



Online semi-supervised learning of composite event rules by combining structure and mass-based predicate similarity

Evangelos Michelioudakis^{1,2} · Alexander Artikis^{1,3} · Georgios Paliouras¹

Received: 14 June 2022 / Revised: 24 July 2023 / Accepted: 7 October 2023
© The Author(s) 2023

Abstract

Symbolic event recognition systems detect event occurrences using first-order logic rules. Although existing online structure learning approaches ease the discovery of such rules in noisy data streams, they assume the existence of fully labelled training data. SPLICE is a recent online graph-based approach that estimates the labels of unlabelled data and makes it possible to learn such rules from semi-supervised training sequences of logical interpretations. However, SPLICE labelling depends significantly on the metric used to compute the distances of unlabelled examples to their labelled counterparts. Moreover, there is no guarantee about the quality of the labelling found in the local graphs that are built as the data stream in. In this paper, we propose a new online learning method, which includes an enhanced hybrid measure that combines an optimised structural distance, and a data-driven one. The former is guided by feature selection targeted to k NN classification, while the latter is a mass-based dissimilarity. Additionally, the enhanced SPLICE method, 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 by learning Event Calculus theories for the tasks of human activity recognition, maritime monitoring, and fleet management. The evaluation suggests that our approach outperforms its predecessor, in terms of inferring the missing labels and improving the predictive accuracy of the underlying structure learning system.

Keywords First-order logic · Metric learning · Mass dissimilarity · Event calculus · Event recognition

1 Introduction

Symbolic *composite event recognition* (CER) systems (Cugola & Margara, 2012) consume input sequences of *simple, derived events* (SDEs), matching them against a knowledge base of first-order rules (Artikis et al., 2012), and recognising *composite events* (CEs)

Editors: Alireza Tamaddon-Nezhad, Alan Bundy, Luc De Raedt, Artur d'Avila Garcez, Sebastijan Dumančić, Cèsar Ferri, Pascal Hitzler, Nikos Katzouris, Denis Mareschal, Stephen Muggleton, Ute Schmid

Extended author information available on the last page of the article

of interest. CEs are usually defined as multi-relational structures over actors and objects involved in an event, and thus, manual derivation of such rules can be cumbersome and error-prone. In addition, CER applications typically operate in data streams of significant volume and velocity (Giatrakos et al., 2020), which further renders the composition of such relational dependencies unrealistic. To that end, methods for learning CE rules in a single pass over a data stream are essential (Srinivasan & Bain, 2017; Dries & Raedt, 2009; Gama, 2010).

Online logic-based learners have been proposed for the discovery of CE structures under uncertainty (Katzouris et al., 2016, 2018; Michelioudakis et al., 2016). They assume that a fully-labelled training sequence arrives for processing, which is an unrealistic assumption in real-life applications.

SPLICE (Michelioudakis et al., 2019) is a recent method that makes semi-supervised learning of logic-based CE rules possible by inferring the missing labels using a graph-based label propagation technique (Zhu et al., 2003). SPLICE represents training instances as sets of logical atoms and employs a structural distance, adapted from Nienhuys-Cheng (1997), to compute the distance of unlabelled data to their labelled counterparts. The labelling is achieved online (single-pass), by storing previously seen labels for future usage, since it is assumed that the labelled data are infrequent. Although SPLICE facilitates the learning of CE rules in the presence of missing labels, its distance measure may be compromised by irrelevant features or imbalanced supervision. Moreover, its approach to online label propagation does not provide any guarantee about the labelling inferred from the local graphs, which are built as the data stream in. In other words, SPLICE ignores previously seen unlabelled examples and their respective distances to the rest of the graph.

In this paper, 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 optimising feature selection for k -nearest neighbour (k NN) classification. To that end, we adapt the Large-Margin Nearest Neighbour (Weinberger & Saul, 2009), a well-known approach to metric learning, for the selection of logical predicates. Finally, in order to provide guarantees about the online labelling, we utilize 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.

The completed training data can be subsequently used by any online supervised structure learner. To demonstrate SPLICE⁺, our proposed approach for semi-supervised learning of event rules, we use the OLED online learner (Katzouris et al., 2016), which constructs Event Calculus (EC) theories (Kowalski & Sergot, 1986; Mueller, 2008) for CER applications.

The proposed method (SPLICE⁺) is compared to its predecessor (SPLICE) and another baseline method adapted from Soonthornphisaj and Kijisirikul (2004) to learn EC theories on the tasks of activity recognition from surveillance video footage, maritime monitoring, and fleet management. In the first task, the goal is to recognise activities taking place between persons, e.g., people meeting or moving together, by exploiting information about observed activities of individuals. In maritime monitoring, the goal is to recognise vessel activities, by exploiting information such as vessel speed, location and communication gaps. Finally, in fleet management, the objective is to recognise vehicle activities, by combining positional and operational information such as vehicle speed, location and fuel level. Our empirical analysis suggests that our improved method outperforms both its predecessor and the baseline. In particular, it infers more accurately the missing labels and

improves the predictive accuracy of the underlying structure learner, at the price of a tolerable increase in processing time.

