-scale Simulations",
  Hellenic Conference on Artificial Intelligence (SETN), pp. 1–10
- Katzouris N., Michelioudakis E., Artikis A., Paliouras G. (2018)
  "Online Learning of Weighted Relational Rules for Complex Event Recognition",
  European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD), pp. 396–413

1.4 Thesis Outline

The remainder of this thesis is organised as follows. In Chapter 2 we briefly present the Event Calculus formalism and the OLED online learner for CER applications. Then, we provide the necessary background on graph-based methods for semi-supervised learning, an appropriate distance function for logical structures and a metric learning technique optimised for kNN classification. We also review the literature on semi-supervised learning approaches, distances for first-order logic, metric learning and feature subset selection methods. In Chapter 3 we present SPLICE, our first proposed approach for online semi-supervised learning of CE rules, while in Chapter 4 we describe our second improved approach (SPLICE$^+$) towards more robust supervision completion. In Chapter 5 we present the experimental evaluation on real datasets for activity recognition, maritime monitoring and fleet management. Finally, in Chapter 6, we discuss open issues, provide directions for future research and conclude.

2 Background

"The beginning is the most important part of the work."
— Plato

This chapter provides the necessary background material for the thesis. We begin by briefly presenting the Event Calculus formalism, as well as, the basic functionality of the OLED system for learning CE rules from data streams. Then, in Section 2.2 we present graph-based semi-supervised learning using Gaussian Random Fields and an online variant based on a graph synopsis. In Section 2.3 we discuss a simple distance metric for comparing sets of logical atoms, while in Sections 2.4 and 2.5 we present a metric learning technique method for kNN classification and a data-driven dissimilarity based on mass estimation. The chapter concludes with a review of the related work.

2.1 Event Calculus and Structure Learning

One way of representing the CE rules is by using the *discrete* Event Calculus (DEC) [Mueller, 2008b]. The ontology of DEC consists of *time-points*, *events* and *fluents*. The underlying time model is linear and represented by integers. A *fluent* is a property whose value may change over time by the occurrence of particular *events*. DEC includes the core domain-independent axioms of the Event Calculus, which determine whether a fluent holds or not at a specific time-point. This axiomatisation incorporates the common sense *law of inertia*, according to which fluents persist over time, unless they are affected by an event occurrence. Event occurrences are denoted by the `HappensAt` predicates, while `HoldsAt` predicates denote whether a fluent holds. The `InitiatedAt` and `TerminatedAt` predicates express the conditions in which a fluent is initiated or terminated, and are triggered by `HappensAt` predicates. The core DEC axioms are defined as follows:

$$\texttt{HoldsAt}(f, t{+}1) \Leftarrow \texttt{InitiatedAt}(f, t) \quad (2.1)$$

$$\texttt{HoldsAt}(f, t{+}1) \Leftarrow \texttt{HoldsAt}(f, t) \wedge \neg\texttt{TerminatedAt}(f, t) \quad (2.2)$$

$$\neg\texttt{HoldsAt}(f, t{+}1) \Leftarrow \texttt{TerminatedAt}(f, t) \quad (2.3)$$

$$\neg\texttt{HoldsAt}(f, t{+}1) \Leftarrow \neg\texttt{HoldsAt}(f, t) \wedge \neg\texttt{InitiatedAt}(f, t) \quad (2.4)$$

Variables and functions start with a lower-case letter, while predicates start with an upper-case letter. Axioms (2.1) and (2.2) express when a fluent holds, while axioms (2.3) and (2.4) denote the conditions in which a fluent does not hold. In CER, as we have formulated it here, the truth values of the composite events (CEs) of interest — the 'query atoms' — are expressed by means of the `HoldsAt` predicate. The incoming 'simple, derived events' (SDEs) are represented by means of `HappensAt`, while any additional contextual information is represented by domain-dependent predicates. The SDEs and such contextual information constitute the 'evidence atoms'. This way, CEs may be defined by means of `InitiatedAt` and `TerminatedAt` predicates, stating the conditions in which a CE is initiated and terminated.

To learn DEC theories, online structure learning methods may be used in order to learn efficiently from data streams. OLED [Katzouris et al., 2016] is an online learner, based on Inductive Logic Programming [Muggleton and Raedt, 1994], constructing CE rules in the Event Calculus, in a single pass over the data stream. OLED constructs rules by encoding each positive example, arriving at the input stream, into a so-called *bottom rule*, i.e., a most specific rule of the form $\alpha \leftarrow \delta_1 \wedge \ldots \wedge \delta_n$, where α is an `InitiatedAt` or `TerminatedAt` atom and δ_i are relational features (e.g., SDEs). A bottom rule is typically too restrictive to be useful, thus, OLED searches the space of all possible rules that θ-subsume the bottom rule. To that end, OLED starts from the most-general rule and gradually specialises that rule, in a top-down fashion, by adding δ_i's to its body and using a rule evaluation function to assess the quality of each constructed specialisation. OLED's single-pass strategy draws inspiration from the VFDT algorithm (Very Fast Decision Trees) [Domingos and Hulten, 2000] which is based on the Hoeffding bound, a statistical tool that allows to approximate the quality of a rule on the entire input using only a subset of the data. Thus, in order to decide between specialisations, OLED accumulates observations from the input stream until the difference between the best and the second-best specialisation satisfies the Hoeffding bound.

Although OLED facilitates the discovery of CE rules, it still is a supervised learner and in the presence of unlabelled training examples it imposes a closed-world assumption, that is, it assumes everything not known is false, i.e., a negative

example. This assumption may seriously compromise the learning task or even worse, render it impossible if very little supervision is available, which is a common scenario in real-life applications.

2.2 Graph-based Semi-Supervised Learning

Graph-based semi-supervised learning techniques [Chapelle et al., 2006; Zhu et al., 2009] construct a graph $\mathcal{G}$, whose vertices V represent the labelled and unlabelled examples in a given dataset and the edges E reflect the pairwise similarities of these examples. Given such a graph, [Blum and Chawla, 2001] proposed to formulate the learning task as a graph *mincut* or *st-cut* problem. In the binary case, the idea is to remove a minimal set of edges, so that the graph is cut into two disjoint sets of vertices; one holding positive and the other negative examples.

More formally, consider a training sequence consisting of l labelled instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and u unlabelled ones $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$. The labelled instances are pairs of a label y_i and a D-dimensional numerical feature vector $\mathbf{x}_i = (x_1, \ldots, x_D) \in \mathbb{R}^D$ of input values, while the unlabelled ones are feature vectors with unknown label. Each of these instances represents either a labelled or an unlabelled vertex of the graph $\mathcal{G}$. These vertices are then connected by undirected weighted edges that encode their similarity according to a given distance function. Consequently, the labelled vertices can be used to determine the labels of the unlabelled ones. Once the graph is built, the task reduces into assigning y values to the unlabelled vertices. Thus, the goal is to find a function $f(\mathbf{x}) \in \{-1, 1\}$ over the vertices, where -1 is a negative label and 1 a positive one, such that $f(\mathbf{x}_i) = y_i$ for labelled instances, and the cut size (the number of removed edges) is minimised in order for the unlabelled ones to be assigned optimal values.

2.2.1 Harmonic Function Method

The mincut formulation proposed by Blum and Chawla [2001] can be represented as a regularised minimisation problem, by using an appropriate loss function, forcing the labelled vertices to retain their values and a regularisation factor controlling the cut size. The cut size is the sum of the weights w_{ij} corresponding to connected vertices i and j having different labels, and is computed as follows:

$$\sum_{i,j:\, f(\mathbf{x}_i) \neq f(\mathbf{x}_j)} w_{ij} = \sum_{i,j=1}^{l+u} w_{ij} \big(f(\mathbf{x}_i) - f(\mathbf{x}_j)\big)^2 \tag{2.5}$$

Equation (2.5) is an appropriate measure of the cut size, since it is affected only by edges for which $f(\mathbf{x}_i) \neq f(\mathbf{x}_j)$. Note that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are not connected, then $w_{ij} = 0$ by definition, while if the edge exists and is not cut, then $f(\mathbf{x}_i) - f(\mathbf{x}_j) = 0$. Thus, the cut size is well-defined even when summing over all vertex pairs. Assuming a loss λ per labelled vertex, the loss for labelled instances should be zero if $f(\mathbf{x}_i) = y_i$ and λ otherwise. Thus, the loss function is defined as follows:

$$\ell\big(\mathbf{x}_i,\, y_i,\, f(\mathbf{x}_i)\big) = \lambda \left(y_i - f(\mathbf{x}_i)\right)^2 \tag{2.6}$$

Combining the loss function of Eq. (2.6) and the cut size, as expressed by Eq. (2.5), as a regularisation factor, the regularised mincut problem is formulated as follows:

$$\min_{f:f(\mathbf{x})\in\{-1,\,1\}} \lambda \sum_{i=1}^{l} \big(y_i - f(\mathbf{x}_i)\big)^2 + \sum_{i,j=1}^{l+u} w_{ij} \big(f(\mathbf{x}_i) - f(\mathbf{x}_j)\big)^2 \tag{2.7}$$

Note that Eq. (2.7) is an integer optimisation problem since f is constrained to produce discrete values. Although efficient polynomial–time algorithms exist to solve the mincut problem, still the formulation has a particular defect. There could be multiple equally good solutions; a label may be positive in one of the solutions, and negative in the rest.

An alternative formulation proposed by Zhu et al. [2003], overcomes these issues, by modelling the regularisation term of Eq. (2.5), that is the cut-size, as the energy function of a Gaussian Random Field. The minimisation of the energy function yields a probabilistic label assignment instead of discrete values. Moreover, the minimum energy function has been shown to respect the harmonic property, i.e., the value of f at each unlabelled vertex is the average of f of the neighbouring vertices. In the context of semi-supervised learning, a harmonic function is a function that retains the values of the labelled data and satisfies the weighted average property on the unlabelled data:

$$\begin{aligned} f(\mathbf{x}_i) &= y_i,\ i = 1, \ldots, l \\ f(\mathbf{x}_i) &= \frac{\sum_{j=1}^{l+u} w_{ij} f(\mathbf{x}_j)}{\sum_{j=1}^{l+u} w_{ij}},\ i = l{+}1, \ldots, l{+}u \end{aligned} \tag{2.8}$$

The former part of formula (2.8) enforces that the labelled vertices retain their values, while the latter averages the labels of all neighbouring vertices of a given vertex, according to the weights of their edges. Therefore, the value assigned to

each unlabelled vertex is the weighted average of its neighbours. The harmonic function leads to the same solution of the problem as defined in Eq. (2.7), except that f is relaxed to produce real values. The main benefit of the continuous relaxation is that a unique optimal closed–form solution exists for f, in terms of the Laplacian matrix of the graph. The drawback of the relaxation is that the solution is a real value for each unlabelled example and does not directly correspond to a label. This issue can be addressed by thresholding f at zero to produce discrete labels or by class mass normalisation [Zhu et al., 2003].

2.2.2 Temporal Label Propagation

Traditional graph-based methods to semi-supervised learning [Zhu et al., 2009] are offline, i.e., they assume that all labelled and unlabelled data are stored in memory and thus are available during the optimisation that yields the harmonic solution. However, that is an unrealistic assumption in online processing of data streams.

Temporal Label Propagation (TLP) [Wagner et al., 2018] has been proposed for fast-moving data streams. TLP stores a synopsis of the full history of the stream in order to retain accumulated knowledge for both labelled and unlabelled examples and incorporate it into subsequent optimisations. To that end, TLP draws inspiration from the connection of label propagation to the theory of electric networks [Zhu et al., 2003] and, in particular, the idea of the short-circuit operator. The latter allows for a graph $\mathcal{G}$ to be encoded into a smaller (re-weighted) graph, using only a subset V_τ of the actual vertices V, called *terminals*. The reduced graph $\mathcal{G}\langle V_\tau \rangle$ is called short-circuit graph and it is known to retain the global properties of $\mathcal{G}$; most importantly, it preserves the effective weights between every pair of terminal vertices. It is proved by Wagner et al. [2018] that the aforementioned property allows for the harmonic solution to be preserved in the synopsis graph encoded by the terminal vertices.

The Laplacian matrix of $\mathcal{G}\langle V_\tau \rangle$, required to obtain the harmonic solution, is given by the *Schur Complement* [Dörfler and Bullo, 2013]. Since computing the Shur Complement is as expensive as computing the harmonic solution on the entire graph $\mathcal{G}$, it provides no substantial speed-up for offline label propagation. However, in order for TLP to operate in a online fashion it computes $\mathcal{G}\langle V_\tau \rangle$ as a sequence of local operations, called *star-mesh transforms*. The latter is a direct consequence of the sequential property of Schur complement (Zhang 2005, Theorem 4.10; Dörfler and Bullo 2013, Lemma III.1).

Definition 2.1. A star-mesh transformation on a vertex v_o of a given graph $\mathcal{G}{=}(V, E, \mathbf{W})$ is defined as follows:

1. *Star*: Remove v_o from $\mathcal{G}$ together with its set E_o of incident edges $(v_o, v) \in E_o$.

2. *Mesh*: For every pair of vertices $v, v' \in V$ such that $(v_o, v) \in E_o$ and $(v_o, v') \in E_o$, add the edge (v, v') to E with weight $w_{v,v'} = w_{v,v_o} w_{v_o,v'} / degree(v_o)$. If (v, v') is already in E, then add the new weight $w_{v,v'}$ to its current weight.

The intuition is to apply star-mesh transforms as the data arrive for processing, in order to continuously update the in-memory graph synopsis and deliver labels for the incoming unlabelled examples by computing the harmonic solution on the compressed graph. The star-mesh transforms remove edges by meshing their weights with the remaining graph, so that the information provided by the removed vertex v_o remains encoded. Thus, the synopsis retains the ability to compute the harmonic solution for the rest of the vertices as if v_o was still in the graph (Wagner et al. 2018, Theorem 4.1).

More formally, consider a (possibly infinite) data stream $\{v_t\}_{t=1}^{\infty}$ of incoming example vertices that can be either labelled or unlabelled. TLP maintains a graph $\mathcal{G}\langle V_\tau \rangle$ that stores the τ more recent unlabelled examples, in addition to a pair of labelled vertex clusters containing all the labelled examples seen so far. When a new unlabelled example arrives, TLP appends it to $\mathcal{G}\langle V_\tau \rangle$, connecting it to the other vertices and "evicting" the oldest unlabelled example by applying the star-mesh transform of Definition 2.1. In the simplest case, where a labelled example arrives, this process just appends it to the appropriate cluster vertex, t hus always maintaining $\tau + 2$ vertices. The harmonic solution for each new unlabelled example is then computed on $\mathcal{G}\langle V_\tau \rangle$ and it is provably equal to the one computed on the entire training stream seen so far.

2.3 Distance between Herbrand Interpretations

Distance functions constitute an essential component of graph-based methods to semi-supervised learning and in fact control the quality of the solution. In the case of numerical data, the Euclidean distance, the Gaussian kernel or Radial Basis Functions are common choices, as are matching distances for categorical data. However, in the presence of relational data there is a need for structure-based distances.

A technique proposed by Nienhuys-Cheng [1997] derives a distance for tree structure formalisms and thus provides a generic and natural approach for syntactic comparison of ground logical atoms present in a Herbrand interpretation. The distance function is defined on a set of expressions (namely ground atoms and

ground terms), motivated by the structure and complexity of the expression, as well as the symbols used therein. Let $\mathcal{E}$ be the set of all expressions in a first-order language and $\mathbb{R}$ the set of real numbers. The distance $d : \mathcal{E} \times \mathcal{E} \mapsto \mathbb{R}$ over expressions $\mathcal{E}$, bounded by 1, is defined as follows:

$$\begin{aligned} &d(e,\, e) = 0,\ \forall e \in \mathcal{E} \\ &d(p(s_1,\, \ldots,\, s_k),\, q(t_1,\, \ldots,\, t_r)) = 1,\ p \neq q \vee k \neq r \\ &d(p(s_1,\, \ldots,\, s_k),\, q(t_1,\, \ldots,\, t_k)) = \tfrac{1}{2k} \textstyle\sum_{i=1}^{k} d(s_i,\, t_i),\ p = q \end{aligned} \tag{2.9}$$

The first formula states that the distance of an expression to itself is zero. The second one states that if predicates p and q are not identical, either in terms of symbol or arity, then their distance is one, because they refer to different concepts. We assume that the negation of a predicate p has always distance 1 from p, and thus, it can be seen as a special case of the second formula, where $q = \neg p$. In case p and q are identical, then their distance is computed recursively by the distance of the terms therein. The distance d is also used by Nienhuys-Cheng [1997] over subsets of $\mathcal{E}$, i.e., sets of ground atoms, by means of the Hausdorff metric [Hausdorff, 1962]. Informally, the Hausdorff metric is the greatest distance you can travel between two sets of points, given for each point in one set the closest point in other set.

The main drawback of the Hausdorff metric is that it does not capture much information about the two sets as it is completely determined by the most distant elements of the sets to their nearest neighbour in the other set [Raedt, 2008; Ramon and Bruynooghe, 1998]. Thus, it may not be representative of the overall dissimilarity of the two sets. Formally, given the sets $\mathcal{E}_1$ and $\mathcal{E}_2$, their Hausdorff distance is computed as follows:

$$\max\{\sup_{x \in \mathcal{E}_1} \inf_{y \in \mathcal{E}_2} d(x, y), \sup_{y \in \mathcal{E}_2} \inf_{x \in \mathcal{E}_1} d(x, y)\}$$

The overall distance for these sets would be represented by one of the pairwise distances, namely the maximum distance among the minimum ones. Moreover, this approach allows one element in one set to match with multiple elements in the other set, which is undesirable because some elements may have no match and thus may be ignored in the resulting distance value. As stated by Raedt [2008], these limitations motivate the introduction of a different notion of matching between two sets, which associates one element in a set to at most one other element. Moreover, Ramon and Bruynooghe [1998] developed a framework for distances over sets of logical atoms that uses a given mapping between the elements in the sets to compute the overall distance among these elements.

2.4 Large Margin Nearest Neighbour Metric Learning

Graph-based semi-supervised learning relies on the *cluster assumption*, that is, similar examples should yield the same labelling, and thus the quality of the distance measure is crucial to the quality of the labels. A common issue with distance measures is that they are agnostic to the semantics of the input features. As a consequence, their measurements may suffer in the presence of irrelevant or noisy features.

Large-margin nearest neighbour metric learning (LMNN) [Weinberger and Saul, 2009] is a state-of-the-art technique that learns a distance pseudo-metric targeted to kNN classification. Intuitively, LMNN attempts to increase the number of training examples in a neighbourhood that share the same label. To that end, it learns a linear transformation of the input space, on which it uses the Euclidean distance. Euclidean distances can be parametrised by a matrix $\mathbf{T}$ that applies a linear transformation to a data vector $\mathbf{x}$ as follows:

$$d_{\mathbf{T}}(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{T}(\mathbf{x}_i - \mathbf{x}_j)||_2^2 \tag{2.10}$$

Euclidean distances in the transformed space can equivalently be viewed as Mahalanobis distances in the original space, by introducting a square matrix $\mathbf{M} = \mathbf{T}^{\mathrm{T}}\mathbf{T}$ as follows:

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}\mathbf{M}(\mathbf{x}_i - \mathbf{x}_j) \tag{2.11}$$

where the Euclidean distance can be recovered by setting $\mathbf{M} = \mathbf{I}$.

In order to optimise kNN classification, one seeks a linear transformation such that nearest neighbours computed by the distance in Eq. (2.10) share the same labels. Towards that goal, LMNN minimises a loss function consisting of two terms, one which pulls *target neighbours* closer together, and another which pushes differently labelled examples apart.

The first term penalises large distances between nearby instances that share the same label and should be nearest neighbours. In terms of the transformation of the input space, the sum of these squared distances is given by

$$\varepsilon_{pull}(\mathbf{M}) = \sum_{i,j \in \mathcal{N}_i^k} d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j), \tag{2.12}$$

where $\mathcal{N}_i^k$ denotes the set of *target* k-nearest neighbours of the instance $\mathbf{x}_i$. The target neighbours of $\mathbf{x}_i$ are those instances that we desire to be the closest to $\mathbf{x}_i$. In the simplest case, the target neighbours may be all example instances having the same label to $\mathbf{x}_i$.

The second term penalises small distances between differently labelled examples, called *impostors*. More formally, for an example $\mathbf{x}_i$ with label y_i and target neighbour $\mathbf{x}_j$, an impostor is any example $\mathbf{x}_l$ with label $y_l \neq y_i$ such that:

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) \leq d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + 1 \tag{2.13}$$

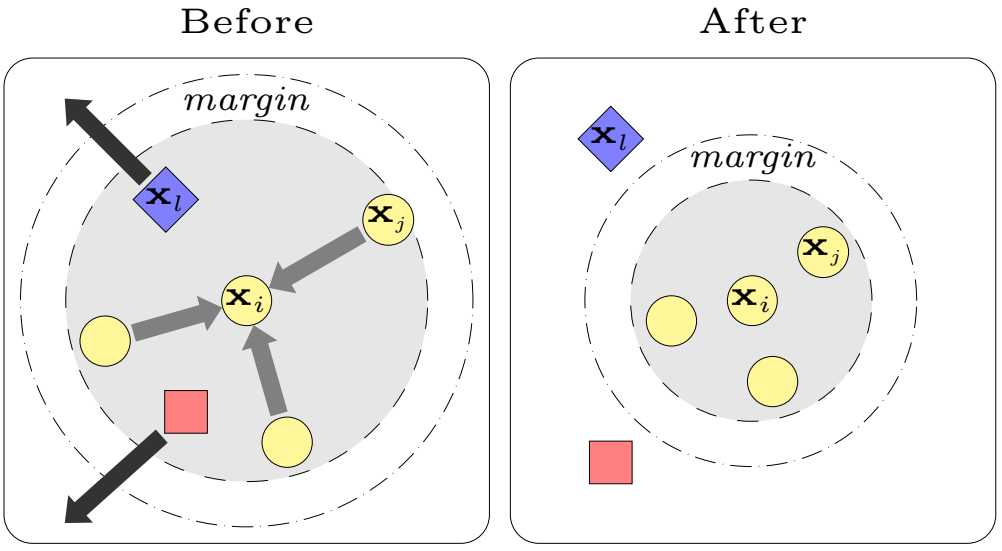

FIGURE 2.1: The neighbourhood of $\mathbf{x}_i$ before/after optimisation. A distance metric is learned so that: (i) target neighbours (yellow circles) lie within a small radius from $\mathbf{x}_i$; (ii) impostors (blue diamond, red square) lie outside this smaller radius by a finite margin. Arrows indicate pull/push operations (after [Weinberger and Saul, 2009]).