In summary, the main contributions of this paper are:

1. An adapted metric learning technique for feature subset 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.
2. A hybrid distance measure which combines the above-mentioned informed structural distance to a mass-based dissimilarity. The hybrid measure exploits both the labelled and unlabelled data to quantify the distance between examples of logical atoms. The resulted measure can be considered as a semi-supervised metric learning method.
3. An online semi-supervised logic-based learner that retains a graph synopsis of temporally adjacent examples, in order to operate on large training sequences, and learns interpretable CE rules in the Event Calculus.
4. The evaluation of SPICE^+ against SPICE on three real (non-synthetic) datasets. A human activity recognition dataset labelled by human experts; a maritime monitoring dataset comprising vessel position signals; and a fleet management dataset consisting of positional and operational data from commercial vehicles. Moreover, to aid research reproducibility, we have made the datasets and the code publicly available.

The remainder of the paper is organised as follows. Section 2 provides the necessary background used in Sect. 3 to describe the proposed hybrid predicate similarity and our online semi-supervised learning system for learning composite event rules. Section 4 reports experimental results on the tasks of human activity recognition, maritime monitoring and vehicle fleet management. Section 5 discusses related work for semi-supervised learning on data streams, feature selection and metric learning techniques, while Sect. 6 concludes and proposes directions for future research.

2 Background

2.1 Online semi-supervised learning for composite event recognition

SPICE (Michelioudakis et al., 2019) enables online structure learning for composite event recognition in the presence of incomplete supervision. Towards this end, it employs a graph-based technique proposed by Zhu et al. (2003), to derive labels for the unlabelled data, based on their distance to their labelled counterparts. Since the CEs are usually defined over multi-relational data (Cugola & Margara, 2012), instead of numerical data points, SPICE employs a distance function for sets of logical atoms, which represent the training examples. The labelling process is achieved in a single-pass over the training data by storing a compressed version of previously seen labelled examples.

SPICE assumes that a training sequence arrives for processing in micro-batches. Each micro-batch contains a sequence of ground evidence atoms, i.e., first-order logic ground atoms, expressed in the Event Calculus (Kowalski & Sergot, 1986). Each micro-batch can be fully labelled, partially labelled or may contain no labels at all. Each micro-batch is grouped into *examples* of ground atoms, as depicted in Fig. 1. The HappensAt and HoldsAt atoms denote simple derived event (SDE) and CE occurrences respectively, while

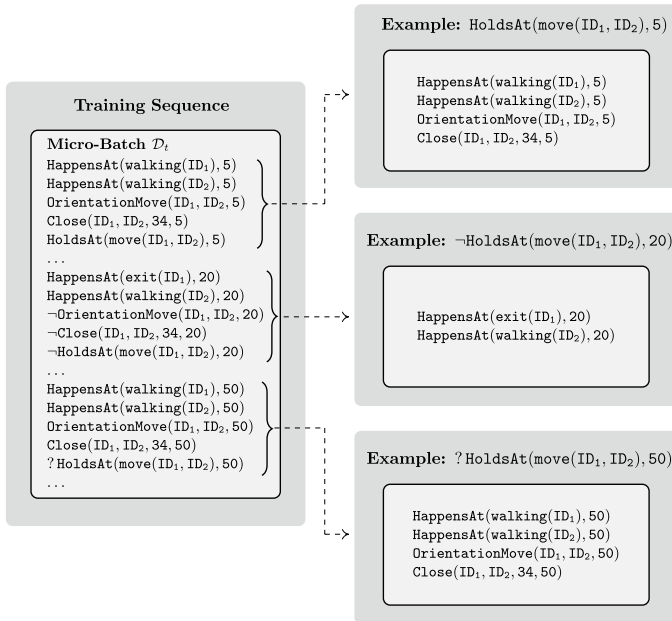


Fig. 1 Examples contain a ground query atom, labelled or unlabelled, and a set of ground evidence atoms that are linked to the query atom through their constants

the remaining atoms express contextual information. The micro-batch size is user-defined and it is measured in time-points. In general the size can be arbitrarily large, however, in practice, small micro-batches improve runtime performance. SPLICE assumes that CEs appear chronologically immediately following the pattern of SDEs causing them. Grouped examples contain exactly one ground query atom and a subset of the evidence atoms in the micro-batch. The selected evidence atoms are linked to the query atom through their shared constants, i.e., they should be relevant to the CE of interest. The top example in Fig. 1 is labelled, stating that two persons are moving together, while the bottom one is unlabelled. Unlabelled atoms are prefixed by the symbol ‘?’.

In order to address the online processing requirement, under the assumption that labels are infrequent, SPLICE caches previously seen labelled examples for future usage. The caching mechanism only stores unique examples, along with their frequency (the number of times they have appeared in the stream), in order to reduce the memory requirements. To that end, SPLICE employs logical unification in order to check for uniqueness. The stored labelled examples, together with the unlabelled examples of the micro-batch, compose the vertices of the graph used in the subsequent steps.