In other words, an impostor $\mathbf{x}_l$ is any differently labelled example that invades the perimeter, plus unit margin, defined by any target neighbour $\mathbf{x}_j$ of the example $\mathbf{x}_i$. Therefore, the second term penalises violations of the above inequality as follows:

$$\varepsilon_{push}(\mathbf{M}) = \sum_{i,j \in \mathcal{N}_i^k} \sum_{l} (1 - y_{il}) \left[1 + d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l)\right]_+ \tag{2.14}$$

where the indicator variable $y_{il} = 1$, if and only if $y_i = y_l$, while $y_{il} = 0$ otherwise. Moreover, $[z]_+ = \max(z, 0)$ denotes the standard hinge loss which monitors the inequality of Eq. (2.13). If the inequality does not hold (i.e., the input $\mathbf{x}_l$ lies a safe distance away from $\mathbf{x}_i$), then the hinge loss has a negative argument

and makes no contribution to the overall loss. The combined loss derived from Eq. (2.12) and Eq. (2.14) is as follows:

$$\varepsilon(\mathbf{M}) = (1-\mu)\,\varepsilon_{pull}(\mathbf{M}) + \mu\,\varepsilon_{push}(\mathbf{M}) \tag{2.15}$$

were the weighting parameter $\mu \in [0,1]$ balances the two goals. Fig. 2.1 illustrates the idea of LMNN. Before learning, an input $\mathbf{x}_i$ may have both target neighbours $\mathbf{x}_j$ and impostors $\mathbf{x}_l$ in its local neighbourhood. After optimisation, the impostors are pushed outside the perimeter established by the target neighbours and a finite margin exists between the perimeter and the impostors.

A variant of the LMNN technique, proposed in Chen et al. [2009], aims to learn a vector $\mathbf{m}$ of feature weights, instead of a distance, by assuming that $\mathbf{M}$ is a diagonal matrix with $\mathbf{M}_{pp} = m_p \geq 0$, and m_p is the weight of the pth feature. Expanding the loss function depicted in Eq. (2.15):

$$\begin{aligned}\varepsilon(\mathbf{m}) = (1-\mu) \sum_{i,j\in\mathcal{N}_i^k} d_{\mathbf{m}}(\mathbf{x}_i,\mathbf{x}_j) + \\ \mu \sum_{i,j\in\mathcal{N}_i^k}\sum_{l}(1-y_{il})\big[1+d_{\mathbf{m}}(\mathbf{x}_i,\mathbf{x}_j)-d_{\mathbf{m}}(\mathbf{x}_i,\mathbf{x}_l)\big]_+\end{aligned} \tag{2.16}$$

The minimisation of the simplified objective function can be represented as a linear optimisation problem with linear constraints:

$$\begin{aligned}&\textbf{minimise} && (1-\mu)\sum_{i,j\in\mathcal{N}_i^k} ||\mathbf{m}(\mathbf{x}_i-\mathbf{x}_j)||^2 + \mu \sum_{i,l,j\in\mathcal{N}_i^k}(1-y_{il})\xi_{ijl} \\ &\textbf{subject to} && \textbf{(1)}\ ||\mathbf{m}(\mathbf{x}_i-\mathbf{x}_l)||^2 - ||\mathbf{m}(\mathbf{x}_i-\mathbf{x}_j)||^2 \geq 1-\xi_{ijl} \\ &&& \textbf{(2)}\ \xi_{ijl} \geq 0 \\ &&& \textbf{(3)}\ \mathbf{m} \geq 0\end{aligned} \tag{2.17}$$

The non-negative slack variables ξ_{ijl} mimic the effect of the hinge loss. In particular, each slack variable $\xi_{ijl} \geq 0$ is used to measure the amount by which the margin inequality in Eq. (2.13) is violated. The optimal weight vector $\mathbf{m}^\star$ captures the importance of each input feature instead of the covariance matrix of the Mahalanobis distance.

2.5 Mass-based Dissimilarity

Supervised learning approaches to feature selection usually require explicit or implicit computation of the information/importance per feature using the labels available in the training examples. However, in a semi-supervised learning task, that information may be inaccurate due the limited number of labels. Therefore, such criteria are not always reliable and their optimality guarantees suffer from the fact that only very few training examples are used during the optimisation.

Ting et al. [2019] recently proposed a mass-based dissimilarity, that employs estimates of the probability mass to quantify the dissimilarity of two points rather than the classic geometric models. Geometric approaches, such as the Euclidean distance, depend on the geometric positions of data points alone to derive a measurement. Instead mass dissimilarity measures mainly depend on the distribution of the data. The intuition is that the distance of two points primarily depends on the amount of probability mass in the region of space covering the two points. Thus, two points in a dense region are less similar to each other than two points of the same interpoint distance in a sparse region.

More formally, let H denote a hierarchical partitioning of a space $\mathbb{R}^q$ into a set of non-overlapping regions that collectively span $\mathbb{R}^q$. Moreover, each region in the hierarchy corresponds to the union of its child regions. Let $\mathcal{H}(D)$ denote the set of all such hierarchical partitions H that are admissible under a dataset D, such that each non-overlapping region contains at least one point from D. Then the smallest region covering a pair of points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^q$, with respect to a hierarchical partitioning model H of $\mathbb{R}^q$, is defined as:

$$R(\mathbf{x}, \mathbf{y}|H) = \underset{r \in H \ s.t. \{\mathbf{x},\mathbf{y}\} \in r}{\operatorname{argmax}} \operatorname{depth}(r; H)$$

where $\operatorname{depth}(r; H)$ is the depth of region r in the hierarchical model H.

Suppose that a dataset D is sampled from an unknown probability density function F. Then, the mass-based dissimilarity of $\mathbf{x}$ and $\mathbf{y}$ w.r.t. D is defined as the expectation of the probability that a randomly chosen point would lie in the region $R(\mathbf{x}, \mathbf{y}|H)$:

$$m(\mathbf{x}, \mathbf{y}|D) = E_{\mathcal{H}(D)}\big[P_F(R(\mathbf{x}, \mathbf{y}|H; D))\big]$$

where the expectation is computed over all possible partitionings $\mathcal{H}(D)$ of the data. In practice, however, mass-based dissimilarity can be estimated from a finite number of partitioning models $H_p \in \mathcal{H}(D),\ p = 1, \ldots, T$ as follows:

$$\tilde{m}(\mathbf{x}, \mathbf{y}|D) = \frac{1}{T} \sum_{p=1}^{T} \tilde{P}(R(\mathbf{x}, \mathbf{y}|H_p; D)) \tag{2.18}$$

where $\tilde{P}(R) = \frac{1}{|D|} \sum_{\mathbf{z} \in D} \mathbb{1}(\mathbf{z} \in R)$ estimates the probability of the region R by counting the data points in that region; and $\mathbb{1}(\cdot)$ denotes an indicator function. Thus, the probability of the data falling into the smallest region containing both $\mathbf{x}$ and $\mathbf{y}$, is analogous to the shortest distance between them measured in the geometric model.

In order to generate partitioning models H, a recursive partitioning scheme is employed based on an *Isolation Forest* [Liu et al., 2008]. Isolation Forest is essentially an ensemble of random trees, called *Isolation Trees*. Each Isolation Tree is built independently using a subset of the data. At each internal node of the tree a random split is made to partition the data at that node into two non-empty subsets. The process is repeated recursively until either every data point is isolated or a given maximum tree height is reached.

Subsequently the resulting Isolation Forest can be used to compute the mass-based dissimilarity of Eq. (2.18). Since each Isolation Tree essentially represents a partitioning H_p, the mass-based dissimilarity can be defined as:

$$\tilde{m}(\mathbf{x}, \mathbf{y}) = \frac{1}{T} \sum_{p=1}^{T} \frac{|R(\mathbf{x}, \mathbf{y}|H_p)|}{|D|} \tag{2.19}$$

where $\frac{|R(\mathbf{x},\mathbf{y}|H_p)|}{|D|}$ estimates the probability of region R, as denoted by $\tilde{P}(R)$ in Eq. (2.18). To compute Eq. (2.19), $\mathbf{x}$ and $\mathbf{y}$ are passed through each Isolation Tree to find the mass of the deepest node containing both $\mathbf{x}$ and $\mathbf{y}$ i.e., $\sum_i |R(\mathbf{x}, \mathbf{y}|H_p)|$. Finally, $\tilde{m}$ is the mean of these masses over the T trees.

2.6 Related Work

In this section we discuss relevant work from the literature. The discussion is divided into four parts. In the first part (Section 2.6.1), we overview learning systems, like OLED, that attempt to learn CE rules in the form of Event Calculus theories. In the second part (Section 2.6.2), we discuss existing propositional

semi-supervised learning methods and argue against their application for supervised learning of CE rules from data streams. In the third part (Section 2.6.3) we present existing relational distances that can be potentially used for graph-based semi-supervised learning, while in the final part (Section 2.6.4), we discuss feature selection and metric learning techniques and their applications to semi-supervised learning.

2.6.1 Learning Complex Event Rules in the Event Calculus

Learning CE rules from sensory input is a challenging task that is receiving increasing attention in the literature, since it constitutes a limiting factor to CER applications. Some recent approaches attempt to learn propositional CE rules in the form of a domain-specific language [Bruns et al., 2019; Margara et al., 2014; Mousheimish et al., 2017]. However, in order to learn relational CE rules in the form of Event Calculus (EC) theories, structure learning techniques are employed, stemming from Inductive Logic Programming (ILP) [De Raedt and Dehaspe, 1997; Quinlan, 1990] and probabilistic graphical models (PGM) [Gogate et al., 2010; Lee et al., 2006; McCallum, 2003; Pietra et al., 1997].

Although several ILP approaches have addressed the problem of learning normal logic programs, see [Katzouris, 2017] for an overview, most of them cannot learn EC theories, since they do not support non-Observational Predicate Learning [Muggleton, 1995] and restrict the usage of Negation as Failure. Ray [2009] proposed XHAIL, a technique that can learn EC logic programs, but does not scale to large datasets. Although Katzouris et al. [2015] extended XHAIL to operate in an incremental way, in order to efficiently learn EC theories from data that arrive over time, nevertheless, ILED is still inappropriate for real-life CER applications because it cannot handle noisy data. The ILASP system [Law et al., 2016, 2018] overcomes the shortcomings of ILED and learns EC theories from noisy data, by attaching penalties to uncovered examples and seeking a set of CE rules that minimise the total cost of these examples. However, in most CER applications data arrive at a high velocity, and thus learning methods should operate within tight memory and time constraints, while both ILED and ILASP, may require multiple passes over the training data.

OLED [Katzouris et al., 2016], as presented in Section 2.1, overcomes these issues, by employing the Hoeffding bound [Hoeffding, 1963b], to approximate the quality of the learned CE rules using only an available subset of the data, thus enabling the efficient construction of EC theories in a single pass over the data stream. Although OLED can efficient learn CE rules from noisy data streams, it cannot perform probabilistic inference. To that end, Michelioudakis et al.

[2016b] proposed OSLα, an approach based on Markov Logic Networks (MLN) [Richardson and Domingos, 2006], that learns MLN–EC [Skarlatidis et al., 2015b] theories — a probabilistic variant of DEC — by adapting the procedure of OSL [Huynh and Mooney, 2011]. Although OSLα inherits the probabilistic properties of MLNs, its structure learning component is sub-optimal, i.e., it tends to generate large sets of clauses, many of which have low heuristic value, and therefore it is much slower than OLED. An in-depth experimental comparison of these methods can be found in [Katzouris et al., 2018].

These techniques have been applied to a variety of CER applications, such as e.g., fraud management [Artikis et al., 2017], traffic management [Michelioudakis et al., 2016a], community detection [Athanasopoulos et al., 2018], and natural language processing [Wu et al., 2018]. Nevertheless, they all assume a fully labelled training input to achieve generalisation, and in the presence of unlabelled training examples they impose closed-world assumption, that is, they assume everything not known is false, i.e., negative examples. This assumption can seriously compromise the learning task or even worse render it impossible if very few labels are available, which is a common scenario in real-life applications.

2.6.2 Semi-Supervised Learning

Semi-supervised learning (SSL) methods [van Engelen and Hoos, 2020; Zhu et al., 2009] exploit information provided by the unlabelled data, instead of relying only on the labelled data, to guide the learning process and enhance predictive accuracy. SSL algorithms are based on the cluster assumption, stating that similar examples tend to belong to the same group, and they are distinguished into *inductive* and *transductive*. The former learn a classification model, whereas the latter are solely concerned with obtaining label predictions for the given unlabelled data.

Self-training techniques [Triguero et al., 2015; Yarowsky, 1995] learn an inductive base classifier from the labelled data and then select confident predictions on the unlabelled data, as pseudo-labels, in order to iteratively re-train the base classifier. If probabilistic predictions are employed to select the pseudo-labels at each iteration, then self-training is similar to the EM algorithm [Dempster et al., 1977; Ghahramani and Jordan, 1993]. Unfortunately, most of these methods extend propositional learners to self-training and cannot be directly applied to logic-based formalisms. Although there are approaches based on decision trees [Leistner et al., 2009; Tanha et al., 2017], which may be used to extend their supervised counterparts for relational data [Blockeel and Raedt, 1998], they cannot learn Event Calculus theories. More importantly, self-training is an iterative

procedure, that may have convergence issues [Culp and Michailidis, 2008], and thus it is inappropriate for online learning.

Co-training or multi-view learning [Blum and Mitchell, 1998; Xu et al., 2013] is an extension of self-training to multiple classifiers, which are iteratively trained on the labelled data, adding their most confident predictions to the labelled dataset of the other classifiers. These methods assume that the training data can be separated into distinct views, namely disjoint feature sets that provide complementary, ideally conditionally independent information about each instance, while each view is sufficient to accurately predict each class. Although the conditional independence assumption can be relaxed [Abney, 2002; Balcan et al., 2004] and heuristics can be devised to automatically split the data into views [Du et al., 2011], there is no guarantee about their efficacy.

Similar approaches, inspired by multi-view learning, employ different classifiers in order to exploit predictive divergence [Goldman and Zhou, 2000; Wang and Zhou, 2007; Zhou and Goldman, 2004]. For instance, Zhou and Li [2005] proposed tri-training, three different classifiers are alternately trained. When two of them agree on their prediction for a specific example, that example is passed onto the third classifier along with its pseudo-label. Li and Guo [2011, 2012] applied tri-training to relational data by using three different ILP learning systems, namely Aleph [Srinivasan, 2003], kFOIL [Landwehr et al., 2006] and nFOIL [Landwehr et al., 2007]. The classifiers are initialised using the labelled data and background knowledge, and then they are refined by iterating over the unlabelled data. Co-forest [Deng and Guo, 2011; Li and Zhou, 2007] is an attempt to extend tri-training to more than three classifiers, that is, to random forests. The idea is to train each decision tree independently on the labelled data and then, in each subsequent iteration, each tree receives pseudo-labelled data based on the joint prediction of all other decision trees.

To that end, a number of semi-supervised boosting approaches have also been proposed [Bennett et al., 2002; d'Alché-Buc et al., 2001; Grandvalet et al., 2001] that employ pseudo-labels to train an ensemble of classifiers. These methods do not exploit any form of label confidence to decide which unlabelled points should be used in the next training iteration. Instead they rely on simple sampling techniques and thus they may suffer from misclassifications. Mallapragada et al. [2009] proposed a variation that makes use of the pairwise distances between labelled and unlabelled examples, similar to graph-based methods, in order to calculate the confidence of the pseudo-labels.

Most of the above wrapper-like approaches rely on intermediate steps and supervised base learners. However, there are other intrinsically semi-supervised

learning methods that naturally extend inductive supervised learners to include the unlabelled data into the objective function. Vapnik [1998] proposed an extension to support-vector machines that modifies the objective by minimising also the number of unlabelled data points that violate the margin, penalised based on their distance to the closest margin boundary. The main disadvantage is that the optimisation is non-convex and NP-hard to solve. Therefore most efforts have been focused on efficient approximations [Bie and Cristianini, 2003; Chapelle et al., 2008; Collobert et al., 2006; Joachims, 2003]. Although SVMs can be applied to logic programming [Muggleton et al., 2005], they cannot learn Event Calculus, since they are restricted to Horn logic.

There are several other approaches that directly incorporate the unlabelled data into their objective function, such as Gaussian processes [Lawrence and Jordan, 2004], density regularisation [Corduneanu and Jaakkola, 2003; Grandvalet and Bengio, 2004], and neural network perturbation-based methods [Bachman et al., 2014; Miyato et al., 2019; Park et al., 2018; Rasmus et al., 2015; Tarvainen and Valpola, 2017]. Although the idea of penalising the sensitivity of the model to small perturbations of the input data or the model parameters is interesting, it is not straightforward for relational learning.

Graph-based methods [Subramanya and Talukdar, 2014] are transductive algorithms, which, unlike inductive ones, do not produce a predictor, they only yield a set of predictions for the set of unlabelled data points. These methods, as discussed in Section 2.2, define a graph over all data points, both labelled and unlabelled, encoding their pairwise similarity using weighted edges. Then the graph is used to infer the labels of the unlabelled vertices. Blum and Chawla [2001] proposed a hard label assignment based on graph mincut. Since the mincut approach can easily lead to degenerate cuts, yielding solutions where almost all unlabelled data fall within the same graph component, normalisation techniques have also been proposed to overcome the issue [Blum et al., 2004; Joachims, 2003]. The graph mincut can be relaxed to produce probabilistic label assignments [Zhu et al., 2003] yielding an efficient closed-form solution. The latter is also strongly related to random walks [Azran, 2007; Szummer and Jaakkola, 2001; Wu et al., 2012] and self-training techniques [Haffari and Sarkar, 2007].

The probabilistic approaches have two drawbacks. First, since the true labels are fixed during the optimisation, it cannot handle label noise well. Second, in irregular graphs, the influence of vertices with high degree is relatively large. Zhou et al. [2003] proposed, instead of clamping the true labels, their deviation from their original values is penalised. Moreover, the penalty term for unlabelled data is regularised by the vertex degree to overcome the latter issue. Class

imbalance is also a common problem in SSL. Zhu et al. [2003] suggested to adjust the decision threshold such that predicted label proportions correspond to predefined label proportions. On the other hand, Wang et al. [2008] developed an optimisation scheme that mitigates the problem by altering the influence of labelled samples based on the label proportions. The same approach was considered by Wang et al. [2013a] from a graph max-cut perspective.

Even though SSL has been extensively studied in static offline environments [Dyer and Polikar, 2012], online SSL that operates on data streams remains an open challenge [Krempl et al., 2014]. To that end, only a couple of online graph-based SSL methods have been proposed to date. Delalleau et al. [2005] proposed an inductive algorithm that first performs label propagation on a given training set of labelled and unlabelled data. Subsequent unlabelled examples are labelled using the previously learned inductive function. However, subsequent unlabelled examples are not incorporated into the learned model, neither are the labelled ones potentially arriving on-stream. Huang et al. [2015] also follow an inductive approach to graph-based SSL, by updating the graph adjacency matrix incrementally. Nonetheless, in that method all incoming data is stored in memory, leading to a cost that grows linearly with the training data. Valko et al. [2010] designed a transductive technique that quantises the stream into a small number of clusters using an online k-center algorithm. In that method, the harmonic solution is computed upon the cluster centers. Finally, Wagner et al. [2018], as presented in Section 2.2.2, a graph synopsis of the stream is stored, to perform label propagation on the compressed graph, yielding a constant memory cost analogous to graph size.

2.6.3 Distances for Relational Data

Graph construction is one of the most important steps in graph-based methods and, the labelling solution is sensitive to the distance measure used to interconnect the examples and capture local similarities. Extensive experiments have been conducted on different graph construction methods [de Sousa et al., 2013; Jebara et al., 2009]. However, all of them consider propositional measures for quantifying edge similarities. In order to apply graph-based SSL to logical representations, distance functions suitable for first-order logic are required.

A substantial amount of work exists in the literature on distance-based methods for learning from relational data. These approaches mainly originated from instance-based learning (IBL) [Aha et al., 1991], which, like SSL, assumes that similar instances belong to the same classes (e.g. kNN). Bisson [1992a,b] proposed a similarity measure, based on the structural comparison of logical

atoms, to perform conceptual clustering. Also, the RIBL measure [Emde and Wettschereck, 1996] extended IBL to the relational case by using a modified version of the similarity measure proposed by Bisson [1992b], and a kNN classifier. The basic idea of RIBL is to measure the similarity of two objects based on the similarity of their attributes, as well as, the similarity of the objects that are related to them. Although these distance measures have been used with success in several applications [Bisson, 1992a; Kirsten and Wrobel, 1998, 2000], they are limited to function-free Horn logic, operating only over constants. Therefore, they require flattening of representations having non-constant terms, and thus cannot be easily applied to nested representations, such as the Event Calculus. Although RIBL has been improved to allow lists and function terms in the input representation by employing an edit distance [Bohnebeck et al., 1998; Horváth et al., 2001], still it is not sensitive to the depth of the structure.

As discussed in Section 2.3, Nienhuys-Cheng [1997] proposed a simple distance metric for comparing sets of logical atoms. However, the measure is not suited to first-order logic, where it is possible that two sets of atoms that differ only, but perhaps very strongly, in one atom are otherwise very similar. Ramon and Bruynooghe [1998] proposed a general framework for defining distance functions over sets of atoms using one-to-one mappings instead of the Haussdorf metric. Additionally, Mavroeidis and Flach [2003] define distances using the lattice structure of the first-order terms, whose metric space is isometrically embeddable in a vector space based on any Minkowski metric. However, in order to generate a common vector space for a set of clauses, all clauses should be available beforehand, which is not possible in online processing. Another similarity measure for Horn clauses was proposed by Ferilli et al. [2009], in order to tackle the problem of indeterminacy, which states that some atoms in one clause can be possibly mapped to several others onto another clause. However that similarity measure has increased computational complexity.

An alternative to the structural distances, are the semantic approaches [Sebag, 1997; Sebag and Schoenauer, 1993], that aim to compile a knowledge base into a similarity measure. The idea is to learn a set of rules from the given training data and then use them to map the structural representation of the examples into a vector space. The resulting numerical vectors capture the semantics between the input data and the target concept and can be compared using typical propositional measures, like the Euclidean distance. However, in an SSL task, the induction of these rules is not very reliable since there are only few labelled data available.

2.6.4 Feature Selection

Although, a number of measures exists for first-order logic, either structural or semantic, none of them essentially accounts for irrelevant or noisy features and thus their credibility may be compromised.

Numerous methods have been proposed in order cope with such problems, stemming either from feature selection [Chandrashekar and Sahin, 2014; Guyon et al., 2006] or metric learning [Kulis, 2013; Wang and Sun, 2015] techniques. Filter methods to feature selection are a popular candidate since they are fast to compute. Existing approaches are based on mutual information [Brown, 2009; Vergara and Estévez, 2014], consistency and constraint scores [Arauzo-Azofra et al., 2008; Benabdeslem and Hindawi, 2014; Zhang et al., 2008], and rough set theory [Modrzejewski, 1993; Pawlak et al., 1995]. However, these methods usually provide only a ranking of the features according to a quality criteria. Thus, the user should select a subset of them based on some hyperparameter, such as, the top k features or use their respective score as a weight in some underlying distance, which is not always possible.