Once the vertices have been collected, they are connected by edges representing the structural similarity of their underlying evidence atoms. The structural similarity is adapted from Nienhuys-Cheng (1997), by replacing the Hausdorff metric by the Kuhn–Munkres algorithm Kuhn (1955). Let a pair of vertices $v_i = \{e_{i1}, \dots, e_{iM}\}$ and $v_j = \{e_{j1}, \dots, e_{jK}\}$ consisting of M and K evidence atoms respectively (let $M > K$). SPLICE first computes the structural distance (Nienhuys-Cheng (1997), Definition 4) between each pair of evidence atoms $d(e_{im}, e_{jk})$, where $m \in \{1, \dots, M\}$ and $k \in \{1, \dots, K\}$ resulting in a $M \times K$ distance matrix \mathbf{D} . For instance, given the top and bottom examples of Fig. 1, the distance between evidence

atoms $e_{top} = \text{HappensAt}(\text{walking}(\text{ID}_1), 5)$ and $e_{bottom} = \text{HappensAt}(\text{walking}(\text{ID}_2), 50)$ is computed as follow:

$$\begin{aligned} d(e_{top}, e_{bottom}) &= \frac{1}{2p} \sum_{i=1}^p d(e_{top,p}, e_{bottom,p}) \\ &= \frac{1}{2 \cdot 2} \left(d(\text{walking}(\text{ID}_1), \text{walking}(\text{ID}_2)) + d(5, 50) \right) \\ &= \frac{1}{4} \left(\frac{1}{2 \cdot 1} d(\text{ID}_1, \text{ID}_2) + 1 \right) = \frac{1}{4} \left(\frac{1}{2} \cdot 1 + 1 \right) = 0.375 \end{aligned}$$

where p is the term position. The matrix \mathbf{D} is then given as an input cost matrix to the Kuhn-Munkres algorithm, in order to find the optimal mapping of evidence atoms. The optimal mapping is denoted here by the function $g : V \times V \mapsto \{(m, k) : m, k \in \{1, \dots, K\}\}$ 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 (M in this case) of the matrix:

$$d_s(v_i, v_j) = \frac{1}{M} \left[(M - K) + \sum_{(m,k) \in g(v_i, v_j)} \mathbf{D}_{mk} \right] \quad (1)$$

where $M - K$ penalises all unmatched evidence atoms by the greatest possible distance, which is 1. Thus, $M - K$ can be seen as a regularisation term. The distance is turned into a similarity $s(v_i, v_j) = 1 - d_s(v_i, v_j)$, yielding a similarity matrix \mathbf{W} . Then, a k NN filter is applied on \mathbf{W} in order to retain only edges between very similar vertices. The resulting graph is used to derive labels for all unlabelled examples in the micro-batch, by obtaining the harmonic solution to the optimisation problem defined by Zhu et al. (2003).

Each labelled micro-batch is forwarded to a logic-based structure learner, such as OLED (Katzouris et al., 2016, 2018), in order to induce new or enhance existing CE rules. The process is repeated for each micro-batch as it arrives.

Although SPLICE facilitates the automated discovery of CE rules in the presence of incomplete supervision, its online procedure and distance function have limitations. The performance of SPLICE is compromised by the presence of irrelevant features, because the distance function is agnostic to the feature semantics. Moreover, it 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 worth noting that, in the case of true streaming (one example per micro-batch), the optimisation reduces to k NN classification (Chapelle et al., 2006).

2.2 Large-margin nearest neighbour metric learning

Graph-based methods to semi-supervised learning rely on the *cluster assumption*, that is, similar examples should yield the same labelling. Thus, the distance function constitutes a key component of these methods and in fact controls the quality of the labelling. A common issue of most distance measures is that they are agnostic to the semantics of input features. As a result, their measurements may suffer in the presence of irrelevant or noisy features.

Large-margin nearest neighbour (LMNN) Weinberger and Saul (2009) is a metric learning technique that learns a distance pseudo-metric targeted to k NN classification. Intuitively, LMNN attempts to increase the number of examples in a neighbourhood that share the same label, by learning a transformation of the input feature space using the Mahalanobis distance:

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^{\top} \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j) \quad (2)$$

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

To that end, 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 instances that share the same label and should be nearest neighbours. In terms of the transformation in the input space, the sum of these squared distances is given by

$$\varepsilon_{pull}(\mathbf{M}) = \sum_{i,j \in \mathcal{N}_i^k} d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \quad (3)$$

where \mathcal{N}_i^k denotes a set of *target k -nearest neighbours* for 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*. 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:

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l) \leq d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + 1 \quad (4)$$

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 \mathbf{x}_i . Hence, the second term penalises violations of the above inequality as follows:

$$\varepsilon_{push}(\mathbf{M}) = \sum_{i,j \in \mathcal{N}_i^k} \sum_l (1 - y_{il}) [1 + d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l)]_+, \quad (5)$$

where the indicator variable $y_{il} = 1$ if and only if $y_i = y_l$, and $y_{il} = 0$ otherwise. Moreover, $[z]_+ = \max(z, 0)$ denotes the standard hinge loss, which monitors inequality (4). If the inequality does not hold (i.e., the input \mathbf{x}_l lies a safe distance away from \mathbf{x}_i), then its hinge loss has a negative argument and doesn't contribute to the overall loss. The combined loss derived from Eq. (3) and Eq. (5) is as follows:

$$\varepsilon(\mathbf{M}) = (1 - \mu) \varepsilon_{pull}(\mathbf{M}) + \mu \varepsilon_{push}(\mathbf{M}), \quad (6)$$

where the weighting parameter $\mu \in [0, 1]$ balances the two goals. Figure 2 illustrates the idea behind LMNN optimisation. 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 derivation 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 p th feature. Thus, the loss function depicted in Eq. (6) becomes:

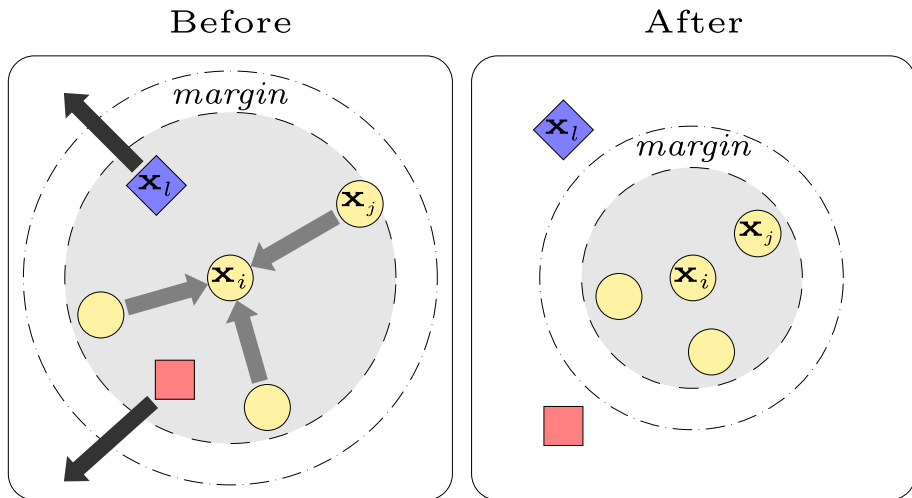


Fig. 2 The neighbourhood of \mathbf{x}_i before and after optimisation. A distance metric is learned so that: (i) target neighbours (yellow circles) lie in a small radius from \mathbf{x}_i ; (ii) impostors (blue diamond, red square) lie outside this smaller radius by a finite margin. Arrows indicate pull and push operations respectively (after Weinberger and Saul (2009)) (Color figure online)

$$\varepsilon(\mathbf{m}) = (1 - \mu) \sum_{i,j \in \mathcal{N}_i^k} d_{\mathbf{m}}(\mathbf{x}_i, \mathbf{x}_j) + \mu \sum_{i,j \in \mathcal{N}_i^k} \sum_l (1 - y_{il}) [1 + d_{\mathbf{m}}(\mathbf{x}_i, \mathbf{x}_j) - d_{\mathbf{m}}(\mathbf{x}_i, \mathbf{x}_l)]_+, \quad (7)$$

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

$$\begin{aligned} &\text{minimise} \quad (1 - \mu) \sum_{i,j \in \mathcal{N}_i^k} \|\mathbf{m}(\mathbf{x}_i - \mathbf{x}_j)\|^2 + \mu \sum_{i,j \in \mathcal{N}_i^k} \sum_l (1 - y_{il}) \xi_{ijl} \\ &\text{subject to} \quad (1) \|\mathbf{m}(\mathbf{x}_i - \mathbf{x}_l)\|^2 - \|\mathbf{m}(\mathbf{x}_i - \mathbf{x}_j)\|^2 \geq 1 - \xi_{ijl} \quad (8) \\ &\quad \quad \quad (2) \xi_{ijl} \geq 0 \\ &\quad \quad \quad (3) \mathbf{m} \geq 0 \end{aligned}$$

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. (4) is violated. The resulting weights \mathbf{m} capture the importance of each input feature and is adapted in our proposed method to perform feature subset selection in order to exclude irrelevant or noisy logic atoms from the similarity measurements of SPLICE⁺.

2.3 Mass-based dissimilarity

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

during the optimisation. In order to improve feature selection, we attempt to combine LMNN to a form of unsupervised metric learning.

A mass-based dissimilarity, proposed by Ting et al. (2019), employs estimates of the probability mass to quantify the dissimilarity of two points. Mass dissimilarity measurements mainly depend on the distribution of the data. The intuition is that the dissimilarity of two points depends on the amount of probability mass in the region 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.