Metric learning, on the other hand, is an approach that aims to learn a distance measure on the feature space in order for some given pairs of data points to be pulled as close as possible, while others be pushed far apart. Popular approaches are either supervised based on Mahalanobis distance learning [Goldberger et al., 2004], or unsupervised ones [Roweis and Saul, 2000; Tenenbaum et al., 2000] that achieve dimensionality reduction through linear reconstruction. The mass-based dissimilarity [Aryal et al., 2014], is also a form of unsupervised metric learning [see Ting et al., 2019, Section 8]. In contrast to supervised methods, it derives dissimilarity directly from data by estimating the probability mass of the region covering the given data points, without any class information. Only a few attempts exist in combining metric learning with graph-based semi-supervised learning. [Wang and Zhang, 2008] introduced the linear neighbourhood propagation (LNP) algorithm, a form of unsupervised metric learning, that constructs a graph such that any data point can be approximated as a linear combination of its neighbours. Then, Karasuyama and Mamitsuka [2013] proposed to regularise LNP using a Gaussian kernel and finding the coefficients that minimise the local reconstruction error between each pair of vertices. Supervised metric learning methods, based on Mahalanobis distances, have also been applied to graph-based SSL [Okada and Nishida, 2010; Pourdamghani et al., 2012]. However none of them combine supervised metric learning to mass-based dissimilarity, in order to exploit both labelled and unlabelled data.

2.7 Summary

In this chapter we presented the basics of the Event Calculus formalism and the OLED system for learning EC theories from annotated data streams. Then, we discussed an efficient method to graph-based SSL that infers the labels of the unlabelled data by computing their distance to their labelled counterparts, as well as, an online variant that retains a synopsis graph of the incoming data in order to guarantee high quality labelling. Since the distance measure is essential for such methods, we argued for the necessity of relational distances and presented a simple distance to compare sets of logical atoms, analysing also its shortcomings. Finally, we presented a metric learning technique optimised for kNN classification that can be adapted for feature weighting and a mass-based dissimilarity that estimates the distance between examples using the data distribution.

In addition, we discussed related work on online learning of CE rules, briefly pointed out their benefits and drawbacks, as well as, the fact that all existing approaches assume a fully supervised training sequence in order to operate. Then we highlighted propositional techniques for semi-supervised learning, such as, self-training and multi-view training, boosting and SVM extensions, and discussed their limitations regarding the learning of CE rules and online processing. We also reviewed related work on graph-based methods and variants that can operate on data streams. To that end, we pointed out the importance of graph construction and presented distance measures appropriate for logical representations, as well as, feature selection and metric learning techniques that are essential for producing quality measurements, robust to noisy and irrelevant features.

3 SPLICE: Semi-Supervised Learning for Complex Event Recognition

"Analogies, it is true, decide nothing, but they can make one feel more at home."
— Sigmund Freud

In this chapter we address the problem of effectively applying online structure learning in the presence of incomplete supervision. Towards that goal, we take advantage of the structural dependencies underlying a logic-based representation and exploit regularities in the relational data, in order to correlate given labelled instances to unlabelled ones and reason about their actual truth values. Structure learning methods attempt to discover multi-relational dependencies in the input data, by combining appropriate evidence predicates, that possibly explain the given supervision, that is, the labelled ground query atoms of interest. The underlying assumption is that the sets of ground evidence atoms that explain particular labelled query atoms are also contiguous to sets of ground evidence atoms that relate to unlabelled instances.

One promising approach to model such similarities for partially supervised data is to use graph-based techniques. As mentioned in Section 2.2, such methods attempt to formulate the task of semi-supervised learning as a cost minimisation problem and then find an optimal assignment of values for the unlabelled instances given a similarity measure. To that end, we adapt the approach of Zhu et al. [2003] to operate on logical interpretations [Blockeel et al., 1999] that arrive over time. The resulting system, SPLICE [Michelioudakis et al., 2019], is designed to infer the missing labels of the incoming examples and scales well to data volumes that batch graph-based methods cannot handle.

Figure 3.1 presents the components and procedure of SPLICE, using, for illustration purposes, the human activity recognition domain as formalised in the Event Calculus. In order to address the online processing requirement, we assume

that the training sequence arrives in micro-batches. At each step t of the online procedure, a training micro-batch $\mathcal{D}_t$ arrives containing a sequence of ground evidence atoms, e.g. two persons walking individually, their distance being less than 34 pixel positions and having the same orientation. Each micro-batch may be fully labelled, partially labelled, or contain no labels at all. Labelling is given in terms of the `HoldsAt` query atoms that essentially represent the CEs of interest (see Section 2.1). Unlabelled query atoms are prefixed by '?', and are filled in by imposing a closed-world assumption. For instance, in micro-batch $\mathcal{D}_t$ there is no labelling for time-point 150, while time-point 100 expresses a positive label for the `move` CE activity.

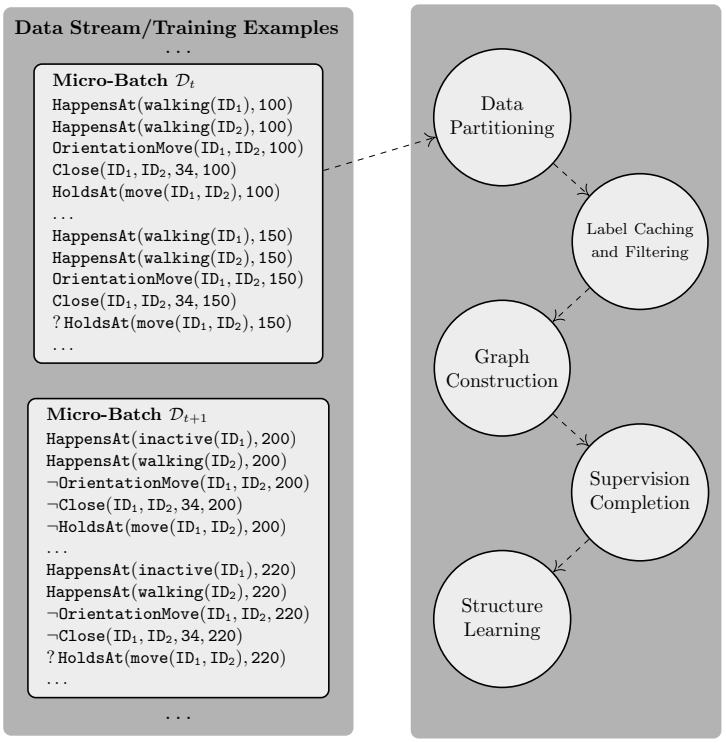

FIGURE 3.1: The Semi-Supervised Online Structure Learning (SPLICE) procedure.

In summary, each micro-batch $\mathcal{D}_t$ is passed onto the data partitioning component that groups the training sequence into examples. Each unique labelled example present in the micro-batch is stored in a cache, in order to be reused in subsequent micro-batches that may have missing labels. Labelled and unlabelled examples are converted into graph vertices, linked by edges that represent their structural similarity (see Section 2.3). The resulting graph is then used to label all unlabelled ground query atoms. Given the fully labelled training sequence, an online structure learning step refines or enhances the current hypothesis of CE rules — and the whole procedure is repeated for the next training micro-batch $\mathcal{D}_{t+1}$. For the online structure learning component, OLED may be used (see Section 2.1). The components of SPLICE are detailed in the following sections. To

aid the presentation, we use examples from activity recognition throughout the thesis.

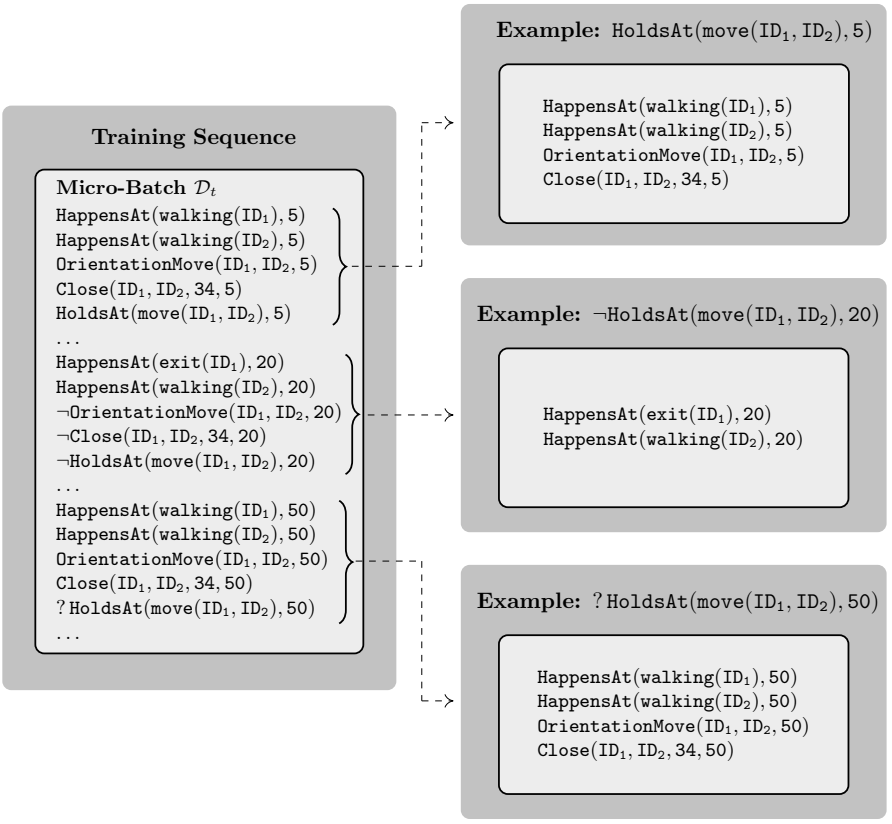

FIGURE 3.2: Data partitioning into examples. Each example contains a ground query atom, either labelled or unlabelled, as well as a set of true ground evidence atoms that relate to the query atom through their constants.

3.1 Data Partitioning

In a typical semi-supervised learning task, the training sequence consists of both labelled instances $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$ and unlabelled ones $\{\mathbf{x}_j\}_{j=l+1}^{u}$ where each label y_i corresponds to a D-dimensional feature vector $\mathbf{x}_i = (x_1, \ldots, x_D) \in \mathbb{R}_D$ of input values. Given a logic-based representation of instances, our approach begins by partitioning the given input data (micro-batch $\mathcal{D}$) into sets of ground evidence atoms, each one connected to a (labelled) ground query atom. The resulting sets are treated as training examples. Let $\mathcal{E} = \{e_1, \ldots, e_M\}$ be the set of all true evidence ground atoms and $\mathcal{Q} = \{q_1, \ldots, q_N\}$ the set of all ground query atoms of interest in micro-batch $\mathcal{D}$. Each example should contain exactly one ground query atom q_i and a proper subset $\mathcal{E}_i \subset \mathcal{E} : i = \{1, \ldots, N\}$ of evidence atoms corresponding to q_i. Given the sets $\mathcal{E}$ and $\mathcal{Q}$, we construct an example for each ground query atom in $\mathcal{Q}$, regardless of whether it is labelled or not. To do so, we

partition the evidence atoms in $\mathcal{E}$ into non-disjoint subsets, by grouping them over the constants they share directly to the ground query atom q_i of each example. A constant is shared if and only if it appears in both atoms and its position in the arguments of both atoms has the same type. Note that the position of a constant in some evidence atom e may differ from that in q_i. We refrained from including longer range dependencies, such as considering evidence atoms that can be reached through several shared constants, to favour run-time performance. However, such an extension is straightforward.

Figure 3.2 illustrates the presented procedure. As usual, `HoldsAt` express query atoms, while all other predicates express evidence atoms. Unlabelled query atoms are denoted by the prefix '?'. Data partitioning takes into account only true evidence atoms and concerns only a specific query predicate. Note that each resulting example has a set $\mathcal{E}_i \subset \mathcal{E}$ of evidence atoms that comprise only constants relevant to the query atom. For instance, the ground evidence atom $\texttt{Close}(\texttt{ID}_1, \texttt{ID}_2, 34, 5)$ appearing only in the top example, shares constants $\texttt{ID}_1, \texttt{ID}_2$ with query atoms of other examples too, but constant `5` is only relevant to the top example. Constant `34` does not appear in any query atom and thus can be ignored. Similarly, ground evidence atoms having constants that appear in many query atoms will appear in all corresponding examples. This is an expected and desirable behaviour, because such predicates indeed capture knowledge that may be important to many query atoms. For instance, consider a ground predicate $\texttt{Person}(\texttt{ID}_1)$ stating that $\texttt{ID}_1$ is a person. If such a predicate was included in the evidence of Figure 3.2, it would appear in every example. Moreover, during partitioning, SPLICE can ignore specific predicates according to a set of given mode declarations [Muggleton, 1995], using the recall number, i.e., if the recall number is zero the predicate is ignored.

Algorithm 1 PARTITION($\mathcal{D}$, $\mathcal{M}$)

Input: $\mathcal{D}$: a training micro-batch, $\mathcal{M}$: a set of mode declarations
Output: V: a set of vertices

Partition $\mathcal{D}$ into $\mathcal{Q}$ and $\mathcal{E}$.
$V = \emptyset$
\# $c_{q1}, \ldots, c_{qn}$ are constants
for all ground query atoms $q(c_{q1}, \ldots, c_{qn}) \in \mathcal{Q}$ **do**
    $\mathcal{E}_q = \emptyset$
    for all true ground evidence atoms $e(c_{e1}, \ldots, c_{em}) \in \mathcal{E} : \text{recall} > 0$ **do**
        $C_{e,q} = \{c \,|\, c \in e \wedge \text{type}(c, e) \in \text{Types}(e) \cap \text{Types}(q)\}$
        if $\forall\, c_i \in C_{e,q}\ \exists\, c_j \in q : c_i = c_j \wedge \text{type}(c_i, e) = \text{type}(c_j, q)$ **then**
            $\mathcal{E}_q = \mathcal{E}_q \cup e(c_{e1}, \ldots, c_{em})$
    $V = V \cup \{(q(c_{q1}, \ldots, c_{qn}),\ \mathcal{E}_q)\}$
return V

We henceforth refer to examples as vertices, since each example is represented by a vertex in the graph which is subsequently used to infer the missing labels. Algorithm 1 presents the pseudo-code for partitioning the input data into examples representing the graph vertices. The algorithm requires as input a training micro-batch $\mathcal{D}$ and a set of mode declarations $\mathcal{M}$, and produces a set of vertices V. At line 1 the micro-batch is partitioned into a set of ground query atoms $\mathcal{Q}$ and a set of ground evidence atoms $\mathcal{E}$. Then at line 3 the algorithm iterates over all ground query atoms and for each one it finds all true ground evidence atoms sharing constants of the same type. The set $C_{e,q}$ includes a constant c of an evidence atom e if and only if the position of c in e has type τ, and τ is also present in the query atom q. Then, e is added to the vertex of q if all constants of $C_{e,q}$ appear in q, and their positions on both e and q have the same type. Function $\text{type}(c, a)$, appearing in line 6, gives the type of the position of constant c in atom a, while $\text{Types}(a)$ gives all the types of a. Finally, for each pair of a ground query atom and its corresponding set of relevant ground evidence atoms, the algorithm creates a vertex and appends it to the vertex set. The algorithm yields a total runtime complexity of $\mathcal{O}(|\mathcal{Q}||\mathcal{E}|)$.

3.2 Label Caching

In order to handle real-life applications where labelled examples are infrequent SPLICE uses a caching mechanism, storing previously seen labelled examples for future usage. At each step of the online supervision completion procedure, SPLICE stores all unique labelled examples that are not present in the cache and then uses the cached examples to complete the missing labels. For each labelled vertex it creates a clause, using the label as the head, the true evidence atoms as the body, and replacing all constants with variables according to a set of given mode declarations [Muggleton, 1995]. For instance, the second vertex of Figure 3.2 can be converted to the following clause:

$$\begin{aligned} &\neg\texttt{HoldsAt}(\texttt{move}(id_1, id_2),\, t) \Leftarrow \\ &\quad \texttt{HappensAt}(\texttt{exit}(id_1),\, t) \wedge \texttt{HappensAt}(\texttt{walking}(id_2),\, t) \end{aligned} \tag{3.1}$$

For each such clause, SPLICE checks the cache for stored vertices that represent identical clauses and stores only the unique ones. These unique, cached vertices are then used as labelled examples in the graph construction process for supervision completion in the current and subsequent micro-batches.

In any learning task, noise, such as contradicting examples, is a potential risk that may compromise the accuracy of the learning procedure. In order to make SPLICE tolerant to noise, we employ the Hoeffding bound [Hoeffding, 1963a], a probabilistic estimator of the error of a model (true expected error), given its empirical error (observed error on a training subset) [Dhurandhar and Dobra, 2012]. Given a random variable $X \in [0,1]$ and an observed mean $\bar{X}$ of its values after N independent observations, the Hoeffding bound states that with probability $1-\delta$ the true mean μ_X of the variable lies in an interval $(\bar{X}-\varepsilon, \bar{X}+\varepsilon)$, where $\varepsilon = \sqrt{\ln(2/\delta)/2N}$. In other words, the true average can be approximated by the observed one with probability $1-\delta$ given an error margin ε.

In order to remove noisy examples, we detect contradictions in the cached labelled vertices, using the subset of training data that has been observed so far in the online process. To do so, we use an idea proposed by Domingos and Hulten [2000]. Let c be the clause of a cached vertex v and n_c the number of times that clause has appeared in the data so far. Recall that the clause of a cached vertex is lifted, i.e. all constants are replaced by variables. Thus lifted clauses may appear many times in the data. Similarly, let $\neg c$ be the opposite clause of c, that is, a clause having exactly the same body but a negated head, and $n_{\neg c}$ its counts. For instance the opposite clause of (3.1) is:

$$\texttt{HoldsAt}(\texttt{move}(id_1, id_2),\, t) \Leftarrow$$
$$\texttt{HappensAt}(\texttt{exit}(id_1),\, t) \wedge \texttt{HappensAt}(\texttt{walking}(id_2),\, t)$$

The goal is to eventually select only one of the two contradicting clauses. We define a function $p(c) = \frac{n_c}{n_c + n_{\neg c}}$ with range in $[0,1]$ that represents the probability of clause c to appear in the data instead of its opposite clause $\neg c$. Then according to the Hoeffding bound, for the true mean of the probability difference $\Delta p = p(c) - p(\neg c)$ it holds that $\Delta\bar{p} - \varepsilon < \Delta p$, with probability $1-\delta$. Hence, if $\Delta\bar{p} > \varepsilon$, we accept the hypothesis that c is indeed the best clause with probability $1-\delta$ and thus v is kept at this point. Similarly, $\neg c$ is the best one if $-\Delta\bar{p} > \varepsilon$. Therefore, in order to select between contradicting labelled examples, it suffices to accumulate observations until their probability difference exceeds ε. Until that point both example vertices are used in the optimisation.

Although we use the Hoeffding inequality to make filtering decisions for contradicting examples, given the data that we have seen so far, the examples are not independent as the Hoeffding bound requires. Consequently, we allow this filtering decision to change in the future, given the new examples that stream-in, by keeping frequency counts of the lifted examples. Furthermore, we assume

Algorithm 2 CACHEUPDATEANDFILTER(V_L, $\mathcal{C}$)

Input: V_L: a set of labelled vertices,
$\mathcal{C}$: a cache containing mappings of vertices to their counts
Output: V_L': a set of filtered labelled vertices, $\mathcal{C}$: the updated cache

1: **for** $v_i \in V_L$ **do**
2: **if** $\exists\, v_j \in \mathcal{C} : \text{canUnify}(\text{clause}(v_i), \text{clause}(v_j))$ **then**
3: $\mathcal{C}[v_j] = \mathcal{C}[v_j] + 1$
4: **else**
5: $\mathcal{C}[v_i] = 1$
6: Initialise accumulated unique filtered labelled nodes $V_L' = \emptyset$
7: **for** $(v_i,\, n) \in \mathcal{C}$ **do**
8: Generate clause $c = \text{clause}(v_i)$ and its opposite $\neg c$
9: **if** $\exists\, v_j \in \mathcal{C} : \text{clause}(v_j) = \neg c$ **then**
10: Compute total number of appearances $N = \mathcal{C}[v_i] + \mathcal{C}[v_j]$
11: Compute frequencies $p_c = \frac{\mathcal{C}[v_i]}{N}, p_{\neg c} = \frac{\mathcal{C}[v_j]}{N}$
12: Compute $\varepsilon = \sqrt{\frac{\ln(2/\delta)}{2N}}$
13: **if** $p_c - p_{\neg c} > \varepsilon$ **then**
14: $V_L' = V_L' \cup v_i$
15: **else**
16: $V_L' = V_L' \cup v_i$
17: **return** V_L', $\mathcal{C}$

that the examples stem from a stationary stochastic process and thus the difference between contradicting example frequencies eventually converges when a sufficient amount of observations is accumulated. This is not the case in other applications [Abdulsalam et al., 2011; Domingos and Hulten, 2000] in which the decision is permanent.

Algorithm 2 presents the pseudo-code for cache update and filtering. The algorithm requires as input the labelled vertices of the current micro-batch and the cached mappings of vertices to their counts, and produces as output the set of filtered labelled vertices and the updated cache. If the clause view of a vertex exists in the cache, then the counter of that vertex is incremented, otherwise the vertex is appended to the cache and its counter is set to 1 (see lines 1–5). For each vertex in the cache we produce its clause and check if the cache contains a vertex representing the opposite clause. In case the opposite clause exists, the Hoeffding bound is calculated, in order to decide whether one of them can be filtered out (see lines 6–16). Given an appropriate hash function for the cache data structure (good distribution of hash keys), all cache query operations take effectively constant time. Hence, the algorithm yields a total time complexity of $\mathcal{O}(|\mathcal{Q}_L|)$, where $|\mathcal{Q}_L|$ is the number of labelled query atoms in a micro-batch.

3.3 Graph Construction

Once the example vertices have been constructed, SPLICE assigns truth values to the unlabelled vertices, by exploiting the information provided by the labelled ones, as well as, the similarity to other unlabelled vertices. A weighted edge between a particular pair of vertices $v_i, v_j : i, j \in \{1, \ldots, N\}$ represents the structural similarity of the underlying ground evidence atom sets in the two vertices. Recall that the number of vertices is equal to the number of ground query atoms in $\mathcal{Q}$, that is N. Let w_{ij} be the edge weight, i.e., the structural similarity of v_i and v_j. If w_{ij} is large enough, then the truth values of the ground query atoms q_i, q_j are expected to be identical. Therefore, the similarity measure essentially controls the quality of the supervision completion solution.