Let H_p denote a hierarchical partitioning of a space \mathbb{R}^n into a set of non-overlapping (bottom-level) regions that collectively span \mathbb{R}^n . 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 partitionings H_p that are admissible under a data set D , such that each non-overlapping region contains at least one point from D . Then the smallest region covering a pair of points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ with respect to a hierarchical partitioning model H_p of \mathbb{R}^n is defined as:

$$R(\mathbf{x}, \mathbf{y} \mid H_p) = \operatorname{argmax}_{r \in H_p \text{ s.t. } \{\mathbf{x}, \mathbf{y}\} \in r} \operatorname{depth}(r; H_p),$$

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

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 smallest region $R(\mathbf{x}, \mathbf{y} \mid H_p)$:

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

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 partitionings $H_p \in \mathcal{H}(D)$, $p = 1, \dots, T$ as follows:

$$\tilde{m}(\mathbf{x}, \mathbf{y} \mid D) = \frac{1}{T} \sum_{p=1}^T \tilde{P}(R(\mathbf{x}, \mathbf{y} \mid H_p; D)), \quad (9)$$

where $\tilde{P}(R) = \frac{1}{|D|} \sum_{\mathbf{z} \in D} \mathbb{1}(\mathbf{z} \in R)$ estimates the probability of the smallest 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 partitionings H_p , a recursive partitioning scheme is employed based on the concept of the *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 partitions the data at that node into two non-empty subsets. The process is repeated recursively until either every data point is isolated, that is all regions contain a single point, or a given maximum tree depth is reached.

Subsequently the resulting Isolation Forest can be used to compute the mass-based dissimilarity of Eq. (9). Since each Isolation Tree essentially represents a partitioning H_p , the mass-based dissimilarity can be defined as:

$$\tilde{m}(\mathbf{x}, \mathbf{y}) = \frac{1}{T} \sum_{p=1}^T \frac{|R(\mathbf{x}, \mathbf{y} \mid H_p)|}{|D|}, \quad (10)$$

where $\frac{|R(\mathbf{x}, \mathbf{y} \mid H_p)|}{|D|}$ estimates the probability of region R , as denoted by $\tilde{P}(R)$ in Eq. (9). To compute Eq. (10), \mathbf{x} and \mathbf{y} are passed through each tree to find the mass of the deepest node, containing both \mathbf{x} and \mathbf{y} i.e., $\sum_p |R(\mathbf{x}, \mathbf{y} \mid H_p)|$. Finally, \tilde{m} is the mean of these masses over the T trees.

2.4 Temporal label propagation

Traditionally, graph-based methods to semi-supervised learning (Zhu et al., 2009) assume that all labelled and unlabelled data are stored in memory and thus are available during the optimisation (label propagation) that yields the harmonic solution. However, that is an unrealistic assumption in online processing of data streams. SPLICE as presented in Sect. 2.1, relaxes this assumption by storing previously seen labelled examples and reusing them in subsequent micro-batches. Nevertheless, it still ignores previously seen unlabelled examples and by extension their respective distances to the rest of the graph. Therefore, SPLICE cannot guarantee that the global harmonic solution obtained by label propagation on the entire graph is preserved on the local graph of the micro-batch.

Temporal Label Propagation (TLP) (Wagner et al., 2018) is a recent approach to label propagation for fast-moving data streams. TLP stores a synopsis of the full history of the stream, in order to retain information about the labelled and the unlabelled examples seen so far and incorporate it into the subsequent optimisations. To that end, TLP allows for a weighted 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 $\mathcal{G}(V_\tau)$ 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. Wagner et al. (2018) proved that this property allows for the harmonic solution to be preserved in the synopsis graph.

The Laplacian matrix of $\mathcal{G}(V_\tau)$, required to obtain the harmonic solution, is given by the *Schur Complement* Dörfler and Bullo (2013). Since computing the Schur 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, TLP operates in an online fashion and computes $\mathcal{G}(V_\tau)$ as a sequence of local operations, called *star-mesh transformations*. This is a direct consequence of the sequential property of the Schur complement (Zhang (2005), Theorem 4.10; Dörfler and Bullo (2013), Lemma III.1).

Definition 1 A star-mesh transformation on a vertex v_o of a weighted 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 $w_{v,v'} = \frac{w_{v,v_o} w_{v_o,v'}}{\text{degree}(v_o)}$. If (v, v') is already in E , then add the new weight $w_{v,v'}$ to its current weight.

The intuition of TLP 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 labelling, 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 $\mathcal{G}\langle V_{\tau} \rangle$ storing the τ more recent unlabelled examples, in addition to a pair of labelled node clusters, each holding all the positive and negative examples seen so far. When a new unlabelled example arrives, TLP appends it to $\mathcal{G}\langle V_{\tau} \rangle$ and if the memory τ has been exceeded, it removes the oldest unlabelled example by applying the star-mesh transform of Definition 1. In the simplest case, where a labelled example arrives, TLP just appends it to the appropriate cluster node, thus always maintaining $\tau + 2$ nodes. 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 stream seen so far.

3 Online semi-supervised learning combining structure and mass-based predicate similarity

SPLICE, as presented in Sect. 2.1, aims to effectively learn the structure of composite event rules in the presence of incomplete supervision. However, we have identified a number of issues in its graph construction process, that may compromise the online labelling of the unlabelled data. First, the underlying structural distance is sensitive to the presence of irrelevant or noisy features. Second, the distance measurements between labelled and unlabelled data, inevitably, are as informative as the provided labels. 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 per micro-batch, provides no guarantee with respect to the global solution obtained if all data were to be accessed at once.