Our approach regarding the computation of the evidence atom similarities is based on a measure of structural dissimilarity $d_s : \mathcal{E} \times \mathcal{E} \mapsto \mathbb{R}$, over a set of first-order expressions $\mathcal{E}$. The distance d_s does not make any syntactical assumptions about the expressions, such as function-free predicates, and thus it is applicable to any domain of interest. As described in Section 2.3, we apply the measure over sets of ground atoms using the Kuhn-Munkres algorithm, which provides an optimal one-to-one mapping given a cost matrix. In our case the cost matrix essentially holds the distances between each pair of ground atoms, computed by Eq. (2.9), present in the sets being compared. In particular, for each pair of vertices $v_i = (\mathcal{E}_i, q_i)$, $v_j = (\mathcal{E}_j, q_j)$ our approach begins by computing the distance between each pair of expressions $d(e_{im}, e_{jk}) : e_{im} \in \mathcal{E}_i, e_{jk} \in \mathcal{E}_j$ resulting in a matrix $\mathbf{C}$ that represents the costs of the assignment problem:

$$\mathbf{C} = \begin{pmatrix} d(e_{i,1}, e_{j,1}) & d(e_{i,1}, e_{j,2}) & \cdots & d(e_{i,1}, e_{j,M}) \\ d(e_{i,2}, e_{j,1}) & d(e_{i,2}, e_{j,2}) & \cdots & d(e_{i,2}, e_{j,M}) \\ \vdots & \vdots & \ddots & \vdots \\ d(e_{i,M}, e_{j,1}) & d(e_{i,M}, e_{j,2}) & \cdots & d(e_{i,M}, e_{j,M}) \end{pmatrix}$$

This matrix is square $M \times M$, assuming that the sets $\mathcal{E}_i$ and $\mathcal{E}_j$ are of equal size. In the general case, of a $M \times K$ matrix, where $M > K$, $\mathbf{C}$ is padded using zero values to complete the smaller dimension and be made square. Intuitively, the zero values in the smaller set capture the notion of unmatched atoms. Matrix $\mathbf{C}$ can then be used as the input cost matrix for the Kuhn-Munkres algorithm, in order to find the optimal mapping of evidence atoms. The optimal mapping is denoted here by the function $g : V \times V \mapsto \{(m, k) : m, k \in \{1, \ldots, K\}\}$ and is the one that minimises the total cost, i.e., the sum of the distances of the mappings. Finally, SPLICE computes the total distance between the vertices v_i, v_j

as the sum of the distances yielded by the optimal mapping normalised by the greater dimension, that is M, of the matrix:

$$d_s(v_i, v_j) = \frac{1}{M}\Big[(M - K) + \sum_{(m,k) \in g(v_i, v_j)} \mathbf{C}_{m,k}\Big] \tag{3.2}$$

The unmatched evidence atoms constitute an important component of the distance, due to the term $M - K$, which penalises every unmatched ground atom by the greatest possible distance, that is 1. Thus, $M - K$ can be seen as a regularisation term. The need to penalise unmatched atoms stems from the fact that they may represent important features that discriminate a positive from a negative example. The distance is turned into a similarity $s(v_i, v_j) = 1 - d_s(v_i, v_j)$ and assigned as the weight w_{ij} of the edge connecting the vertices v_i, v_j. The measure denoted by the function s is symmetric and is used to calculate the similarity of all distinct vertex pairs. The process generates a $N \times N$ symmetrical adjacency matrix $\mathbf{W}$ comprising the weights of all graph edges. Hence, matrix $\mathbf{W}$ is computed using Eq. (3.2) through function s. To avoid self-loops, i.e., edges that connect a vertex to itself, we set the diagonal of the $\mathbf{W}$ matrix to zero:

$$\mathbf{W} = \begin{pmatrix} 0 & s(v_1,\, v_2) & \cdots & s(v_1,\, v_N) \\ s(v_2,\, v_1) & 0 & \cdots & s(v_2,\, v_N) \\ \vdots & \vdots & \ddots & \vdots \\ s(v_N,\, v_1) & s(v_N,\, v_2) & \cdots & 0 \end{pmatrix}$$

In order to turn the similarity matrix $\mathbf{W}$ into a graph, we use a connection heuristic, which introduces edges only between vertices that are very similar, i.e., they have a high weight. In the simplest case, we connect the vertices v_i, v_j if $s(v_i,\, v_j) \geq \epsilon$, given some threshold value ϵ (ϵNN). Another alternative is to use k nearest neighbour (kNN) to choose the edges that will be kept. According to this approach, for each vertex v_i we identify the closest (most similar) k vertices. Note that if v_i is among v_j's k nearest neighbours, the reverse is not necessarily true. Therefore, as soon as kNN is applied, matrix $\mathbf{W}$ is no longer symmetric. In order to avoid tie-breaking, we modified kNN to select the top k distinct weights in a vertex neighbourhood, and then connect all neighbours having such a weight.

3.4 Supervision Completion

Given the weight matrix $\mathbf{W}$, we apply one of the two connection heuristics – thresholding and kNN – mentioned above to obtain a sparse matrix $\mathbf{W}'$, having

zeros for unconnected vertices and some positive similarity value $w_{ij} \in (0, 1]$ for the rest. Matrix $\mathbf{W}'$ is then used to solve a cost minimisation problem and assign truth values to the unlabelled ground query atoms.

Let $l + u = N$ be the number of labelled and unlabelled vertices. The closed-form solution of the optimisation problem for the harmonic function (see Section 2.2) in matrix notation is as follows. Let D_{ii} be the weighted degree of vertex i, i.e., the sum of the edge weights connected to i. Let $\mathbf{D}$ be a $N \times N$ diagonal matrix, containing D_{ii} on the diagonal, computed over the matrix $\mathbf{W}'$. Then the unnormalised graph Laplacian matrix $\mathbf{L}$ is defined as follows:

$$\mathbf{L} = \mathbf{D} - \mathbf{W}'$$

In this case, the Laplacian matrix essentially encodes the extent to which the harmonic function f (see Eq. (2.8)) differs at a vertex from the values of nearby vertices. Assuming that vertices are ordered so that the labelled ones are listed first, the Laplacian matrix can be partitioned into four sub-matrices as follows:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{ll} & \mathbf{L}_{lu} \\ \mathbf{L}_{ul} & \mathbf{L}_{uu} \end{bmatrix}$$

The partitioning is useful for visualising the parts of $\mathbf{L}$. Sub-matrices $\mathbf{L}_{ll}, \mathbf{L}_{lu}, \mathbf{L}_{ul}$ and $\mathbf{L}_{uu}$ comprise, respectively, the harmonic function differences between labelled vertices, labelled to unlabelled, unlabelled to labelled and unlabelled to unlabelled. Note that $\mathbf{L}_{lu}$ and $\mathbf{L}_{ul}$ are not symmetric if the kNN connection heuristic has been applied on $\mathbf{W}$.

Let $\mathbf{f} = \big(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{l+u})\big)^{\mathrm{T}}$ be the vector of f values of all vertices and the partitioning of $\mathbf{f}$ into $(\mathbf{f}_l, \mathbf{f}_u)$ holds the values of the labelled and unlabelled vertices respectively. Then by solving the constrained optimisation problem, expressed in Eq. (2.7), using Lagrange multipliers and matrix algebra, one can formulate the harmonic solution as follows:

$$\begin{aligned} \mathbf{f}_l &= \mathbf{y}_l \\ \mathbf{f}_u &= -\mathbf{L}_{uu}^{-1}\mathbf{L}_{ul}\mathbf{y}_l \end{aligned} \tag{3.3}$$

Note that Eq. (3.3) requires the computation of the inverse of matrix $\mathbf{L}_{uu}$ that may be singular, due to many zero values (sparsity). In order to avoid such situations, we compute the pseudo-inverse. Since the optimal solution is required to

comprise the labels assigned to unlabelled vertices in $[-1, 1]$, the resulting solution $\mathbf{f}_u$ is thresholded at zero to produce binary labels. Alternatively, an adaptive thresholding approach may be used to handle the possible class imbalance by exploiting the class prior probabilities, such as, the log-odds threshold approach proposed by Zhu et al. [2003], called *class mass normalisation*. However, in an online learning task, the class prior probabilities are usually unknown, and difficult to estimate from data, since there are few available labels, which in turn may yield much worse performance than the harmonic threshold.

Algorithm 3 SUPERVISIONCOMPLETION(V, h, s)

Input: V: a set of labelled and unlabelled vertices,
h: a connection heuristic, s: structural similarity
Output: $\mathbf{f}_u$: labels for the unlabelled query atoms

1: Initialise matrix $\mathbf{W}$ to be the zero matrix $\mathbf{0}$
2: **for** $v_i \in V_U$ **do**
3: **for** $v_j \in V$ **do**
4: $w_{ij} = s(v_i, v_j)$
5: Apply the connection heuristic: $\mathbf{W}' = h(\mathbf{W})$
6: Compute Laplacian matrix: $\mathbf{L} = \mathbf{D} - \mathbf{W}'$
7: Compute the harmonic solution: $\mathbf{f}_u = -\mathbf{L}_{uu}^{-1}\mathbf{L}_{ul}\mathbf{y}_l$
Perform thresholding to acquire binary labels
8: **for** $f_i \in \mathbf{f}_u$ **do**
9: **if** $f_i <$ small value **then** $f_i = -1$ which represents false
10: **else** $f_i = 1$ which represents true
11: **return** $\mathbf{f}_u$

Algorithm 3 presents the pseudo-code for connecting the graph vertices and performing supervision completion. The algorithm requires as input a connection heuristic, a structural similarity and a set of vertices, and produces as output a set of labels for the unlabelled vertices. First, we compute the similarity between pairs of vertices (see lines 1–4). Note that Eq. (3.3) only requires $\mathbf{L}_{uu}$ and $\mathbf{L}_{ul}$ and thus we only compute unlabelled-to-unlabelled and unlabelled-to-labelled connections. Then, we apply the connection heuristic to the matrix $\mathbf{W}$ holding the similarity values, compute the Laplacian matrix $\mathbf{L}$ and solve the optimisation problem (see lines 5–7). Finally, for the resulting vector $\mathbf{f}_u$ holding the values of the unlabelled vertices, we perform thresholding on each value, yielding binary labels (see lines 8-10). Since the unlabelled examples are typically many more than the labelled ones (in a micro-batch), the inversion of the Laplacian matrix, yielding time $|\mathcal{Q}_U|^3$, is the main overhead of the algorithm, where $|\mathcal{Q}_U|$ denotes the number of unlabelled ground query atoms in a micro-batch. Algorithm 4 presents the complete SPLICE procedure.

Algorithm 4 SPLICE(h, s, δ, $\mathcal{M}$)

Input: h: connection heuristic, s: structural similarity,
δ: Hoeffding bound confidence, $\mathcal{M}$: Mode declarations

Initialise cache containing list of vertices and their counts $\mathcal{C} = \emptyset$
for $t = 1$ to I micro-batches **do**
    Receive a micro-batch $\mathcal{D}_t = (\mathcal{Q}_t, \mathcal{E}_t)$
    # $\mathcal{Q}_t$ is a set of ground query atoms and $\mathcal{E}_t$ a set of ground evidence atoms.
    Partition data into vertices $V = \text{PARTITION}(\mathcal{D}_t, \mathcal{M})$
    Partition V into labelled V_L and unlabelled V_U vertices
    $V'_L, \mathcal{C} = \text{CACHEUPDATEANDFILTER}(V_L, \mathcal{C})$
    Union of the unique labelled nodes with unlabelled ones: V'=$V'_L \cup V_U$
    $\mathbf{f}_u = \text{SUPERVISIONCOMPLETION}(V', h, z)$
    Perform a structure learning step using $(\mathbf{f}_l, \mathbf{f}_u)$

4 SPLICE+: Semi-Supervised Learning Combining Structure and Mass-based Predicate Similarity

"Information is the resolution of uncertainty."
— Claude Shannon

In the previous chapter, we presented SPLICE, a technique that aims to effectively learn the structure of complex event rules in the presence of incomplete supervision. However, there are a couple of downsides, related to its graph construction process (see Figure 3.1), that may compromise the online labelling of the unlabelled data. First, the underlying structural distance may be deluded in the presence of irrelevant or noisy features. Second, the distance measurements between labelled and unlabelled data, inevitably, are as informative as the provided labels. For instance, if the given labels are not representative of the underlying class distribution, so are the measurements. Third, the online labelling inferred from the local graphs built per micro-batch, provides no guarantee with respect to the global solution obtained if all data where to be accessed at once.

In this chapter our goal is to improve the quality of the graph construction, as used by SPLICE, leading to a more robust and accurate labelling of the incoming unlabelled data. Figure 4.1, extending Figure 3.1, presents an overview of these improvements. We propose a hybrid distance measure composed of two elementary distances, that combined aim to eliminate the drawbacks of the structural distance alone. The former part of the distance measure is an enhanced version of Eq. (3.2) that accounts for irrelevant or noisy features by selecting only a subset of them, that is, the ones optimising kNN classification on the labelled data. Since such a feature selection is achieved using only the labelled data, the selected features may not always be representative of the underlying classes. Therefore, we

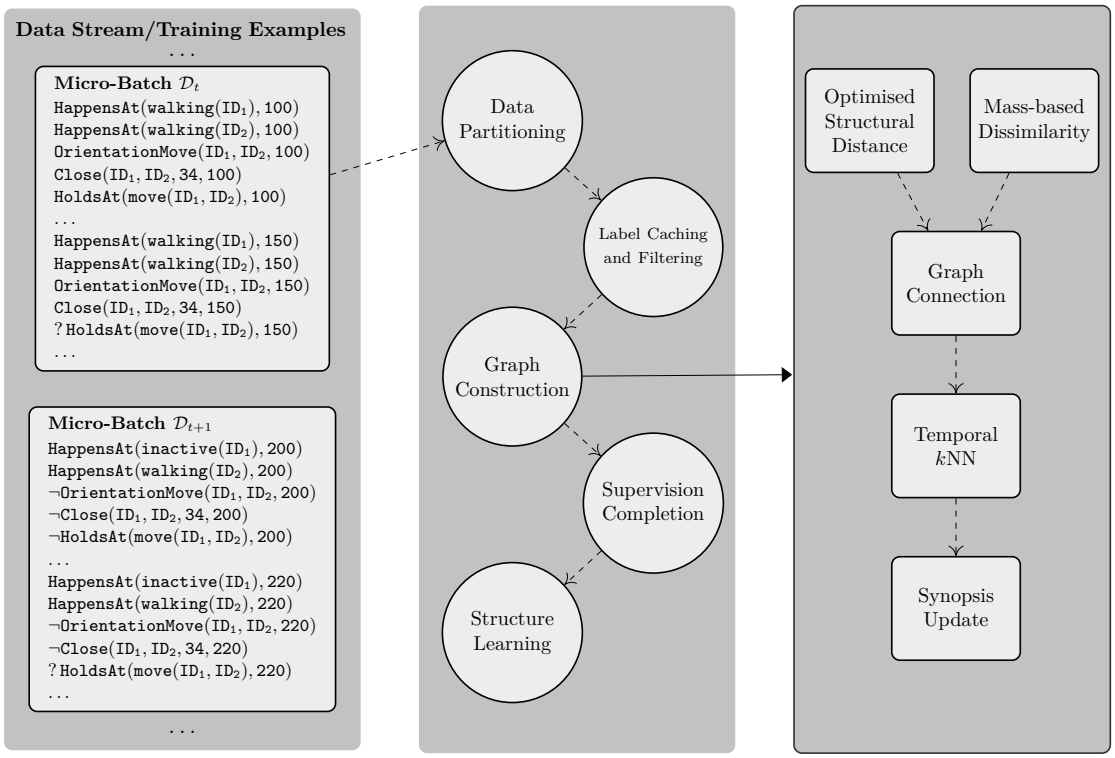

FIGURE 4.1: The SPLICE+ procedure.

combine the optimised structural distance with a data-driven mass-based dissimilarity, adapted for logical structures. The latter employs mass estimation theory to compute the relative distance between examples, which, intuitively, is measured as the probability density of their least general generalisation [Plotkin, 1971].

Moreover, in order to render SPLICE aware of the temporal nature of the data in online processing and CER, we further alter its strategy for interconnecting graph vertices. We connect each unlabelled vertex to its k-nearest labelled neighbours, as well as, the temporally preceding unlabelled vertex. Therefore, we promote interactions between temporally adjacent unlabelled vertices during label propagation. Finally, similar to Wagner et al. [2018], we store a synopsis of the full history of the stream, by means of a short-circuit operator, that preserves the effective distances of labelled and unlabelled example vertices to subsequent optimisations.

Henceforth, we refer to our enhanced approach as SPLICE+. The proposed improvements introduced by our method are detailed in the following subsections.

4.1 Large-Margin Feature Selection for Logical Predicates

In order to render the structural distance of Eq. (3.2) aware of irrelevant or noisy features, we introduce a mechanism for feature selection based on the ideas of LMNN metric learning. We adapt the idea of feature weighting, as presented in Section 2.4, by learning a binary vector, instead of real-valued one, that represents the set of selected logical atoms to be used for computing distances. Towards that goal, we use an approach similar to propositionalization [Alphonse and Matwin, 2002; Zucker and Ganascia, 1996]. More formally, let $\mathcal{A}$ be a set of first-order atoms that can be constructed from a Hebrand base $\mathcal{B}$ and a set of mode declarations $\mathcal{M}$, by replacing constants with variables. Assuming a strict ordering of atoms in $\mathcal{A}$, let $\mathbf{b}$ be a vector of binary variables, one of each first-order atom $a_i \in \mathcal{A}$. Thus, each indicator variable $b_i = 1$ if the ith atom is selected, and $b_i = 0$ otherwise. Since each labelled training example is essentially a clause c, it can also be seen as a binary vector $\mathbf{x}_c = [x_1, \ldots, x_{|\mathcal{A}|}]^{\mathrm{T}}$, where each variable x_i refers to the presence of the corresponding atom a_i from $\mathcal{A}$ in the clause represented by $\mathbf{x}_c$. For instance, assuming that $\mathcal{B}$ contains the ground atoms appearing in Figure 3.2, we can create an ordered set of atoms as follows:

$$\mathcal{A} = \big\{\texttt{HappensAt}(\texttt{walking}(x),\, t), \texttt{HappensAt}(\texttt{walking}(y),\, t), \texttt{HappensAt}(\texttt{exit}(x),\, t),\\ \texttt{OrientationMove}(x,\, y,\, t), \texttt{Close}(x,\, y,\, 34,\, t)\big\}$$

Then, the top example from Figure 3.2 is represented as $\mathbf{x}_{top} = [1, 1, 0, 1, 1]$, then middle one as $\mathbf{x}_{mid} = [0, 1, 1, 0, 0]$ and the bottom one as $\mathbf{x}_{bot} = [1, 1, 0, 1, 1]$. Thus, the distance between two such examples is essentially a Hamming distance, which is equivalent to the general Minkowski distance for $p = 1$. Since the Minkowski distance is a generalisation of the Euclidean distance, we reformulate the loss function of Eq. (2.16) as follows:

$$\varepsilon(\mathbf{b}) = (1-\mu) \sum_{j \in \mathcal{N}_i^k} \mathbf{b}|\mathbf{x}_i - \mathbf{x}_j| + \mu \sum_{i,j \in \mathcal{N}_i^k} \sum_{l} (1 - y_{il}) \big[1 + \mathbf{b}|\mathbf{x}_i - \mathbf{x}_j| - \mathbf{b}|\mathbf{x}_i - \mathbf{x}_l|\big]_+$$

where $\mathbf{x}$ is the clausal form of an example represented as a binary vector according to a predetermined strict ordering over $\mathcal{A}$, and $\mathbf{b}$ is the vector of indicator variables, denoting which features in $\mathbf{x}$ are selected. Moreover, we drop the first term of the loss function (corresponding to ε_{pull}), since it has been shown by Song et al. [2017] that the simpler problem often results in better solutions.

Moreover, the simpler loss function no longer depends on the parameter μ. The resulting minimisation problem is an integer linear programming problem and can be solved using variants of the branch-and-bound or branch-and-cut methods, albeit less efficiently than the real-valued problem[1]:

$$\begin{aligned} \textbf{minimise} \quad & \sum_{i,j \in \mathcal{N}_i^k} \sum_{l} (1 - y_{il}) \xi_{ijl} \\ \textbf{subject to} \quad & \textbf{(1)}\ \mathbf{b}|\mathbf{x}_i - \mathbf{x}_l| - \mathbf{b}|\mathbf{x}_i - \mathbf{x}_j| \geq 1 - \xi_{ijl} \\ & \textbf{(2)}\ \mathbf{b}\mathbf{x}_i \geq 1 \\ & \textbf{(3)}\ \xi_{ijl} \in \mathbb{N}^{\geq} \\ & \textbf{(4)}\ \mathbf{b} \in \{0,1\}^{|\mathcal{A}|} \end{aligned} \tag{4.1}$$

The intuition of our proposed feature subset selection, called Large-Margin Feature Selection (LMFS), is to keep the minimal set of logical atoms (features) that are necessary to discriminate the given set of labelled examples. Note that the slack variables that monitor the hinge loss are integers instead of real values since a hamming distance yields only integer differences. Moreover, we have added an extra constraint that forces all labelled examples to have at least one positive atom that is selected. This constraint is necessary to avoid extremely sparse solutions that remove many atoms yielding empty examples. As soon as the optimal vector $\mathbf{b}$ has been found, we can generalise all example vertices by removing irrelevant features, that is, features for which $b_i = 0$. Then, the structural distance can be computed as usual, by applying Eq. (3.2) over the generalised vertices:

$$d_s^{\mathbf{b}}(v_i, v_j) = d_s(v_i^{\mathbf{b}}, v_j^{\mathbf{b}}) \tag{4.2}$$

where v_i, v_j are vertices and $v_i^{\mathbf{b}}, v_j^{\mathbf{b}}$ their generalised counterparts, having some first-order atoms removed. One issue that may arise from selecting features using only the labelled examples is that some atoms that appear only in unlabelled examples are not considered during the optimisation. Regarding those atoms, that appear only in the unlabelled examples, we assume that they are always selected ($b = 1$) and use them in distance measurements.

LMNN requires that training examples are accompanied by some form of labelling. Then the optimisation above would retain the features that are necessary to discriminate between these labels. However, in a Hamming space distances change quite abruptly because a single mismatch between two binary vectors

[1] Note that in a semi-supervised problem the labelled examples are very few and sparse, leading to a very small number of constraints and thus Eq. 4.1 can be solved fast enough.

always yields a penalty of 1 between the vectors. In other words, while in an Euclidean space two points can be close or far in a specific dimension, according to their real-valued difference, in a Hamming space they are either the same or different in that dimension. Thus, clauses formed from training examples may appear very different inside the boundaries of a specific class, leading to very sparse solutions. This is because the optimisation would force them to become similar by removing atoms that cause mismatches. To avoid such situations, we perform clustering of the examples of each class and use the clusters as distinct classes to solve the optimisation problem.

Since we are interested in clustering the examples of each class into cohesive clusters, we cannot use a distance-based clustering, as it will suffer from the same noisy and irrelevant features that we aim to get rid of in the first place. To avoid that pitfall, we employ a clustering approach based on θ-subsumption. Examples in a cluster that are connected through a θ-subsumption relation and have the same label, define a taxonomic hierarchy containing all examples that are members of a specific concept. For instance, if two examples of the same class and length only differ in one atom, they cannot subsume each other and hence cannot be in the same cluster under θ-subsumption. Consider the top and middle examples of Figure 3.2. They should form different unit clusters, since they belong to opposite classes. If the bottom example was also positive, then it would belong to the same cluster as the top example since it θ-subsumes the top example. Thus, the resulting set of clusters represents a strict partitioning of the example space into distinct sub-concepts. Given such a clustering, the optimisation of Eq. (4.1) should select features that respect that partitioning, identifying which features are necessary for discriminating each sub-concept.

Algorithm 5 FEATURESELECTION(V_L, $\mathcal{M}$)

Input: V_L: a set of labelled example vertices, $\mathcal{M}$: a set of mode declarations
Output: $\mathbf{b}$: a vector of binary values corresponding to selected features

1: Partition V_L into positive V_P and negative V_N vertices.
2: Find $v_i^{\mathtt{max}} = \text{argmax}_{v_i \in V_P} |v_i|$ and $v_j^{\mathtt{max}} = \text{argmax}_{v_j \in V_N} |v_j|$.
3: Form unit clusters $C = \{\{v_i^{\mathtt{max}}\}, \{v_j^{\mathtt{max}}\}\}$.
4: **for** $v_i \in V_{L/v_i^{\mathtt{max}}, v_j^{\mathtt{max}}}$ **do**
5: **for** $c \in C$ **do**
6: **if** $\exists\, v' \in c : clause(v_i)\theta \subseteq clause(v')$ **then**
7: $c = c \bigcup v_i$
8: Solve optimisation of Eq. (4.1) using C as a set of examples.
9: **return** $\mathbf{b}$

Algorithm 5 presents the pseudo-code for selecting the first-order predicates that best discriminate the known labelled vertices into sub-concepts. The algorithm requires as input a set of labelled examples, a set of mode declarations, and

produces a vector of selected features. It starts by partitioning the given examples into positive and negative ones (line 1). Then for each of the two sets it finds the example having the most evidence atoms (ties are broken randomly) and creates unit clusters (lines 2–3). For each of the remaining examples it either appends it to an existing cluster, if another example exists that is θ-subsumed by the candidate, or it creates a new unit cluster (lines 4–7). Finally, it solves the optimisation problem of Eq. (4.1) using the clusters as distinct classes and returns the vector of selected features (lines 8–9).

4.2 Mass Dissimilarity for Logical Predicates

Supervised learning approaches to feature selection require explicit or implicit computation of the information/importance of each feature using the labels available in the training examples. However, in a semi-supervised learning task, the few labels that are often available are not sufficient for acquiring trustworthy estimation of the feature importance. Thus, common feature selection criteria are not reliable and their optimality guarantees suffer from the fact that only a few training examples are available.

In order to address the issue, we combine the optimised structural distance, as presented in Section 2.6.4, with a data-driven dissimilarity that uses mass estimation to measure the distance between data points. The intuition of the measure is that two points are considered to be more similar if they coexist in a sparse space rather than in a dense one. Unlike the distance estimation presented in Section 4.1, the proposed approach exploits both labelled and unlabelled data to quantify the distances between examples.

To that end, we adapt the approach presented in Section 2.5 to handle logical structures by means of the Herbrand base $\mathcal{B}$ and a set of mode declarations $\mathcal{M}$, the combination of which generates a set of logical atoms $\mathcal{A}$ (see Section 4.1). Since the space of these logical atoms is a hypercube $\{0,1\}^{|\mathcal{A}|}$, we can define a hierarchical partitioning H of the hypercube by randomly constructing a Half-Space Tree[2] [Ting et al., 2013]. In contrast to the approach of Ting et al. [2019], which assumes real-valued features, we can construct the trees beforehand because each internal node of the tree may only have one possible split, since all logical atoms are binary by definition. Algorithm 6 presents the pseudo-code for creating a forest of such binary random trees.

[2]Half-Space Trees are just like Isolation Trees, but instead of random splits they perform median splits. Thus, in a Hamming Space they are equivalent. However, we chose to use the former term since it implies a half-space partitioning which is more appropriate for binary features.

Algorithm 6 CREATEFOREST($\mathcal{A}$, T, h)

Input: $\mathcal{A}$: a set of first-order atoms, T: number of trees, h: maximum tree height
Output: $\mathcal{F}$: a set of Half-Space Trees

```
1: F = ∅
2: for i = 1 to T do
3:     F = F ⋃ CREATETREE(A, 0, h)
4: return F
5:
6: function CREATETREE(A, d, h)                    ▷ d is the current depth of the tree
7:     if d > h ∨ |A| < 1 then
8:         return Node(size ← 0, split ← ∅, left ← ∅, right ← ∅)
9:     else
10:        Randomly select an atom a ∈ A.
11:        return Node(size ← 0, split ← a, left ← CREATETREE(A/a, d+1, h),
12:                    right ← CREATETREE(A/a, d+1, h))
```

The algorithm requires as input a set of first-order atoms, the required number of trees, and a maximum height for each tree. We start from an empty set and iteratively generate random trees (see lines 1–4). Each node in the tree consists of a split atom, a left and right subtree, as well as, a size variable that stores the number of examples that have matched the path to this node. Each tree is built recursively by picking an atom at random from the given set of available atoms and creating two random subtrees on the remaining atoms (lines 9–11). The process terminates if no atoms are left in the set $\mathcal{A}$ or the maximum height is reached.

Algorithm 7 UPDATEFOREST($\mathcal{F}$, V)

Input: $\mathcal{F}$: a set of Half-Space trees, V: a set of example vertices

```
1: for tree ∈ F ∧ v ∈ V do
2:     UPDATESIZE(tree, v)
3:
4: function UPDATESIZE(tree, v)
5:     tree.size ← tree.size + 1
6:     if tree.left ≠ ∅ ∧ tree.split ∉ v then UPDATESIZE(tree.left, v)
7:     else if tree.right ≠ ∅ ∧ tree.split ∈ v then UPDATESIZE(tree.right, v/tree.split)
```

Note that during tree creation, each internal node of each tree has zero size. Tree creation happens before any data are processed. The size of the nodes is updated as more data stream in. Algorithm 7 describes this update process. The algorithm requires as input a forest of binary random trees and a set of examples. For each example it updates the counts of the internal nodes of each tree (lines 1–2). The update procedure is a recursive process that increments the size of the current node and then proceeds to the update of the child node that matches the split criterion of the current node (lines 5–7). Since each example is a set of atoms

the split criterion match is checked by the membership of the split atom. Thus, the path from the root to the leaf that contains the matched atoms of the given example increment the counts of its nodes.

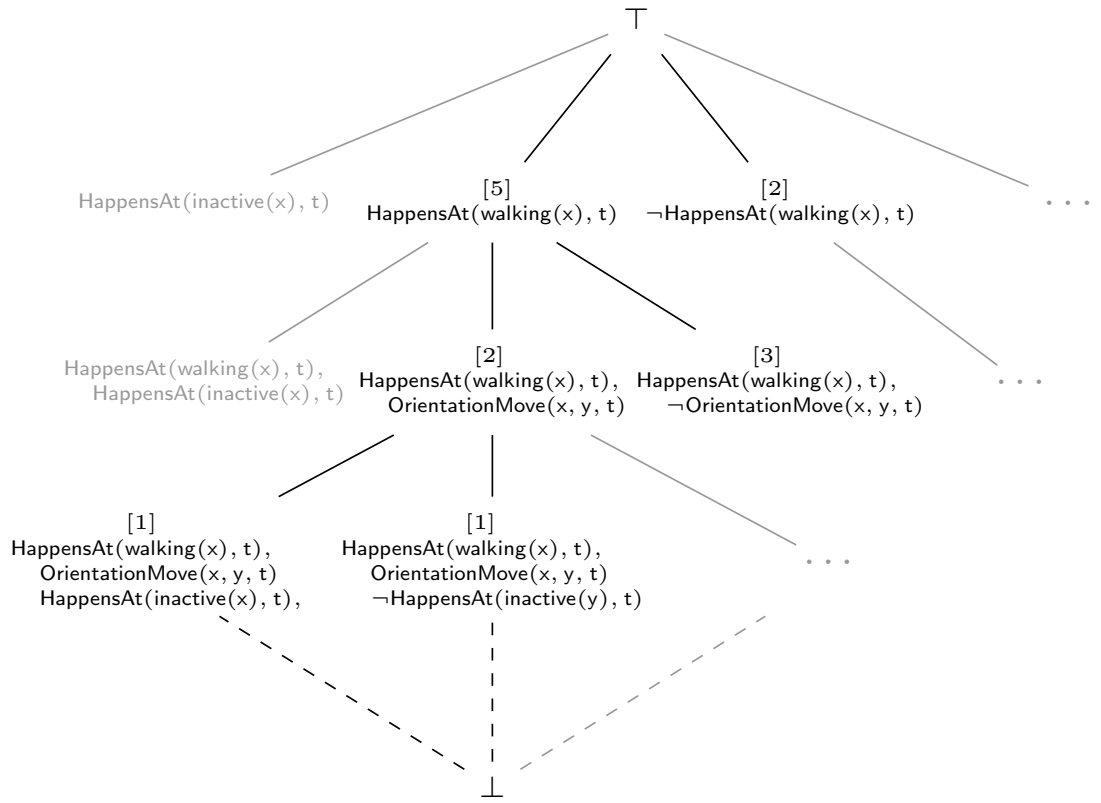

FIGURE 4.2: Path selected by a random tree from the subsumption lattice.

The intuition behind this relational version of Half-Space Trees is that we estimate the mass of specific areas of the subsumption lattice generated from a given Herbrand base $\mathcal{B}$ and constrained by the mode declarations $\mathcal{M}$. Figure 4.2 depicts a part of the subsumption lattice constructed from the atoms appearing in the training sequence of Figure 4.1. The highlighted part of the lattice presents a possible Half-Space Tree constructed by selecting one split atom per level, while the numbers represent the node sizes. In this case, the sizes correspond to the three examples of Figure 3.2. Therefore, each tree essentially represents only a part of the lattice and estimates the mass of each node from data. Given two examples, their overlap (set of common atoms) is quantified as the size of the deepest node in the tree that contains all common atoms along its path from the root. If the size is small, then these two examples are located in a sparse part of the space and thus they are considered more similar. Consider for instance the top and middle examples of Figure 3.2. Their set of common atoms is just the atom $\texttt{HappensAt}(\texttt{walking}(x), t)$, which, in the tree appearing in Figure 4.2, is located in the first level of the tree and has size 3. Therefore, these examples co-exist in a dense part of the tree and they may be considered less similar.

The resulting Half-Space Forest can be used to compute the mass-based dissimilarity of Eq. (2.19) for a pair of examples as follows:

$$\tilde{m}(v_i, v_j) = \frac{1}{T} \sum_{p=1}^{T} \frac{|R(v_i, v_j | H_p)|}{|D|}$$

where v_i, v_j are two examples, H_p is a binary Half-Space Tree (out of T), D is the set of all examples used to update the trees and R, similar to Section 2.5, is the deepest region covering both examples.

4.3 Robust Graph Construction and Labelling

Given a set of examples, our goal is to connect them by edges representing the similarity of the underlying evidence atom sets. The resulting graph is used to derive labels for all unlabelled example vertices in the current data micro-batch. In order to construct the similarity graph for label propagation we combine the mass-based dissimilarity, as presented in Section 4.2, with the optimised structural distance of Eq. (4.2) as follows:

$$d_h^{\mathbf{b}}(v_i, v_j) = \alpha \; d_s^{\mathbf{b}}(v_i, v_j) + (1 - \alpha) \; \tilde{m}(v_i, v_j)$$

where α controls the relative importance of each of the two distances. Similar to SPLICE, the hybrid distance is turned into a similarity as $1 - d_h^{\mathbf{b}}(v_i, v_j)$. Fully connecting the vertices generates a $N \times N$ symmetrical adjacency matrix $\mathbf{W}$, comprising the weights of all graph edges. In order to make the graph sparser, we aim to select the stronger edges in each neighbourhood. To that end, SPLICE+ uses a temporal variant of kNN that connects each unlabelled vertex to its k-nearest (most similar) labelled neighbours, as well as to its temporally adjacent ones. The intuition behind this extension of kNN is that temporally adjacent vertices should affect the labelling of each other, even if they are not very similar. In terms of label propagation, temporally adjacent neighbours should exchange information about their respective labelling, albeit weighted by their similarity.

Moreover, in order to obtain guarantees for the online labelling achieved by label propagation on the local graphs built from the micro-batches, SPLICE+ stores a synopsis of the graph, as presented in Section 2.2.2. Given a memory size parameter τ, the synopsis removes older vertices from the graph (when memory size is exceeded), in order to make room for newer ones, by meshing their edges

to the rest of the graph using star-mesh transforms. The harmonic solution computed on the compressed graph is guaranteed to be equal to the one computed on the entire stream seen so far. Therefore, the synopsis renders the labelling invariant to different batch sizes. Algorithm 8 presents the graph construction pseudo-code.

Algorithm 8 GRAPHCONSTRUCTION($\mathcal{F}$, V)

Input: $\mathcal{F}$: a set of Half-Space Trees, V: a set of example vertices
1: Partition example vertices into labelled and unlabelled $V = (V_L, V_U)$
2: UPDATEFOREST($\mathcal{F}, V_L^t \cup V_U^t$)
3: **if** $V_L^t \neq \emptyset$ **then**
4: $\quad \mathbf{b} =$ FEATURESELECTION($V_L, \mathcal{M}$)
5: $V_\tau \leftarrow V_L \cup V_U / V_U^t$
6: **for** $v_i \in \mathcal{V}_\tau$ **do**
7: $\quad$ **for** $v_j \in V_U^t$ **do**
8: $\quad\quad w_{v_i,v_j} \leftarrow 1 - d_h^{\mathbf{b}}(v_i, v_j)$
9: $V_\tau \leftarrow V_\tau \cup V_U^t$
10: Apply the connection heuristic: $\mathbf{W}_h =$ temporal-kNN($\mathbf{W}$)
11: **while** $|V_\tau| > \tau + |V_L|$ **do**
12: $\quad$ Find oldest vertex $v_o \leftarrow V_\tau / V_L$.
13: $\quad$ **for** all vertex pairs $v \neq v'$ in V_τ **do**
14: $\quad\quad w_{v,v'} \leftarrow w_{v,v'} + \frac{w_{v_o,v} w_{v_o,v'}}{\text{degree}(v_o)}$
15: $\quad$ Remove all v_o edges from $\mathbf{W}_h$.
16: **return** $\mathbf{W}_h$

The algorithm requires as input a pre-built Half-Space Forest, and a set of examples. The examples are partitioned into labelled and unlabelled at line 1. Then only the examples received in the current micro-batch t (labelled V_L^t and unlabelled V_U^t) are used for updating the forest counts at line 2. Subsequently, if the micro-batch t contains only unlabelled examples and labels have been added in V_L since the last time LMFS was run, the optimal set of features is re-computed (lines 3–4). In lines 5–9 the graph connection process takes place. Each stored example is connected to the unlabelled examples received at micro-batch t. The set V_τ of stored examples is composed of all the labelled examples V_L and the τ stored unlabelled ones, $V_U \setminus V_U^t$, where τ is the synopsis size. Then, the temporal-kNN connection heuristic is applied at line 10 to make the graph sparser. As a final step, while the number of stored unlabelled examples is greater than the given memory size τ, the algorithm removes the oldest example together with its edges and applies a star-mesh transform to its neighbours (lines 11–15).

5 Experimental Study

"It doesn't matter how beautiful your theory is, if it doesn't agree with experiment, it's wrong."
— Richard P. Feynman

In this chapter, we present experimental results for both SPLICE and SPLICE$^+$ on the task of Complex Event Recognition (CER), using OLED [Katzouris et al., 2016] for learning complex event rules in the Event Calculus. For the evaluation, we use a publicly available benchmark activity recognition dataset of the CAVIAR project[1], a publicly available maritime surveillance dataset concerning the area of Brest, France[2], and a fleet management dataset, recording the activity of vehicles around Greece and neighbouring countries[3]. Part of our evaluation also compares SPLICE and SPLICE$^+$ to simple kNN and a baseline that runs Iterative Cross-Training [Soonthornphisaj and Kijsirikul, 2004] combining ILASP [Law et al., 2016, 2018], a state-of-the-art ILP system for learning Event Calculus theories and Naive Bayes (ILASP-NB). All experiments were conducted on a Linux machine having an Intel i7 4790@3.6GHz CPU (4 cores, 8 threads) and 16GiB of RAM. All presented experiments can be reproduced, following the provided instructions[4].

5.1 Description of Datasets

The activity recognition dataset comprises 28 surveillance videos, where each video frame is annotated by human experts on two levels. The first level contains SDEs (simple, derived events) that concern instantaneous activities of individual persons, detected on video frames, such as when a person is walking or staying

[1] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1
[2] https://zenodo.org/record/1167595
[3] https://www.vodafoneinnovus.com
[4] https://users.iit.demokritos.gr/~vagmcs/pub/splice_plus

inactive. In addition, the coordinates of tracked persons are used to capture qualitative spatial relations, e.g. two persons being relatively close to each other. The second level contains CEs (composite events), describing the activities between multiple persons and/or objects, i.e., people meeting and moving together, leaving an object and fighting. Similar to earlier work [Katzouris et al., 2016; Skarlatidis et al., 2015c], we focus on the `meet` and `move` CEs, and from the 28 videos, we extract 19 sequences that contain annotations for these CEs. The rest of the sequences in the dataset are ignored, as they do not contain positive examples of these two target CEs. Out of the 19 sequences, 8 are annotated with both `meet` and `move` activities, 9 are annotated only with `move` and 2 only with `meet`. The total length of the extracted sequences is 12,869 video frames. Each frame is annotated with the (non-)occurrence of a CE and is considered an example instance. The dataset contains a total of 63,147 SDEs and 12,869 annotated CE instances. Out of those, there are 6,272 example instances of `move` and 3,722 instances of `meet`. Consequently, for both CEs the number of negatives is significantly larger than the number of positives.

The maritime dataset consists of vessel position signals sailing the Atlantic Ocean, around Brest, France. The SDEs take the form of compressed trajectories, comprising "critical points", such as *communication gap* (a vessel stops transmitting position signals), *vessel speed change*, and *turn*. It has been shown that compressing vessel trajectories allows for accurate trajectory reconstruction, while at the same time improving stream reasoning times significantly [Patroumpas et al., 2017]. We focus on the `rendezVous` and `pilotOps` CEs. The former expresses a potentially illegal activity where two vessels are moving slowly in the open sea and are close to each other, possibly exchanging commodities, while the latter describes the activity of piloting a vessel. Since the dataset is unlabelled, we produced synthetic annotation using the RTEC engine [Artikis et al., 2015] and hand-crafted rules of `rendezVous` and `pilotOps` CEs [Pitsikalis et al., 2019]. The CE annotation is publicly available[5]. We have extracted 6 sequences for each CE from the dataset. Regarding `rendezVous`, the total length of the sequences is 11,930 timestamps, while for `pilotOps`, sequences comprise 6,678 timestamps. There are 1,425 instances in which `rendezVous` occurs and 769 in which `pilotOps` occurs.

The fleet management dataset consists of commercial vehicles moving around Greece and neighbouring countries. The SDEs include information, such as vehicle speed changes, proximity to points of interest and operational status, such as fuel level. We focus on the `nonEconomicDriving` and `dangerousDriving` CEs, since they are the more complex ones. The former expresses a driving activity

[5] https://zenodo.org/record/2557290

where the driver is over-speeding despite having limited fuel, while the latter describes various dangerous driving behaviours, that is, over-speeding on ice, abrupt acceleration or braking and cornering other vehicles. Since the dataset is unlabelled, we produced synthetic annotation similar to the maritime dataset. We extracted 10 and 12 sequences for `nonEconomicDriving` and `dangerousDriving` respectively. The `nonEconomicDriving` CE, comprises $13{,}255$ timestamps, while `dangerousDriving` comprises $13{,}387$ timestamps. There are $1{,}589$ instances in which `nonEconomicDriving` occurs and 639 in which `dangerousDriving` occurs.

5.2 Experimental Setup

The evaluation concerns two learning scenarios. In the first scenario, a number of micro-batches were selected uniformly at random and their labels were hidden from the learner. We experimented retaining 5%, 10%, 20%, 40% and 80% of the micro-batches labelled. The micro-batches were selected using stratified sampling in order to retain the original class proportions in each sample. We repeated the random selection 20 times, leading to 20 runs per supervision level, in order to obtain a good estimate of the performance.

However, in a typical stream learning situation, the assumption of labels arriving randomly on-stream is unrealistic. A more appropriate assumption is that a fully labelled training set appears at the beginning of the stream, or stored in a database as historical data, while the rest of the data stream-in completely unsupervised. Moreover, in contrast to the random selection scenario, labels sampled only from a specific time frame are less representative of the actual distribution of the underlying classes, which makes the problem more challenging.

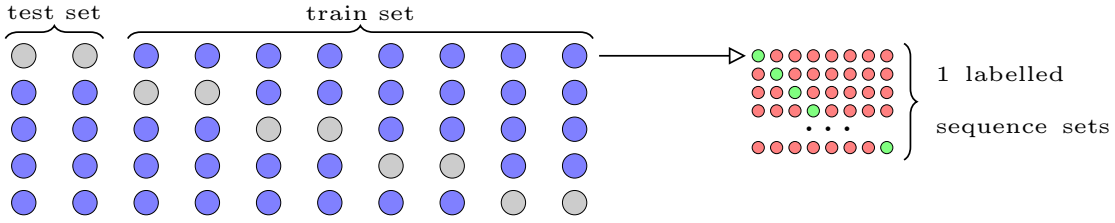

FIGURE 5.1: Process of creating labelled sequences from train sets. Purple and gray dots represent train and test sequences respectively. Green dots represent fully labelled sequences, while red dots represent completely unlabelled ones.