The new method presented here, called SPLICE⁺, improves the quality of the graph construction component, leading to more robust and accurate labelling of the incoming unlabelled data. An overview of the components of SPLICE⁺ is shown in Fig. 3. We present a hybrid distance measure composed of two elementary distances, that overcomes the drawbacks of the structural distance of SPLICE. The first of the two distance components is an enhanced version of Eq. (1) that accounts for irrelevant or noisy features by selecting only a subset of them, that is, the ones optimising k NN classification on 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 combine the optimised structural distance with a data-driven mass-based dissimilarity, adapted to logical atoms. This dissimilarity samples the space of logical structures, and employs mass estimation theory to compute the relative distance between examples, measured as the probability density of their least general generalisation (Plotkin, 1971).

In order to render SPLICE aware of the temporal nature of the data in CER, we further alter its strategy for interconnecting graph vertices. We connect each unlabelled vertex to its k -nearest labelled neighbours, as well as, to the temporally preceding unlabelled vertex. This way we promote interactions between temporally adjacent unlabelled vertices during label propagation. Finally, we store a synopsis of the full history of the stream, by means of a short-circuit operator, which preserves the effective distances of labelled and unlabelled example vertices to be used in subsequent optimisations. The proposed improvements introduced in SPLICE⁺

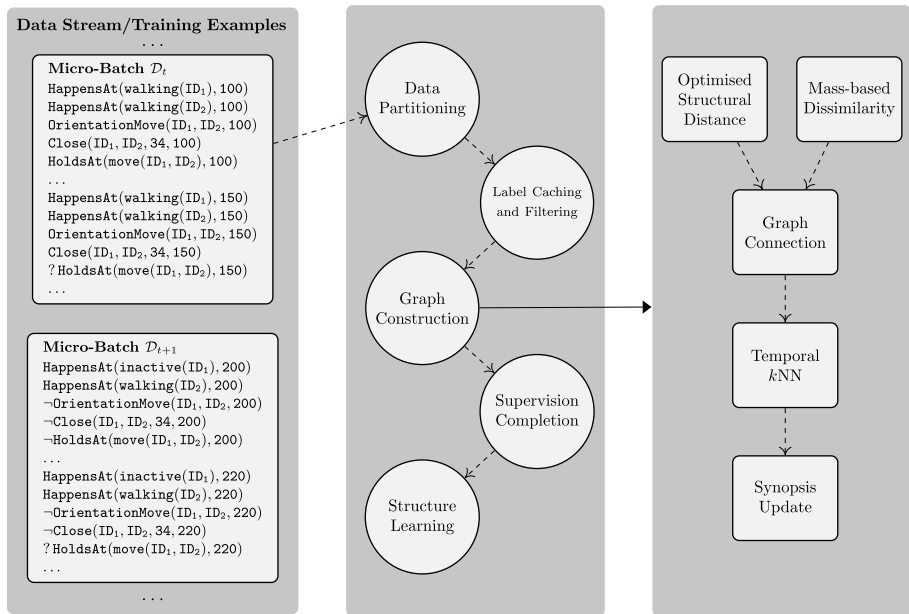


Fig. 3 The SPLICE⁺ procedure

are detailed in the following subsections. To aid the presentation, we employ examples from human activity recognition in video recordings.

3.1 Large-margin feature selection for logical structures

In order to render the structural distance of Eq. (1) aware of irrelevant or noisy features, we introduce a mechanism for feature selection based on the approach of LMNN metric learning. We adapt the idea of feature weighting, as presented in Sect. 2.2, to learn a binary vector, instead of real-value one, representing the set of selected logical atoms that should be used for computing distances. Towards that goal, we use a similar approach to propositionalization (Zucker & Ganasia, 1996; Alphonse & Matwin, 2002). Let \mathcal{A} be a set of first-order atoms that can be constructed from a Herbrand base \mathcal{B} and a set of mode declarations \mathcal{M} , by replacing constants with variables in \mathcal{B} . Assuming a strict ordering of atoms in \mathcal{A} , let \mathbf{b} be a vector of binary variables, one for each first-order atom $a_i \in \mathcal{A}$. Thus, each indicator variable $b_i = 1$ if the i th atom is selected, and $b_i = 0$ otherwise. Since each labelled training example is essentially a clause c , it can also be represented by a binary vector $\mathbf{x}_c = [x_1, \dots, x_{|\mathcal{A}|}]^T$, where each variable x_i refers to the presence of the corresponding atom a_i from \mathcal{A} in clause c . For instance, assuming that \mathcal{B} contains the ground atoms appearing in Fig. 1, we can create an ordered set of atoms as follows:

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

The top example from Fig. 1 is represented as $\mathbf{x}_{top} = [1, 1, 0, 1, 1]$, the middle one as $\mathbf{x}_{mid} = [0, 1, 1, 0, 0]$ and the bottom one as $\mathbf{x}_{bot} = [1, 1, 0, 1, 1]$. The dissimilarity of two such examples can be measured by a simple 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. (7) as follows:

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

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 programming problem and can be solved using variants of the branch-and-bound or branch-and-cut methods (Williams, 2009), albeit less efficiently, since it is NP-hard, than the real-valued problem¹:

$$\begin{aligned} & \text{minimise} && \sum_{i,j \in \mathcal{N}_i^k} \sum_l (1 - y_{il}) \xi_{ijl} \\ & \text{subject to} && \text{(1) } \mathbf{b} \mid \mathbf{x}_i - \mathbf{x}_l \mid - \mathbf{b} \mid \mathbf{x}_i - \mathbf{x}_j \mid \geq 1 - \xi_{ijl} \\ & && \text{(2) } \mathbf{b} \mathbf{x}_i \geq 1 \\ & && \text{(3) } \xi_{ijl} \in \mathbb{N}^{\geq} \\ & && \text{(4) } \mathbf{b} \in \{0, 1\}^{|\mathcal{A}|} \end{aligned} \quad (11)$$

The intuition of our proposed for feature subset selection, called Large-Margin Feature Selection (LMFS), is to keep the minimal set of logical atoms (features) that are necessary to discriminate a 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 examples to have at least one selected logical atom. This constraint is necessary to avoid degenerate solutions that remove all atoms yielding empty examples.

Given the optimal vector \mathbf{b} , we can generalise all future examples by removing features for which $b_i = 0$. For instance, if the optimal vector is $\mathbf{b}^* = [1, 1, 1, 0, 0]$, then the top example would become $\mathbf{v}_{top}^{\mathbf{b}} = \mathbf{b}^* \mathbf{x}_{top} = [0, 0, 0, 1, 0] = \{\text{HappensAt}(\text{walking}(x), t), \text{HappensAt}(\text{walking}(y), t)\}$, while the middle example would become $\mathbf{v}_{mid}^{\mathbf{b}} = \mathbf{b}^* \mathbf{x}_{mid} = [0, 0, 1, 0, 0] = \{\text{HappensAt}(\text{exit}(x), t)\}$. Then, the structural distance can be computed as usual, by applying Eq. (1) on the generalised examples:

$$d_s^{\mathbf{b}}(v_i, v_j) = d_s(v_i^{\mathbf{b}}, v_j^{\mathbf{b}}), \quad (12)$$

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. (11) can be solved fast enough.

Table 1 Distance measures notation

Symbol	Description
d_s	Structural distance (SPLICE)
d_s^b	Optimised (LMFS) structural distance
\tilde{m}	Mass-based dissimilarity
d_h^b	Optimised (LMFS) hybrid distance (SPLICE ⁺)
d_h	Non-optimised hybrid distance

where v_i, v_j are examples and v_i^b, v_j^b their generalised counterparts, where some first-order atoms have been removed. The distance measures notation is summarised in Table 1. One issue that may arise from selecting features using only the labelled examples is that some atoms may appear only in unlabelled examples, and thus, 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 training examples to be accompanied by labels, which in our case leads to the selection of features that discriminate between positive and negative labelled examples. However, in a Hamming space distances change quite abruptly because a single mismatch between two binary vectors always yields a penalty of 1 between the vectors. Therefore, while in a 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 since the optimisation would try to force them to become similar by removing atoms that cause mismatches. To avoid such situations, similar to Deng and Luo (2015), 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 be on the same cluster under θ -subsumption. Consider the top and middle examples of Fig. 1. They should form a pair of unit clusters, since they belong to opposite classes. If the bottom example was also positive, then it should 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. The examples inside a specific cluster express more general versions of the same concept (under θ -subsumption), while examples between clusters of the same class express alternative definitions of the concept. Thus the push constraint of Eq. (11) enforces alternative definitions to differ in at least one atom. Given such a clustering, the optimisation of Eq. (11) should select features that respect that partitioning, identifying which features are necessary for discriminating each sub-concept (Deng & Luo, 2015).

Algorithm 1 presents the pseudo-code for selecting the first-order atoms that best discriminate the known labelled examples into sub-concepts. The algorithm requires as input a set of labelled examples, a set of mode declarations, and produces a vector of selected

Input: \mathcal{V}_L : a set of labelled examples, \mathcal{M} : a set of mode declarations

Output: \mathbf{b} : a vector of binary values corresponding to selected features

- 1: Partition \mathcal{V}_L into positive \mathcal{V}_P and negative \mathcal{V}_N examples.
 - 2: Find $v_i^{\max} = \operatorname{argmax}_{v_i \in \mathcal{V}_P} \mathbf{v} \cdot \mathbf{i}$ and $v_j^{\max} = \operatorname{argmax}_{v_j \in \mathcal{V}_N} \mathbf{v} \cdot \mathbf{j}$.
 - 3: Form unit clusters $C = \{\{v_i^{\max}\}, \{v_j^{\max}\}\}$.
 - 4: **for** $v_i \in \mathcal{V}_L \setminus C$ **do**
 - 5: **for** $c \in C$ **do**
 - 6: **if** $\exists v' \in c : \text{clause}(v_i)\theta \subseteq \text{clause}(v')$ **then**
 - 7: $c = c \cup v_i$
 - 8: **end if**
 - 9: **end for**
 - 10: **end for**
 - 11: Solve optimisation of Eq. (11) using C as a set of examples.
 - 12: **return** \mathbf{b}
-

Algorithm 1 LMFS $\mathcal{V}_L, \mathcal{M}$

features. It starts by partitioning the given examples into positive and negative (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 with these examples (lines 2–3). For each of the remaining examples it either appends 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 of Eq. (11) using the clusters as classes and returns the vector of selected features (lines 8–9).