Our second evaluation scenario simulates this more realistic setting, using 1, 2 and 4 labelled training sequences (out of 6) for the maritime dataset, and 1, 2, 4 and 8 (out of 19) for the CAVIAR and the fleet management dataset. We considered only sequences that contain both positive and negative examples and generated 5 test sets. For each test set we used the remaining sequences for

creating the train sets. More precisely, each of the remaining sequences is used to generate multiple train sets containing a number of labelled sequences appearing at the beginning of the train set, while the rest of the train set remains completely unlabelled. For instance a 1-*labelled sequence set* contains one sequence that is fully labelled and appears first in the train set, while every other sequence in the set remains completely unlabelled. In order to avoid the selection bias, we exhaustively generated all possible 1-labelled sequence sets for each test set, while for 2, 4 and 8 we randomly selected some candidate sets. This process led to 40 runs for the `meet` and `nonEconomicDriving` CEs, 72 for the `move` CE, 60 for the `dangerousDriving` CE, and 30 for the `pilotOps` and `rendezVous` CEs. Since each sequence contains different proportions of positive and negative examples, the runs were not stratified.

Throughout the experimental analysis, we used $T{=}100$ random trees for computing the mass-based dissimilarity and set $\alpha{=}0.5$ in order to balance the influence of each distance measure. The accuracy results for both supervision completion and structure learning were obtained using the F_1-score. All reported statistics are micro-averaged over the recognised instances of CEs. For the CAVIAR dataset, the reported statistics for structure learning were collected using 10-fold cross validation over the 19 video sequences, while complete videos were left out for testing. The same number of folds were also used for the fleet management dataset. In the maritime dataset, the statistics were collected using 6-fold cross validation over the selected sequences, while complete sequences were left out for testing.

5.3 Evaluation on Activity Recognition

First, we compare the performance of SPLICE⁺, SPLICE and ILASP-NB on the activity recognition dataset for both `meet` and `move` CEs. Figure 5.2 depicts the F_1-score achieved by the supervision completion on both scenarios, without any structure learning. The results suggest that SPLICE⁺ effectively infers the missing labels and its performance increases as more supervision is given. More importantly, it significantly outperforms SPLICE and ILASP-NB in most cases even for high supervision levels (80% uniform supervision or 8 sequences). As expected, the difference is greater in the more realistic scenario, where labelled data are provided only at the beginning of the training sequence. ILASP-NB achieves comparable accuracy to SPLICE⁺ on the random supervision scenario for low supervision levels. However, it is worth noting that ILASP-NB is a batch learning system that requires all data to be available at once and may require multiple

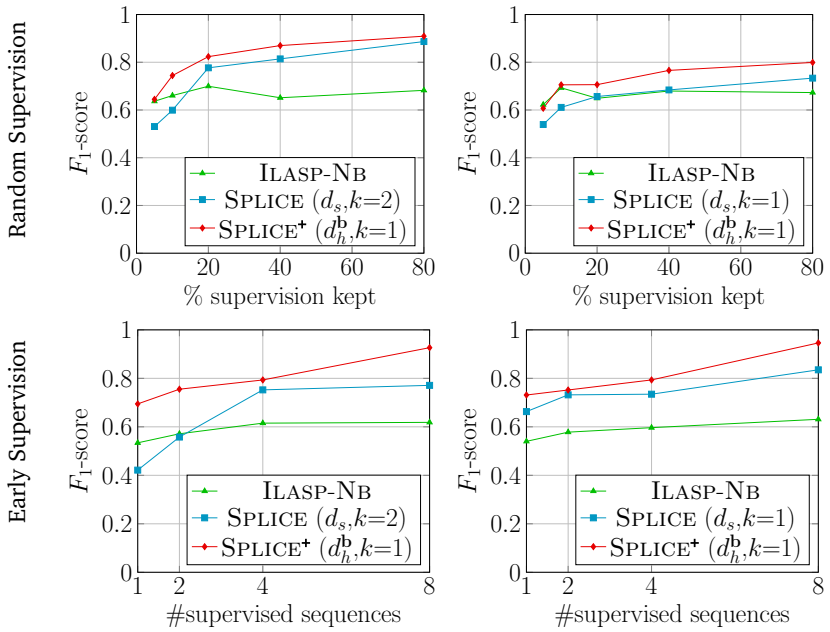

FIGURE 5.2: F_1-score of supervision completion on `meet` (left) and `move` (right) as supervision increases. In the first scenario, supervision arrives uniformly at random (top), while in the second one is provided only at the beginning of the sequence (bottom). The notation d_s and d_h^b refers to the structural and hybrid distances respectively.

iterations to converge. As a result, ILASP-NB was about 12 times slower than SPLICE+ in the activity recognition dataset.

On the other hand, the improved performance in SPLICE+ comes at the cost of a decrease in runtime compared to SPLICE, i.e., the time to process both the train and test set, as shown in Figure 5.3. Note that SPLICE+ is always slower than SPLICE, since it has to update the trees for every micro-batch and select appropriate features when required. However the penalty is tolerable in absolute times, as it does not exceed 15 seconds. Note that runtime tends to increase between 5% and 20% of random supervision, and then falls again, as more supervision is given. This is due to fact that SPLICE+, for efficiency reasons, performs feature selection only when a labelled micro-batch is followed by an unlabelled one (see Algorithm 8). In the absence of unlabelled micro-batches, we simply store the incoming labels and perform the costly optimisation task only when necessary, i.e., in the presence of unlabelled data. In the cases of 20% or 40% supervision, this situation occurs much more frequently than when 5% or 80% supervision is provided. A similar pattern is observed in the early supervision setting, where feature selection only runs once, since all labelled examples arrive at the beginning of training.

Figure 5.4 presents the structure learning results using the OLED system for constructing CE rules. We compare OLED using SPLICE for supervision completion against OLED using SPLICE+, and OLED alone without any supervision completion.

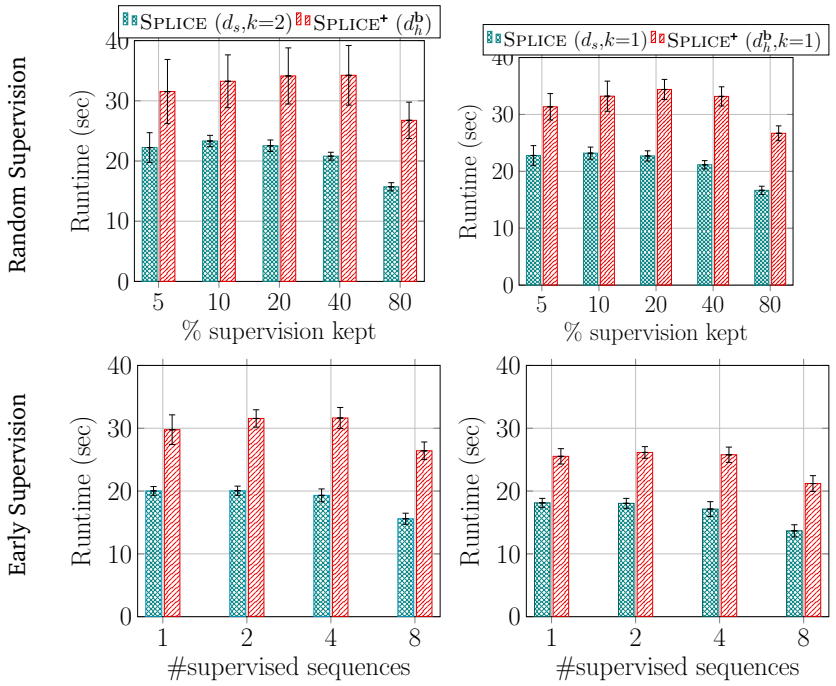

FIGURE 5.3: Runtime performance of supervision completion on `meet` (left) and `move` (right) as supervision increases. The runtime is macro-averaged over all samples. In the first scenario, supervision arrives uniformly at random (top), while in the second one is provided only at the beginning of training (bottom). We do not present the runtime of ILASP-NB here, since it is much higher than SPLICE$^+$ (≈ 400 seconds) and the scaling does not help the discussion of the results.

OLED alone only uses the supervised portion of each dataset for training, while everything else is ignored. As expected, SPLICE-OLED and SPLICE$^+$-OLED always outperform OLED, confirming that our supervision completion approach is indeed very helpful for learning good CE rules in the presence of missing labels. Comparing SPLICE$^+$-OLED to SPLICE-OLED the only noticeable difference is in the `meet` CE, when only limited supervision is available (less than 20% or 2 sequences). In that case, the SPLICE$^+$ labels lead to better structure learning, in both supervision settings. The same does not seem to hold for the `move` CE. This is mainly due to the fact that the `move` activity can be captured by a single rule and thus it is easier to learn from a small portion of data, while `meet` requires several distinct rules.

5.4 Evaluation on Maritime Monitoring

For the maritime monitoring dataset, we performed the same evaluation, as the one presented for activity recognition, for both the `pilotOps` and `rendezVous` CEs. The F_1-score of supervision completion on both scenarios, using the same notation is presented in Figure 5.5. The results suggest that SPLICE$^+$ effectively

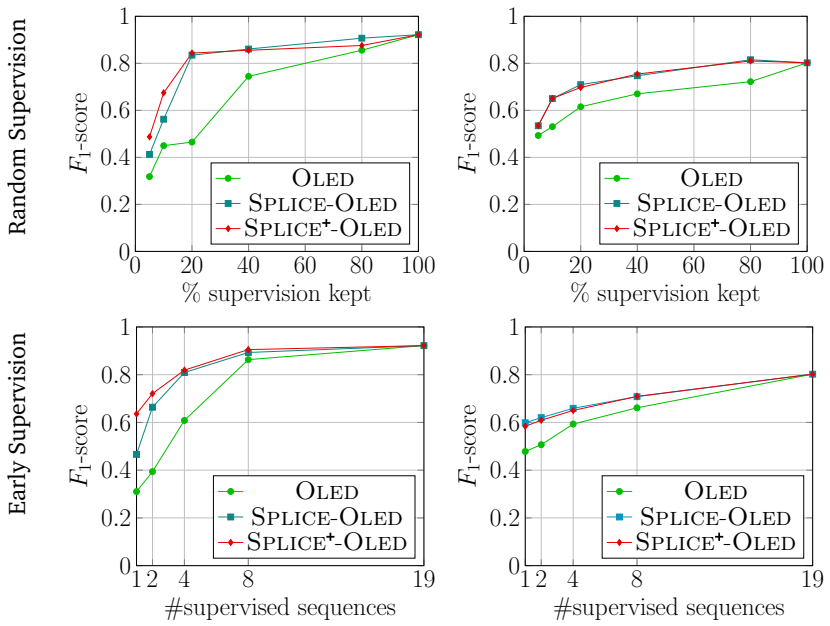

FIGURE 5.4: Structure learning using OLED on `meet` (left) and `move` (right) as supervision increases. In the first scenario, supervision arrives uniformly at random (top), while in the second one it is provided at the beginning of the training sequence (bottom).

infers the missing labels, significantly outperforms SPLICE in all cases, even for high supervision levels (80% uniform supervision or 8 sequences), while the difference is again larger in the second scenario, where supervision appears only in the beginning of training. Note that ILASP-NB did not converge after 5 hours of runtime and thus it was excluded from the evaluation.

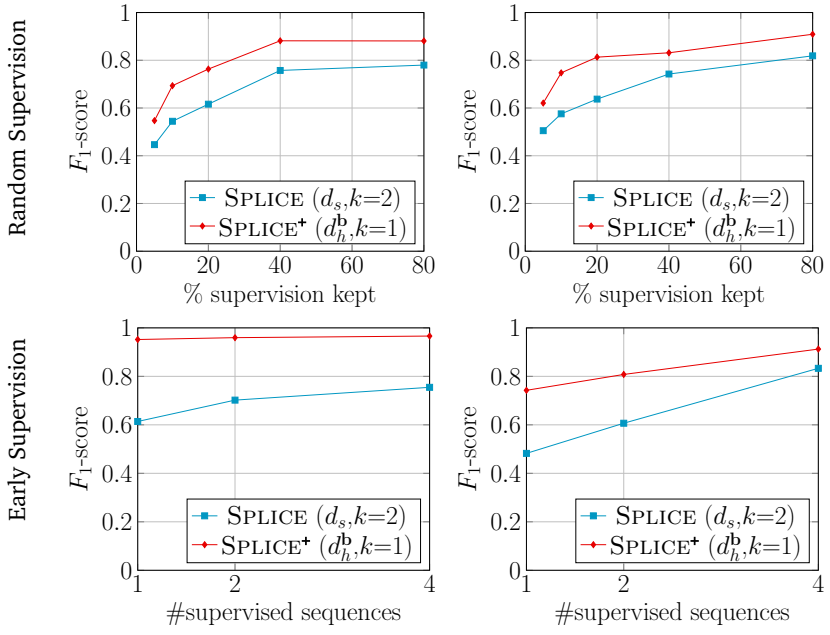

FIGURE 5.5: F_1-score of supervision completion on `pilotOps` (left) and `rendezVous` (right) as supervision increases.

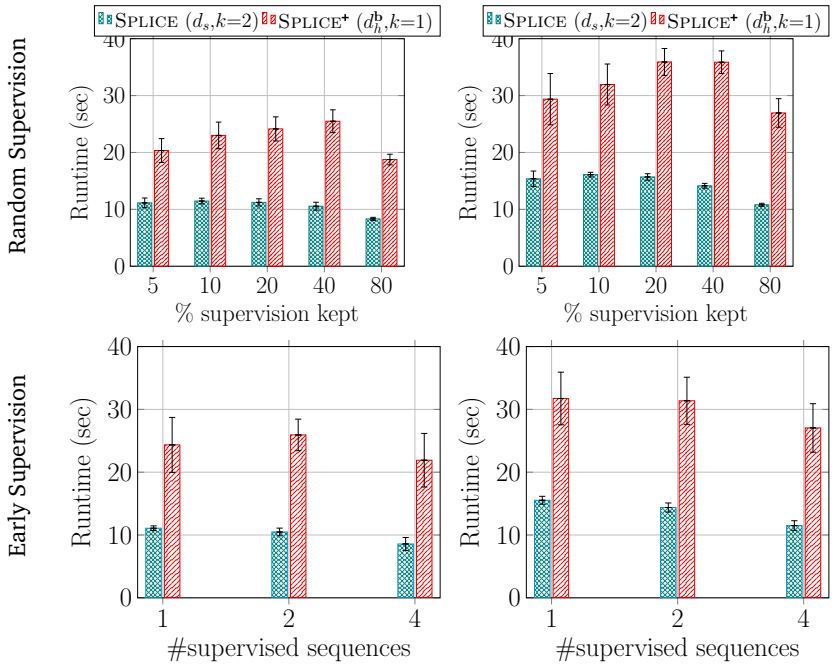

FIGURE 5.6: Runtime of supervision completion on `pilotOps` (left) and `rendezVous` (right) as supervision increases. The runtime is macro-averaged over all samples.

An interesting observation is that in the early supervision scenario the F_1-score of the `pilotOps` CE is very high even for 1 labelled sequence and does not change much as the supervision increases, which indicates that one sequence has enough labels to efficiently infer all the missing ones. Note that this performance is not matched by the random supervision scenario, even at 80%. However, in the random supervision scenario, in contrast to the early supervision one, some unlabelled data arrive before all the labelled data have been collected, which leads to mistakes.

As expected, the improved labelling accuracy of SPLICE⁺ comes with a cost in runtime performance over SPLICE. Recall that SPLICE⁺ is slower than SPLICE because it needs to update the trees for each micro-batch and perform feature selection. However, the computational penalty is still tolerable since it is below 25 seconds, as shown in Figure 5.6.

In Figure 5.7, we present the structure learning results of SPLICE-OLED against SPLICE⁺-OLED, and OLED alone using the incomplete dataset (OLED alone uses only the labelled examples). SPLICE⁺-OLED clearly outperforms both SPLICE-OLED and OLED alone by a large margin, which indicates the usefulness of the proposed approach. On the other hand, note that SPLICE-OLED does not always performs better than OLED alone. In particular, in some cases, such as below 10% supervision or given a single supervised sequence of data in `rendezVous` may also yield worse results, which is justified by the fact that SPLICE makes mistakes

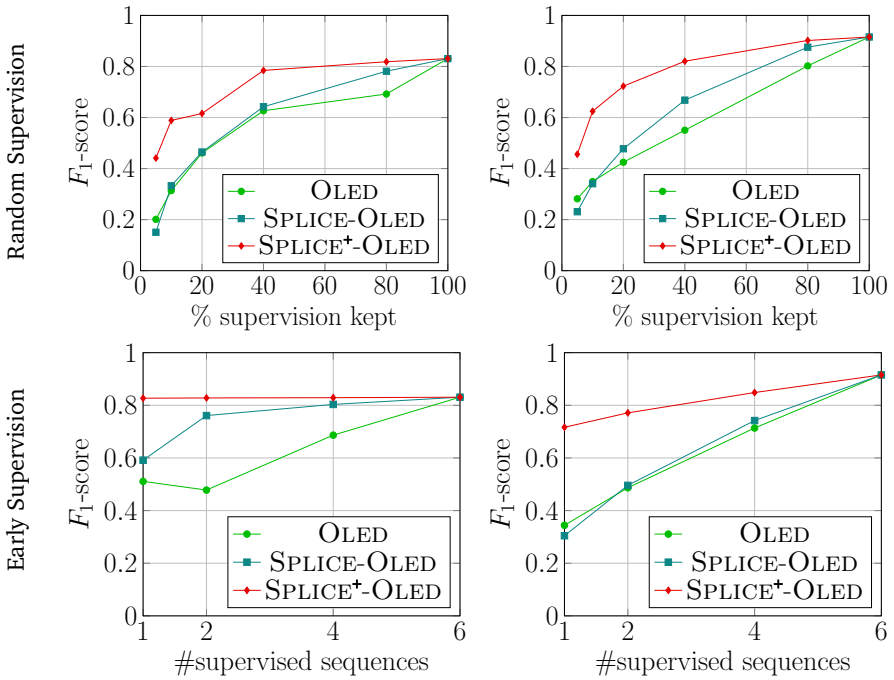

FIGURE 5.7: Structure learning on `pilotOps` (left) and `rendezVous` (right) as supervision increases. In the first scenario, supervision arrives uniformly at random (top), while in the second one is provided at the beginning of the training sequence (bottom).

that misguide structure learning. However, as soon as, enough supervision is provided it performs similar to OLED alone or even better.

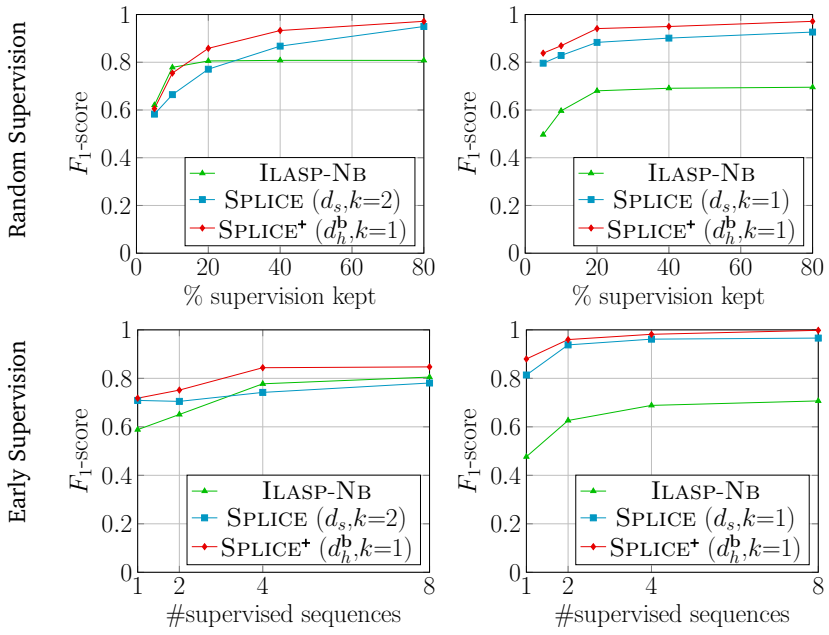

FIGURE 5.8: F_1-score of supervision completion on `nonEconomicDriving` (left) and `dangerousDriving` (right) as supervision increases.

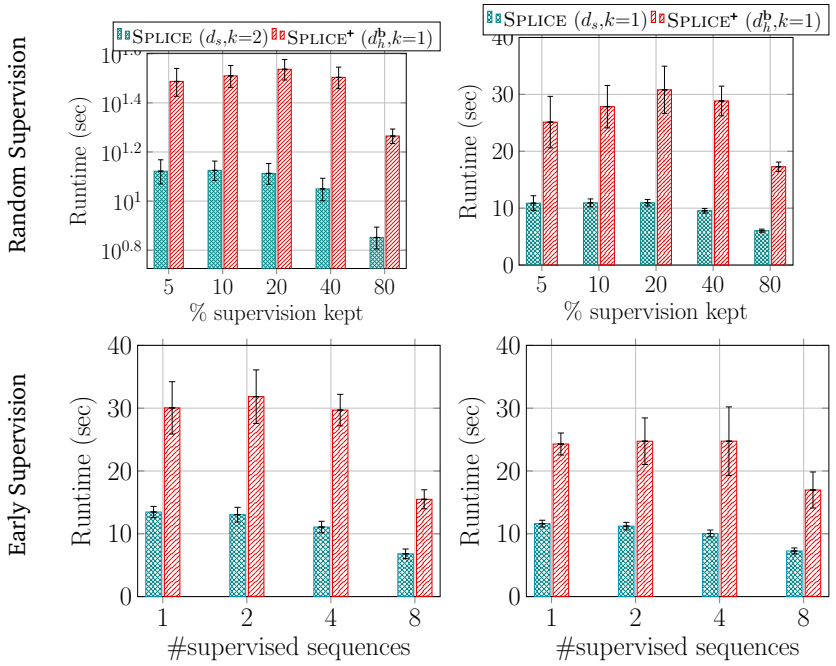

FIGURE 5.9: Runtime of supervision completion on `nonEconomicDriving` (left) and `dangerousDriving` (right) as supervision increases. The runtime is macro-averaged over all samples.

5.5 Evaluation on Fleet Management

For the fleet management dataset, the F_1-score of supervision completion on both scenarios for the `nonEconomicDriving` and `dangerousDriving` CEs is depicted

in Figure 5.8. The results appear to be consistent with the previous tasks, since SPLICE$^+$ yields the best overall performance. However, in this dataset the difference with SPLICE is smaller, due to the fact that fleet management dataset does not contain irrelevant or noisy features. Thus, the difference in performance is only due to the graph synopsis, that yields improved solutions, instead of the hybrid distance that accounts for feature significance. ILASP-NB, on the other hand, achieves comparable performance only in the random supervision scenario for `nonEconomicDriving`.

The absolute difference in runtime cost between SPLICE$^+$ and SPLICE is similar to that observed in the activity recognition dataset, as shown in Figure 5.9. Briefly, the computational penalty is typically below 20 seconds, due to the updates of the trees and feature selection. ILASP-NB, on the other hand, is 3 times slower than SPLICE$^+$, since it requires all the data to be available at once.

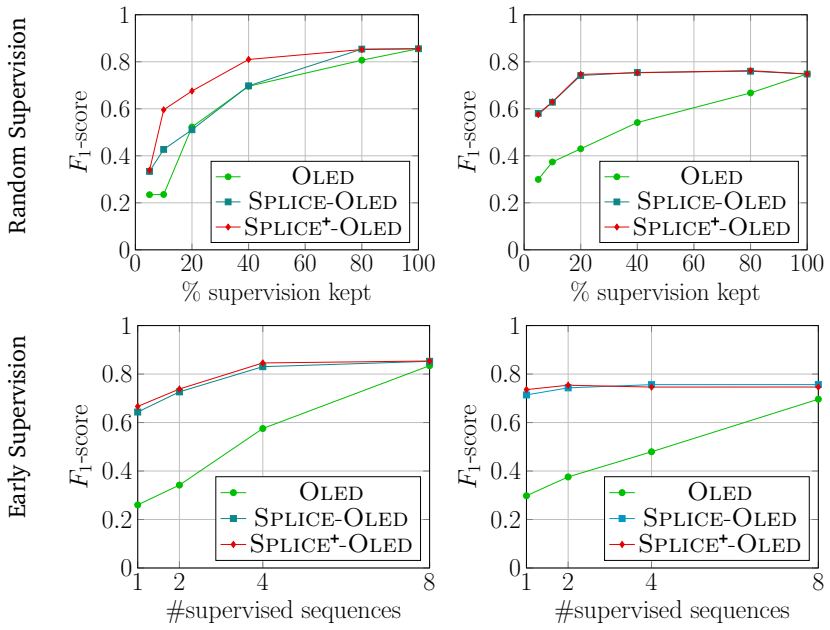

FIGURE 5.10: Structure learning on `nonEconomicDriving` (left) and `dangerousDriving` (right) as supervision increases. In the first scenario, supervision arrives uniformly at random (top), while in the second one, supervision is provided only at the beginning of the training sequence (bottom).

Finally, in Figure 5.10, we present the structure learning results of SPLICE-OLED against SPLICE$^+$-OLED, and OLED alone using the incomplete dataset (OLED alone uses only the labelled examples). SPLICE$^+$-OLED outperforms SPLICE-OLED in only one of the four subfigures, namely `nonEconomicDriving` under random supervision. This is in agreement with the supervision completion results, shown in Figure 5.8.

5.6 Discussion

In this section, we analyze the significance of the hybrid distance measure and the impact of the batch size on our proposed approaches. First, we present an ablation study over the distance components of SPLICE+ and discuss when each component may or may not be useful for improving performance. Then, we argue that the batch size does not significantly impact SPLICE+, due to its robust graph construction, while on the other hand, it may impact the perfromance of SPLICE.

5.6.1 Ablation Study

In order to further examine the contribution of each of the proposed improvements over SPLICE, in Table 5.1 we present the results of an ablation study on the activity recognition dataset, comparing the different components of SPLICE+ against each other. The first important observation is that the structural distance (d_s) performs very well, which indicates the importance of temporal connectivity. Mass-based dissimilarity alone ($\tilde{m}$) performs well enough in `meet` but rather poorly in `move`, especially in the early supervision setting. However, when combined with the structural distance (d_h) it always performs better than the structural distance alone. Another interesting observation is that while feature selection on the structural distance ($d_s^{\mathbf{b}}$) does not always achieve better results than the structural distance alone (d_s), it yields a synergistic effect when combined with the mass-based dissimilarity ($d_h^{\mathbf{b}}$). For instance, in the random supervision setting, using $d_h^{\mathbf{b}}$, instead of d_h, increases F_1-score from 0.63 to 0.67 for `meet` (corresponding to 242 errors on average), while in the early supervision setting, it increases F_1-score from 0.65 to 0.7 for `meet` and 0.7 to 0.73 for `move` (corresponding to 162 and 136 errors respectively). Therefore, the proposed hybrid measure achieves the best overall performance.

Similar to the task of human activity recognition, we analyse in the maritime monitoring dataset, the contribution of each of the proposed improvements over SPLICE. In Table 5.2 we compare the different components of SPLICE+ against each other. First, note that the structural distance (d_s) alone again performs well enough, given enough supervision (10% or 2 supervised sequences). Mass-based dissimilarity alone ($\tilde{m}$) seems to perform well in the `pilotOps` CE, but yields very poor performance in the `rendezVous` CE. Similar to activity recognition, when combined with the structural distance (d_h) a small, but consistent improvement of the performance is observed. In contrast to activity recognition, feature selection on the structural distance ($d_s^{\mathbf{b}}$) does not seem to improve the performance further. This is due to the synthetic supervision of the maritime dataset, which leads to

CE	Distance	Random Supervision		Early Supervision	
		5%	10%	1	2
meet	d_s	0.62	0.70	0.56	0.69
	$d_s^{\mathbf{b}}$	0.59	0.70	0.64	0.71
	m	0.65	0.75	0.64	0.70
	d_h	0.63	0.76	0.65	0.73
	$d_h^{\mathbf{b}}$	**0.67**	**0.77**	**0.70**	**0.76**
move	d_s	0.57	0.64	0.67	0.71
	$d_s^{\mathbf{b}}$	**0.58**	0.67	0.69	0.71
	m	0.56	0.68	0.60	0.54
	d_h	0.57	**0.69**	0.70	0.73
	$d_h^{\mathbf{b}}$	**0.58**	**0.69**	**0.73**	**0.75**

TABLE 5.1: Comparison of SPLICE⁺ on `meet` and `move` using the simple structural distance (d_s) and the hybrid distance ($d_h^{\mathbf{b}}$).

noise-free labels and features. Moreover, there are three irrelevant features in the dataset but only one of them appears frequently in the positive examples of the CEs, which renders the measurements of d_s quite similar to $d_s^{\mathbf{b}}$. Feature selection in this context is not expected to add value.

CE	Distance	Random Supervision		Early Supervision	
		5%	10%	1	2
rendezVous	d_s	0.59	0.70	0.69	0.79
	$d_s^{\mathbf{b}}$	0.59	0.70	0.69	0.79
	m	0.53	0.65	0.53	0.53
	d_h	**0.62**	**0.75**	**0.74**	**0.81**
	$d_h^{\mathbf{b}}$	**0.62**	**0.75**	**0.74**	**0.81**
pilotOps	d_s	0.47	0.63	0.78	0.90
	$d_s^{\mathbf{b}}$	0.47	0.63	0.78	0.90
	m	**0.56**	**0.69**	0.94	0.94
	d_h	**0.56**	**0.69**	**0.95**	**0.96**
	$d_h^{\mathbf{b}}$	**0.56**	**0.69**	**0.95**	**0.96**

TABLE 5.2: Comparison of SPLICE⁺ on `pilotOps` and `rendezVous`, using the simple structural distance (d_s) and the hybrid distance ($d_h^{\mathbf{b}}$).

Finally, the results in the fleet management dataset do not provide any insight in the discussion, since there are no irrelevant or noisy features in the dataset, leading to very similar distance measurements. As a general conclusion from the ablation studies that we performed in these datasets, SPLICE⁺ using all of its features, seems to provide the best performance, irrespective of the supervision setting.

CE	Batch size	Number of supervised sequences			
		1	2	4	8
meet	10	0.44/0.69	0.59/0.78	0.73/0.78	0.78/0.93
	25	0.43/0.69	0.57/0.74	0.72/0.78	0.78/0.93
	50	0.42/0.69	0.51/0.77	0.67/0.77	0.77/0.93
	100	0.42/0.69	0.56/0.76	0.75/0.80	0.77/0.93
move	10	0.66/0.73	0.73/0.75	0.71/0.79	0.84/0.94
	25	0.66/0.73	0.74/0.74	0.72/0.79	0.84/0.94
	50	0.66/0.73	0.74/0.78	0.74/0.81	0.84/0.94
	100	0.66/0.73	0.73/0.75	0.73/0.80	0.84/0.94

TABLE 5.3: F_1-score as batch size increases for `meet` and `move` CEs: SPLICE/SPLICE⁺.

5.6.2 Batch Size Impact

Table 5.3 presents the change in performance as the batch size increases on the activity recognition dataset. The F_1-score of SPLICE tends to fluctuate more than that of SPLICE⁺, as the batch size increases. For instance, in the `meet` CE, when 2 or 4 supervised sequences are provided, the F_1-score of SPLICE varies from 0.01 to 0.08, while SPLICE⁺ varies from 0.01 to 0.04. Corresponding changes are also noticeable when 1 or 8 supervised sequences are given to SPLICE, while SPLICE⁺ does not vary at all in these cases. Such variations also appear at a much smaller scale in `move`. These results suggest that SPLICE⁺ seems to be more robust to different batch sizes than SPLICE, which indicates the importance of the graph construction process.

CE	Batch size	Number of supervised sequences		
		1	2	4
pilotOps	10	0.63/0.96	0.88/0.97	0.92/0.97
	25	0.69/0.96	0.85/0.96	0.91/0.97
	50	0.71/0.96	0.88/0.96	0.91/0.97
	100	0.61/0.95	0.70/0.96	0.75/0.97
rendezVous	10	0.63/0.74	0.77/0.86	0.87/0.93
	25	0.58/0.74	0.72//0.86	0.84/0.90
	50	0.56/0.74	0.75/0.86	0.83/0.90
	100	0.48/0.75	0.61/0.81	0.83/0.92

TABLE 5.4: F_1-score of `pilotOps`, `rendezVous` for varying batch sizes: SPLICE/SPLICE⁺.

In Table 5.4, we present the change in F_1-score as the batch size increases for the matitime monitoring dataset. SPLICE⁺ is more robust than SPLICE, and this is more apparent in this dataset, compared to the activity recognition one. On `pilotOps`, when 1 supervised sequence is provided, the F_1-score of SPLICE varies from 0.02 to 0.1, while that of SPLICE⁺ varies only by 0.01. For 2 supervised

sequences the variation is even greater, since the F_1-score of SPLICE varies from 0.03 to 0.18, in contrast to that of SPLICE$^+$ where the variation remains at 0.01. The same holds for the `rendezVous` CE, where for 1 supervised sequence, SPLICE varies from 0.08 to 0.15, while SPLICE$^+$ varies only by 0.01.

5.7 Summary

The experimental results on the three real-life datasets showed that our proposed method can effectively learn Event Calculus theories even in the presence of irrelevant or noisy features. In such cases, it outperforms its predecessor (SPLICE), while in the simpler cases, such as in the fleet management dataset, it yields at least as good performance. On the other hand, ILASP-NB can achieve comparable accuracy in few cases for limited supervision but at the cost of significantly increased runtime performance. Although ILASP is a state-of-the-art ILP learning system that can learn Event Calculus, it is not designed for semi-supervised learning. Moreover, the co-training procedure of ILASP-NB is not practical for temporal data and may not converge in real-life datasets.

6 Conclusions and Future Work

"Science never solves a problem without creating ten more."
— George Bernard Shaw

In this thesis we focused on scalable semi-supervised learning for complex event recognition applications and proposed two methods for learning complex event rules from partially supervised training sequences. In Chapter 1 we briefly presented the basics of symbolic event recognition and pointed out the main advantages of the logic-based approaches, such as, the ability for robust temporal reasoning and the availability of machine learning tools that facilitate the automated discovery of complex event rules from data streams. Then, we argued that the assumption of a fully-labelled training sequence that arrives for processing is unrealistic in real-life applications, since usually sparse, infrequent labels are provided either on-stream or in the form of historical data. In Chapter 2 we briefly introduced the Event Calculus formalism and the OLED system for learning such CE rules from data streams. Then, we presented the background on graph-based methods to semi-supervised learning and distance metrics required for this thesis. Finally, we provided an overview of related approaches and argued that they are not appropriate for learning CE rules from data streams, either due to scalability issues or due to other assumptions (e.g., numerical data). In Chapters 3 and 4 we proposed two scalable approaches for inferring the missing labels in a partially supervised training sequence, thus enabling any supervised learner to operate on the completed training data. In Chapter 5, we presented an experimental study showing that both of these approaches can efficiently complete the missing labels and improve the predictive accuracy of the underlying structure learner. Moreover, the results suggest that they are efficient enough to be used in large temporal datasets. In what remains we conclude this thesis by summarising the basic traits of our proposed approaches and the respective experimental results, while we also provide some directions for future research.

6.1 Conclusions

In Chapter 3, we presented SPLICE, a novel approach to online structure learning of CE rules that operates on partially supervised training sequences. SPLICE infers the missing supervision continuously as the data arrive in micro-batches. To that end, it employs a graph-cut minimisation technique and a distance function for first-order logic to derive labels for unlabelled data, by computing their distance to their labelled counterparts. As it processes the input stream, SPLICE caches previously seen labelled examples for future usage and filters noisy, contradicting labelled examples that may compromise the overall accuracy of the learning task. Then, each fully-supervised micro-batch can be used by any online supervised structure learning system, such as OLED, in order to derive new or enhance existing complex event rules.

In Chapter 4, we presented SPLICE$^+$, an improved approach to online structure learning of CE rules from partially-supervised training sequences. Similar to its predecessor (SPLICE), the new method infers the missing labels continuously as the data arrive, and passes them on to an online supervised structure learner that constructs CE rules. In contrast to SPLICE, SPLICE$^+$ employs a hybrid distance measure combining a structural distance optimised for kNN classification through feature selection, and a data-driven measure, based on mass estimation. The combined measure exploits the labelled data for supervised metric learning and the unlabelled data for estimating the distribution of examples. Moreover, SPLICE$^+$ constructs a temporal graph and maintains a synopsis of the data stream to achieve robust labelling.

In Chapter 5, we presented experimental results in the domain of complex event recognition, using a benchmark dataset for activity recognition, a real dataset for maritime monitoring, and a dataset for fleet management. The results showed that SPLICE can enable the underlying structure learner to learn good CE rules, even in the presence of very limited given annotation. On the other hand, SPLICE$^+$ outperforms its predecessor (SPLICE) in terms of completing the missing labels and improving the predictive accuracy of the underlying structure learner. Moreover, it seems particularly effective when the supervision is provided only at the beginning of the stream. Finally, the comparison to a batch learning system combining ILASP and Naive Bayes, to perform a form of co-training, resulted in inferior results and much higher computational requirements.

6.2 Future Work

There are several interesting directions in which we indend to extend the work presented in this thesis. The main ones are presented below:

Active Learning

SPLICE$^+$ is capable of identifying noisy or irrelevant features using a novel informed distance measure. However, the proposed approach still may not correctly infer the labels of rare patterns or emerging classes. To that end, we would like to investigate active learning techniques [Fu et al., 2013], in order to select promising example instances or areas of the sample space and request user information to further enhance the predictions, in the presence of very noisy labels or concept drift. A straightforward approach of achieving that, similar to the uncertainty sampling methods [Culotta and McCallum, 2005], is to exploit the real-valued labelling generated by the harmonic solution (see Section 2.2.1). Label values closer to the decision boundary, that is closer to zero, can be considered more uncertain and be chosen for investigation.

Distributed Semi-Supervised Learning

The scalability of the learning systems presented in this work can be further improved, via parallelizing parts of the learning task and distributing the workload over multiple processing cores or computing nodes. Some degree of parallelization already exists in the methods that we presented in this thesis, e.g., both methods compute the distances between examples in parallel. However, there are other promising directions that we have indentified for further improving the performance, such as building and updating the Half-Space Trees in parallel. Additionally, there are promising ideas for distributing the computation by partitioning the input data over multiple computing nodes, using the ground query atom constants. For instance, every distinct pair of IDs involved in a query may be processed by a different computing node. To that end, an efficient data partitioning technique is necessery, in order to guarantee good load balancing among the nodes, while maintaining low data redundancy. The resulting distributed version of SPLICE can be combined with a distributed version of OLED [Katzouris et al., 2019] in order to learn CE rules efficiently.

Bibliography

Abdulsalam, H., Skillicorn, D. B., and Martin, P. (2011). Classification using streaming random forests. *IEEE Trans. Knowl. Data Eng.*, 23(1):22–36.

Abney, S. P. (2002). Bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 360–367.

Aha, D. W., Kibler, D. F., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

Alphonse, É. and Matwin, S. (2002). Feature subset selection and inductive logic programming. In Sammut, C. and Hoffmann, A. G., editors, *Proceedings of the 9th International Conference on Machine Learning*, pages 11–18. Morgan Kaufmann.

Anicic, D., Rudolph, S., Fodor, P., and Stojanovic, N. (2012). Stream reasoning and complex event processing in ETALIS. *Semantic Web*, 3(4):397–407.

Arauzo-Azofra, A., Benítez, J. M., and Castro, J. L. (2008). Consistency measures for feature selection. *J. Intell. Inf. Syst.*, 30(3):273–292.

Artikis, A., Katzouris, N., Correia, I., Baber, C., Morar, N., Skarbovsky, I., Fournier, F., and Paliouras, G. (2017). A prototype for credit card fraud management: Industry paper. In *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, pages 249–260. ACM.

Artikis, A., Paliouras, G., Portet, F., and Skarlatidis, A. (2010). Logic-based representation, reasoning and machine learning for event recognition. In *Proceedings of the 4th ACM International Conference on Distributed Event-Based Systems*, pages 282–293.

Artikis, A., Sergot, M. J., and Paliouras, G. (2015). An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):895–908.

Artikis, A., Skarlatidis, A., Portet, F., and Paliouras, G. (2012). Logic-based event recognition. *Knowledge Engineering Review*, 27(4):469–506.

Aryal, S., Ting, K. M., Haffari, G., and Washio, T. (2014). Mp-dissimilarity: A data dependent dissimilarity measure. In *Proceedings of the IEEE International Conference on Data Minin (ICDM)*, pages 707–712.

Athanasopoulos, G., Paliouras, G., Vogiatzis, D., Tzortzis, G., and Katzouris, N. (2018). Predicting the evolution of communities with online inductive logic programming. In *Proceedings of the 25th International Symposium on Temporal Representation and Reasoning (TIME)*, pages 4:1–4:20.

Azran, A. (2007). The rendezvous algorithm: multiclass semi-supervised learning with markov random walks. In *Proceedings of the 24th International Conference on Machine Learning*, pages 49–56.

Bachman, P., Alsharif, O., and Precup, D. (2014). Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 3365–3373.

Balcan, M., Blum, A., and Yang, K. (2004). Co-training and expansion: Towards bridging theory and practice. In *Advances in Neural Information Processing Systems 17*, pages 89–96.

Benabdeslem, K. and Hindawi, M. (2014). Efficient semi-supervised feature selection: Constraint, relevance, and redundancy. *IEEE Trans. Knowl. Data Eng.*, 26(5):1131–1143.

Bennett, K. P., Demiriz, A., and Maclin, R. (2002). Exploiting unlabeled data in ensemble methods. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 289–296.

Bie, T. D. and Cristianini, N. (2003). Convex methods for transduction. In *Advances in Neural Information Processing Systems 16*, pages 73–80.

Bisson, G. (1992a). Conceptual clustering in a first order logic representation. In *Proceedinds of the 10th European Conference on Artificial Intelligence*, pages 458–462. Wiley.

Bisson, G. (1992b). Learning in FOL with a similarity measure. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 82–87. AAAI Press / MIT Press.

Blockeel, H. and Raedt, L. D. (1998). Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297.

Blockeel, H., Raedt, L. D., Jacobs, N., and Demoen, B. (1999). Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery*, 3(1):59–93.

Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conference on Machine Learning*, pages 19–26. Morgan Kaufmann.

Blum, A., Lafferty, J. D., Rwebangira, M. R., and Reddy, R. (2004). Semi-supervised learning using randomized mincuts. In *Proceedings of the 21st International Conference on Machine Learning*. ACM.

Blum, A. and Mitchell, T. M. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 92–100. ACM.

Bohnebeck, U., Horváth, T., and Wrobel, S. (1998). Term comparisons in first-order similarity measures. In *Proceedings of the 8th International Workshop on Inductive Logic Programming*, pages 65–79. Springer.

Brendel, W., Fern, A., and Todorovic, S. (2011). Probabilistic event logic for interval-based event recognition. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3329–3336.

Broda, K., Clark, K., Miller, R., and Russo, A. (2009). SAGE: A logical agent-based environment monitoring and control system. In *Proceedings of the European Conference on Ambient Intelligence*, pages 112–117.

Brown, G. (2009). A new perspective for information theoretic feature selection. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 49–56.

Bruns, R., Dunkel, J., and Offel, N. (2019). Learning of complex event processing rules with genetic programming. *Expert Syst. Appl.*, 129:186–199.

Callens, L., Carrault, G., Cordier, M., Fromont, É., Portet, F., and Quiniou, R. (2008). Intelligent adaptive monitoring for cardiac surveillance. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*, pages 653–657.

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28.

Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. MIT Press.

Chapelle, O., Sindhwani, V., and Keerthi, S. S. (2008). Optimization techniques for semi-supervised support vector machines. *J. Mach. Learn. Res.*, 9:203–233.

Chaudet, H. (2006). Extending the event calculus for tracking epidemic spread. *Artif. Intell. Medicine*, 38(2):137–156.

Chen, B., Liu, H., Chai, J., and Bao, Z. (2009). Large margin feature weighting method via linear programming. *IEEE Trans. Knowl. Data Eng.*, 21(10):1475–1488.

Collobert, R., Sinz, F. H., Weston, J., and Bottou, L. (2006). Large scale transductive svms. *J. Mach. Learn. Res.*, 7:1687–1712.

Corduneanu, A. and Jaakkola, T. S. (2003). On information regularization. In *Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence*, pages 151–158.

Cugola, G. and Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Computing Survey*, 44(3):15:1–15:62.

Culotta, A. and McCallum, A. (2005). Reducing labeling effort for structured prediction tasks. In *Proceedings of the 20th National Conference on Artificial Intelligence*, page 746–751. AAAI Press.

Culp, M. and Michailidis, G. (2008). An iterative algorithm for extending learners to a semi-supervised setting. *Journal of Computational and Graphical Statistics*, 17(3):545–571.

d'Alché-Buc, F., Grandvalet, Y., and Ambroise, C. (2001). Semi-supervised marginboost. In *Advances in Neural Information Processing Systems 14*, pages 553–560.

De Raedt, L. and Dehaspe, L. (1997). Clausal discovery. *Machine Learning*, 26(2-3):99–146.

de Sousa, C. A. R., Rezende, S. O., and Batista, G. E. A. P. A. (2013). Influence of graph construction on semi-supervised learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 160–175.

Delalleau, O., Bengio, Y., and Roux, N. L. (2005). Efficient non-parametric function induction in semi-supervised learning. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 96–103.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Sociaty*, 39(1):1–38.

Deng, C. and Guo, M. (2011). A new co-training-style random forest for computer aided diagnosis. *J. Intell. Inf. Syst.*, 36(3):253–281.

Dhurandhar, A. and Dobra, A. (2012). Distribution-free bounds for relational classification. *Knowledge and Information Systems*, 31(1):55–78.

Domingos, P. M. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining*, pages 71–80.

Dörfler, F. and Bullo, F. (2013). Kron reduction of graphs with applications to electrical networks. *IEEE Trans. on Circuits and Systems*, 60-I(1):150–163.

Dousson, C. and Maigat, P. L. (2007). Chronicle recognition improvement using temporal focusing and hierarchization. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 324–329.

Dries, A. and Raedt, L. D. (2009). Towards clausal discovery for stream mining. In *Proceedings of the 19th International Conference on Inductive Logic Programming (ILP)*, pages 9–16.

Du, J., Ling, C. X., and Zhou, Z. (2011). When does cotraining work in real data? *IEEE Trans. Knowl. Data Eng.*, 23(5):788–799.

Dyer, K. B. and Polikar, R. (2012). Semi-supervised learning in initially labeled non-stationary environments with gradual drift. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9.

Emde, W. and Wettschereck, D. (1996). Relational instance-based learning. In *Proceedings of the 13th International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann.

Etzion, O. and Niblett, P. (2010). *Event Processing in Action*. Manning Publications Company.

Falcionelli, N., Sernani, P., de la Torre, A. B., Mekuria, D. N., Calvaresi, D., Schumacher, M., Dragoni, A. F., and Bromuri, S. (2019). Indexing the event calculus: Towards practical human-readable personal health systems. *Artif. Intell. Medicine*, 96:154–166.

Ferilli, S., Basile, T. M. A., Biba, M., Mauro, N. D., and Esposito, F. (2009). A general similarity framework for horn clause logic. *Fundam. Inform.*, 90(1-2):43–66.

Fu, Y., Zhu, X., and Li, B. (2013). A survey on instance selection for active learning. *Knowl. Inf. Syst.*, 35(2):249–283.

Fürnkranz, J., Gamberger, D., and Lavrac, N. (2012). *Foundations of Rule Learning*. Cognitive Technologies. Springer.

Gama, J. (2010). *Knowledge Discovery from Data Streams*. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press.

Gayathri, K. S., Easwarakumar, K. S., and Elias, S. (2017). Probabilistic ontology based activity recognition in smart homes using markov logic network. *Knowl. Based Syst.*, 121:173–184.

Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning*. MIT Press.

Ghahramani, Z. and Jordan, M. I. (1993). Supervised learning from incomplete data via an EM approach. In *Proceedings of the 7th Conference on Advances in Neural Information Processing Systems 6*, pages 120–127. Morgan Kaufmann.

Giatrakos, N., Alevizos, E., Artikis, A., Deligiannakis, A., and Garofalakis, M. N. (2020). Complex event recognition in the big data era: a survey. *VLDB J.*, 29(1):313–352.

Gogate, V., Webb, W. A., and Domingos, P. M. (2010). Learning efficient markov networks. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems 2010*, pages 748–756. Curran Associates, Inc.

Goldberger, J., Roweis, S. T., Hinton, G. E., and Salakhutdinov, R. (2004). Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, pages 513–520.

Goldman, S. A. and Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 327–334. Morgan Kaufmann.

Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems 17*, pages 529–536.

Grandvalet, Y., d'Alché-Buc, F., and Ambroise, C. (2001). Boosting mixture models for semi-supervised learning. In *Proceedings of the 2001 International Conference on Artificial Neural Networks*, pages 41–48.

Grez, A., Riveros, C., and Ugarte, M. (2019). A formal framework for complex event processing. In *Proceedings of the 22nd International Conference on Database Theory*, pages 5:1–5:18.

Grez, A., Riveros, C., Ugarte, M., and Vansummeren, S. (2020). On the expressiveness of languages for complex event recognition. In *Proceedings of the 23rd International Conference on Database Theory*, pages 15:1–15:17.

Guillou, X. L., Cordier, M., Robin, S., and Rozé, L. (2008). Chronicles for on-line diagnosis of distributed systems. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 194–198.

Guyon, I., Nikravesh, M., Gunn, S. R., and Zadeh, L. A., editors (2006). *Feature Extraction - Foundations and Applications*, volume 207 of *Studies in Fuzziness and Soft Computing*. Springer.

Haffari, G. and Sarkar, A. (2007). Analysis of semi-supervised learning with the yarowsky algorithm. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 159–166.

Hausdorff, F. (1962). *Set Theory*. AMS Chelsea Publishing Series. Chelsea Publishing Company.

Hoeffding, W. (1963a). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

Hoeffding, W. (1963b). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

Horváth, T., Wrobel, S., and Bohnebeck, U. (2001). Relational instance-based learning with lists and terms. *Mach. Learn.*, 43(1/2):53–80.

Huang, L., Liu, X., Ma, B., and Lang, B. (2015). Online semi-supervised annotation via proxy-based local consistency propagation. *Neurocomputing*, 149:1573–1586.

Huynh, T. N. and Mooney, R. J. (2011). Online Structure Learning for Markov Logic Networks. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, volume 2, pages 81–96.

Jebara, T., Wang, J., and Chang, S. (2009). Graph construction and *b*-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 441–448.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 290–297.

Kafali, Ö., Romero, A. E., and Stathis, K. (2017). Agent-oriented activity recognition in the event calculus: An application for diabetic patients. *Comput. Intell.*, 33(4):899–925.

Karasuyama, M. and Mamitsuka, H. (2013). Manifold-based similarity adaptation for label propagation. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 1547–1555.

Katzouris, N. (2017). *Scalable Relational Learning for Event Recognition*. PhD thesis, National and Kapodistrian University of Athens.

Katzouris, N., Artikis, A., and Paliouras, G. (2015). Incremental learning of event definitions with inductive logic programming. *Machine Learning*, 100(2-3):555–585.

Katzouris, N., Artikis, A., and Paliouras, G. (2016). Online learning of event definitions. *Theory and Practice of Logic Programming*, 16(5-6):817–833.

Katzouris, N., Artikis, A., and Paliouras, G. (2019). Parallel online event calculus learning for complex event recognition. *Future Generation Comp. Syst.*, 94:468–478.

Katzouris, N., Michelioudakis, E., Artikis, A., and Paliouras, G. (2018). Online learning of weighted relational rules for complex event recognition. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 396–413.

Kim, J., Jeon, M., Park, H., Bae, S., Bang, S., and Park, Y. (2019). An approach for recognition of human's daily living patterns using intention ontology and event calculus. *Expert Syst. Appl.*, 132:256–270.

Kirsten, M. and Wrobel, S. (1998). Relational distance-based clustering. In *Proceedings of the 8th International Workshop on Inductive Logic Programming*, pages 261–270. Springer.

Kirsten, M. and Wrobel, S. (2000). Extending k-means clustering to first-order representations. In *Proceedings of the 10th International Conference on Inductive Logic Programming*, pages 112–129. Springer.

Kowalski, R. A. and Sergot, M. J. (1986). A logic-based calculus of events. *New Generation Computing*, 4(1):67–95.

Krempl, G., Zliobaite, I., Brzezinski, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., and Stefanowski, J. (2014). Open challenges for data stream mining research. *SIGKDD Explorations*, 16(1):1–10.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Kulis, B. (2013). Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

Landwehr, N., Kersting, K., and Raedt, L. D. (2007). Integrating naïve bayes and FOIL. *Journal of Machine Learning Research*, 8:481–507.

Landwehr, N., Passerini, A., Raedt, L. D., and Frasconi, P. (2006). kfoil: Learning simple relational kernels. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 389–394. AAAI Press.

Law, M., Russo, A., and Broda, K. (2016). Iterative learning of answer set programs from context dependent examples. *Theory Pract. Log. Program.*, 16(5-6):834–848.

Law, M., Russo, A., and Broda, K. (2018). Inductive learning of answer set programs from noisy examples. *Advances in Cognitive Systems*.

Lawrence, N. D. and Jordan, M. I. (2004). Semi-supervised learning via gaussian processes. In *Advances in Neural Information Processing Systems 17*, pages 753–760.

Lee, S., Ganapathi, V., and Koller, D. (2006). Efficient structure learning of markov networks using l_1-regularization. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems*, pages 817–824. MIT Press.

Leistner, C., Saffari, A., Santner, J., and Bischof, H. (2009). Semi-supervised random forests. In *Proceedings of the IEEE 12th International Conference on Computer Vision*, pages 506–513.

Li, M. and Zhou, Z. (2007). Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 37(6):1088–1098.

Li, Y. and Guo, M. (2011). Web page classification using relational learning algorithm and unlabeled data. *Journal of Computers*, 6(3):474–479.

Li, Y. and Guo, M. (2012). A new relational tri-training system with adaptive data editing for inductive logic programming. *Knowl.-Based Syst.*, 35:173–185.

Liu, F. T., Ting, K. M., and Zhou, Z. (2008). Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, pages 413–422.

Loreti, D., Chesani, F., Mello, P., Roffia, L., Antoniazzi, F., Cinotti, T. S., Paolini, G., Masotti, D., and Costanzo, A. (2019). Complex reactive event processing for assisted living: The habitat project case study. *Expert Syst. Appl.*, 126:200–217.

Luckham, D. C. (2002). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. ddison-Wesley Longman Publishing Co., Inc.

Mallapragada, P. K., Jin, R., Jain, A. K., and Liu, Y. (2009). Semiboost: Boosting for semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(11):2000–2014.

Margara, A., Cugola, G., and Tamburrelli, G. (2014). Learning from the past: automated rule generation for complex event processing. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (DEBS)*, pages 47–58.

Mavroeidis, D. and Flach, P. A. (2003). Improved distances for structured data. In *Proceedings of the 13th International Workshop on Inductive Logic Programming (ILP)*, pages 251–268.

McCallum, A. (2003). Efficiently inducing features of conditional random fields. In *Proceedings of the 19th conference on Uncertainty in Artificial Intelligence*, pages 403–410.

Michelioudakis, E., Artikis, A., and Paliouras, G. (2016a). Online structure learning for traffic management. In *Proceedings of the 26th International Conference on Inductive Logic Programming*, pages 27–39.

Michelioudakis, E., Artikis, A., and Paliouras, G. (2019). Semi-supervised online structure learning for composite event recognition. *Machine Learning*, 108(7):1085–1110.

Michelioudakis, E., Skarlatidis, A., Paliouras, G., and Artikis, A. (2016b). Online structure learning using background knowledge axiomatization. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, volume 1, pages 242–237.

Miyato, T., Maeda, S., Koyama, M., and Ishii, S. (2019). Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(8):1979–1993.

Modrzejewski, M. (1993). Feature selection using rough sets theory. In *Proceedings of the European Conference on Machine Learning*, pages 213–226.

Montali, M., Maggi, F. M., Chesani, F., Mello, P., and van der Aalst, W. M. P. (2013). Monitoring business constraints with the event calculus. *ACM TIST*, 5(1):17:1–17:30.

Mousheimish, R., Taher, Y., and Zeitouni, K. (2017). Automatic learning of predictive CEP rules: Bridging the gap between data mining and complex event processing. In *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems (DEBS)*, pages 158–169.

Mueller, E. T. (2008a). Event Calculus. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 671–708. Elsevier.

Mueller, E. T. (2008b). Event Calculus. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 671–708. Elsevier.

Muggleton, S. (1995). Inverse entailment and progol. *New Generation Comput.*, 13(3&4):245–286.

Muggleton, S., Lodhi, H., Amini, A., and Sternberg, M. J. E. (2005). Support vector inductive logic programming. In *Proceedings of the 8th International Conference on Discovery Science*, pages 163–175.

Muggleton, S. and Raedt, L. D. (1994). Inductive logic programming: Theory and methods. *J. Log. Program.*, 19/20:629–679.

Nienhuys-Cheng, S.-H. (1997). Distance Between Herbrand Interpretations: A Measure for Approximations to a Target Concept. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, pages 213–226. Springer-Verlag.

Okada, S. and Nishida, T. (2010). Multi class semi-supervised classification with graph construction based on adaptive metric learning. In *Proceedings of the 20th International Conference on Artificial Neural Networks (ICNN)*, pages 468–478.

Park, S., Park, J., Shin, S., and Moon, I. (2018). Adversarial dropout for supervised and semi-supervised learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 3917–3924.

Paschke, A. and Kozlenkov, A. (2009). Rule-based event processing and reaction rules. In *Proceedings of the International Symposium on Rule Interchange and Applications*, pages 53–66.

Patroumpas, K., Alevizos, E., Artikis, A., Vodas, M., Pelekis, N., and Theodoridis, Y. (2017). Online event recognition from moving vessel trajectories. *GeoInformatica*, 21(2):389–427.

Patroumpas, K., Artikis, A., Katzouris, N., Vodas, M., Theodoridis, Y., and Pelekis, N. (2015). Event recognition for maritime surveillance. In *Proceedings of the 18th International Conference on Extending Database Technology (EDBT)*, pages 629–640.

Pawlak, Z., Grzymala-Busse, J. W., Slowinski, R., and Ziarko, W. (1995). Rough sets. *Commun. ACM*, 38(11):88–95.

Pietra, S. D., Pietra, V. D., and Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

Pitsikalis, M., Artikis, A., Dreo, R., Ray, C., Camossi, E., and Jousselme, A. (2019). Composite event recognition for maritime monitoring. In *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems (DEBS)*, pages 163–174.

Plotkin, G. D. (1971). *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University.

Pourdamghani, N., Rabiee, H. R., and Zolfaghari, M. (2012). Metric learning for graph based semi-supervised human pose estimation. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, pages 3386–3389.

Prapas, I., Paliouras, G., Artikis, A., and Baskiotis, N. (2018). Towards human activity reasoning with computational logic and deep learning. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence (SETN)*, pages 27:1–27:4.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, 5:239–266.

Raedt, L. D. (2008). *Logical and Relational Learning: From ILP to MRDM (Cognitive Technologies)*. Springer.

Ramon, J. and Bruynooghe, M. (1998). A framework for defining distances between first-order logic objects. In *Proceedings of the 8th International Workshop on Inductive Logic Programming*, pages 271–280. Springer.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems 28*, pages 3546–3554.

Ray, O. (2009). Nonmonotonic abductive inductive learning. *J. Appl. Log.*, 7(3):329–340.

Richardson, M. and Domingos, P. M. (2006). Markov logic networks. *Mach. Learn.*, 62(1-2):107–136.

Rodríguez, N. D., Cuéllar, M. P., Lilius, J., and Calvo-Flores, M. D. (2013). A survey on ontologies for human behavior recognition. *ACM Comput. Surv.*, 46(4):43:1–43:33.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Schultz-Møller, N. P., Migliavacca, M., and Pietzuch, P. R. (2009). Distributed complex event processing with query rewriting. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems (DEBS)*.

Sebag, M. (1997). Distance induction in first order logic. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, pages 264–272.

Sebag, M. and Schoenauer, M. (1993). A rule-based similarity measure. In *1st European Workshop on Topics in Case-Based Reasoning*, pages 119–131.

Skarlatidis, A. (2014). *Event Recognition Under Uncertainty and Incomplete Data*. PhD thesis, University of Piraeus.

Skarlatidis, A., Artikis, A., Filipou, J., and Paliouras, G. (2015a). A probabilistic logic programming event calculus. *Theory Pract. Log. Program.*, 15(2):213–245.

Skarlatidis, A., Paliouras, G., Artikis, A., and Vouros, G. A. (2015b). Probabilistic event calculus for event recognition. *ACM Trans. Comput. Log.*, 16(2):11:1–11:37.

Skarlatidis, A., Paliouras, G., Artikis, A., and Vouros, G. A. (2015c). Probabilistic Event Calculus for Event Recognition. *ACM Transactions on Computational Logic*, 16(2):11:1–11:37.

Song, K., Nie, F., Han, J., and Li, X. (2017). Parameter free large margin nearest neighbor for distance metric learning. In Singh, S. P. and Markovitch, S., editors, *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 2555–2561. AAAI Press.

Soonthornphisaj, N. and Kijsirikul, B. (2004). Combining ILP with semi-supervised learning for web page categorization. In *Proceedings of the International Conference on Computational Intelligence*, pages 322–325.

Srinivasan, A. (2003). The aleph manual. Technical Report 4, Computing Laboratory, Oxford University.

Srinivasan, A. and Bain, M. (2017). An empirical study of on-line models for relational data streams. *Machine Learning*, 106(2):243–276.

Storf, H., Kleinberger, T., Becker, M., Schmitt, M., Bomarius, F., and Prueckner, S. (2009). An event-driven approach to activity recognition in ambient assisted living. In *Proceedings of the European Conference on Ambient Intelligence*, pages 123–132.

Subramanya, A. and Talukdar, P. P. (2014). *Graph-Based Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Szummer, M. and Jaakkola, T. S. (2001). Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems 14*, pages 945–952. MIT Press.

Tanha, J., van Someren, M., and Afsarmanesh, H. (2017). Semi-supervised self-training for decision tree classifiers. *Int. J. Machine Learning & Cybernetics*, 8(1):355–370.

Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 1195–1204.

Tenenbaum, J. B., Silva, V. d., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

Ting, K. M., Zhou, G., Liu, F. T., and Tan, S. C. (2013). Mass estimation. *Machine Learning*, 90(1):127–160.

Ting, K. M., Zhu, Y., Carman, M. J., Zhu, Y., Washio, T., and Zhou, Z. (2019). Lowest probability mass neighbour algorithms: relaxing the metric constraint in distance-based neighbourhood algorithms. *Machine Learning*, 108(2):331–376.

Triguero, I., García, S., and Herrera, F. (2015). Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowl. Inf. Syst.*, 42(2):245–284.

Tsilionis, E., Koutroumanis, N., Nikitopoulos, P., Doulkeridis, C., and Artikis, A. (2019). Online event recognition from moving vehicles: Application paper. *Theory Pract. Log. Program.*, 19(5-6):841–856.

Turaga, P. K., Chellappa, R., Subrahmanian, V. S., and Udrea, O. (2008). Machine recognition of human activities: A survey. *IEEE Trans. Circuits Syst. Video Techn.*, 18(11):1473–1488.

Valko, M., Kveton, B., Huang, L., and Ting, D. (2010). Online semi-supervised learning on quantized graphs. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 606–614.

van Engelen, J. E. and Hoos, H. H. (2020). A survey on semi-supervised learning. *Mach. Learn.*, 109(2):373–440.

Vapnik, V. (1998). *Statistical learning theory*. Wiley.

Vergara, J. R. and Estévez, P. A. (2014). A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1):175–186.

Wagner, T., Guha, S., Kasiviswanathan, S. P., and Mishra, N. (2018). Semi-supervised learning on data streams via temporal label propagation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 5082–5091.

Wang, F. and Sun, J. (2015). Survey on distance metric learning and dimensionality reduction in data mining. *Data Min. Knowl. Discov.*, 29(2):534–564.

Wang, F. and Zhang, C. (2008). Label propagation through linear neighborhoods. *IEEE Trans. Knowl. Data Eng.*, 20(1):55–67.

Wang, J., Jebara, T., and Chang, S. (2008). Graph transduction via alternating minimization. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1144–1151.

Wang, J., Jebara, T., and Chang, S. (2013a). Semi-supervised learning using greedy max-cut. *J. Mach. Learn. Res.*, 14(1):771–800.

Wang, W. and Zhou, Z. (2007). Analyzing co-training style algorithms. In *Proceedings of the 18th European Conference on Machine Learning*, pages 454–465.

Wang, Y., Cao, K., and Zhang, X. (2013b). Complex event processing over distributed probabilistic event streams. *Comput. Math. Appl.*, 66(10):1808–1821.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244.

Wu, B., Russo, A., Law, M., and Inoue, K. (2018). Learning commonsense knowledge through interactive dialogue. In *Technical Communications of the 34th International Conference on Logic Programming*, pages 12:1–12:19.

Wu, X., Li, Z., So, A. M., Wright, J., and Chang, S. (2012). Learning with partially absorbing random walks. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, pages 3086–3094.

Xu, C., Tao, D., and Xu, C. (2013). A survey on multi-view learning. *CoRR*, abs/1304.5634.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.

Zhang, D., Chen, S., and Zhou, Z. (2008). Constraint score: A new filter method for feature selection with pairwise constraints. *Pattern Recognition*, 41(5):1440–1451.

Zhang, F. (2005). *The Schur Complement and Its Applications*. Springer.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2003). Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pages 321–328.

Zhou, Y. and Goldman, S. A. (2004). Democratic co-learning. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 594–602.

Zhou, Z. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919. AAAI Press.

Zhu, X., Goldberg, A. B., Brachman, R., and Dietterich, T. (2009). *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers.

Zucker, J. and Ganascia, J. (1996). Representation changes for efficient learning in structural domains. In Saitta, L., editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 543–551. Morgan Kaufmann.

Symbols

$\mathcal{G}$	The graph.
V	The set of vertices.
E	The set of edges.
$\mathcal{Q}$	The set of query atoms.
$\mathcal{E}$	The set of evidence atoms.
$\mathcal{F}$	The set of random binary (Half-Space) trees.
$\mathcal{B}$	The Herbrand base.
$\mathcal{M}$	The set of mode declarations.
$\mathcal{A}$	The set of logical atoms constructed from a Herbrand base $\mathcal{B}$ and a set of mode declarations $\mathcal{M}$.
$\|\mathcal{S}\|$	The cardinality of set $\mathcal{S}$.
$\mathbf{C}$	The cost matrix.
$\mathbf{W}$	The distance matrix.
w_{ij}	The edge weight (distance) connecting vertices i and j.
$\mathbf{D}$	The degree matrix.
$\mathbf{L}$	The Laplacian matrix.
$\mathbf{M}$	The Mahalanobis distance covariance matrix.
$\mathcal{N}_i^k$	The set of target k-nearest neighbors of an example i.
$\mathbf{b}$	The binary feature vector.
$v_i^{\mathbf{b}}$	The vertex i having only the logical atoms indicated by the feature vector $\mathbf{b}$.
H_p	The hierarchical partitioning p.
$\mathcal{H}(\mathcal{D})$	The set of all partitionings over a dataset $\mathcal{D}$.
$R(x, y\|H_p)$	The smallest region covering x and y in the partitioning H_p.

$\mathcal{C}$	The cache containing mappings of vertices to their counts.
d_s	The structural distance (SPLICE).
$d_s^{\mathbf{b}}$	Optimised (LMFS) structural distance.
$\tilde{m}$	Mass-based dissimilarity.
$d_h^{\mathbf{b}}$	Optimised (LMFS) hybrid distance (SPLICE$^+$).
d_h	Non-optimised hybrid distance.
$\neg$	Negation.
$\wedge$	Conjunction.
$\Leftarrow$	Implication.

“Everything not saved will be lost”
— Nintendo “Quit Screen” message