3.2 Mass dissimilarity for logical structures

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 distance, as presented in Sect. 3.1, 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 the previous section, the proposed approach exploits both labelled and unlabelled data to quantify the distances between examples of interest.

To that end, we adapt the approach presented in Sect. 2.3 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 Sect. 3.1). Since the space of 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 (Ting et al., 2013). In contrast to Ting et al. (2019), which assumes real-valued features, we can construct the trees beforehand because each internal node of the tree can only have one possible split, since atoms are binary. Algorithm 2 presents the pseudo-code for creating a forest of binary random trees.

Input: \mathcal{A} : a set of first-order atoms, T : number of trees, h : maximum tree height

Output: \mathcal{F} : a half-space forest (set of half-space trees)

```

1:  $\mathcal{F} = \emptyset$ 
2: for  $i = 1$  to  $T$  do
3:    $\mathcal{F} = \mathcal{F} \cup \text{CREATETREE}(\mathcal{A}, 0, h)$ 
4: end for
5: return  $\mathcal{F}$ 
6:
7: function  $\text{CREATETREE}(\mathcal{A}, d, h)$   $\triangleright d$  is the current depth of the tree
8:   if  $d > h \vee |\mathcal{A}| < 1$  then
9:     return  $\text{Node}(\text{size} \leftarrow 0, \text{split} \leftarrow \emptyset, \text{left} \leftarrow \emptyset, \text{right} \leftarrow \emptyset)$ 
10:  else
11:    Randomly select an atom  $a \in \mathcal{A}$ .
12:    return  $\text{Node}(\text{size} \leftarrow 0, \text{split} \leftarrow a,$ 
13:       $\text{left} \leftarrow \text{CREATETREE}(\mathcal{A} \setminus a, d+1, h),$ 
14:       $\text{right} \leftarrow \text{CREATETREE}(\mathcal{A} \setminus a, d+1, h))$ 
15:  end if
16: end function

```

Algorithm 2 $\text{CREATEFOREST } \mathcal{A}, T, h$

The algorithm requires as input a set of first-order atoms, a 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.

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 3 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 increments the counts of its nodes. Updating the size of single tree, given a single example, is equivalent to traversing a single path from the root to a leaf in a binary tree. Thus, the runtime complexity of updating the entire forest is $\mathcal{O}(|\mathcal{F}||\mathcal{V}| \log(h))$, where h is the height of the tree and trees are height-balanced.

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 depicts a part of the subsumption lattice constructed from the atoms appearing in the training sequence of Fig. 3. The top of the lattice represents the empty clause, a rule having no literals, while the bottom

Input: \mathcal{F} : a set of half-space trees, \mathcal{V} : a set of examples

```

1: for each tree  $\in \mathcal{F}$  and  $v \in \mathcal{V}$  do
2:   UPDATE_SIZE(tree,  $v$ )
3: end for
4:
5: function UPDATE_SIZE(tree,  $v$ )
6:   tree.size  $\leftarrow$  tree.size + 1
7:   if tree.left  $\neq \emptyset \wedge$  tree.split  $\notin v$  then UPDATE_SIZE(tree.left,  $v$ )
8:   else if tree.right  $\neq \emptyset \wedge$  tree.split  $\in v$  then UPDATE_SIZE(tree.right,
    $v \setminus \text{split}$ )
9: end function

```

Algorithm 3 UPDATEFOREST \mathcal{F}, \mathcal{V}

represents the bottom clause, a rule containing all possible literals. Note that the bottom clause is a theoretical concept and it is never actually constructed. 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 size of each node. In this case the sizes correspond to the three examples of Fig. 1. 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 Fig. 1. Their set of common atoms is just the atom `HappensAt(walking(x), t)`, which, in the tree appearing in Fig. 4, 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. Note that the dissimilarity estimates the mass of the least general generalisation (*lgg*) of the examples, as defined by Plotkin (1971), which represents the least general rule that covers both examples. If the mass of their *lgg* is high then the rule is very common and covers a lot of examples which indicates that the rule is not very interesting.

The resulting Half-Space Forest can be used to compute the dissimilarity of Eq. (10) for a pair of examples as follows:

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

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 Sect. 2.3, is the deepest region covering both examples. Thus, given the tree presented in Fig. 4, the top and middle examples of Fig. 1 would have dissimilarity $\tilde{m}(v_{top}, v_{middle}) = 1$, since $|R(v_{top}, v_{middle} \mid H_1)| = 3$, $|D| = 3$ and $T = 1$.

3.3 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

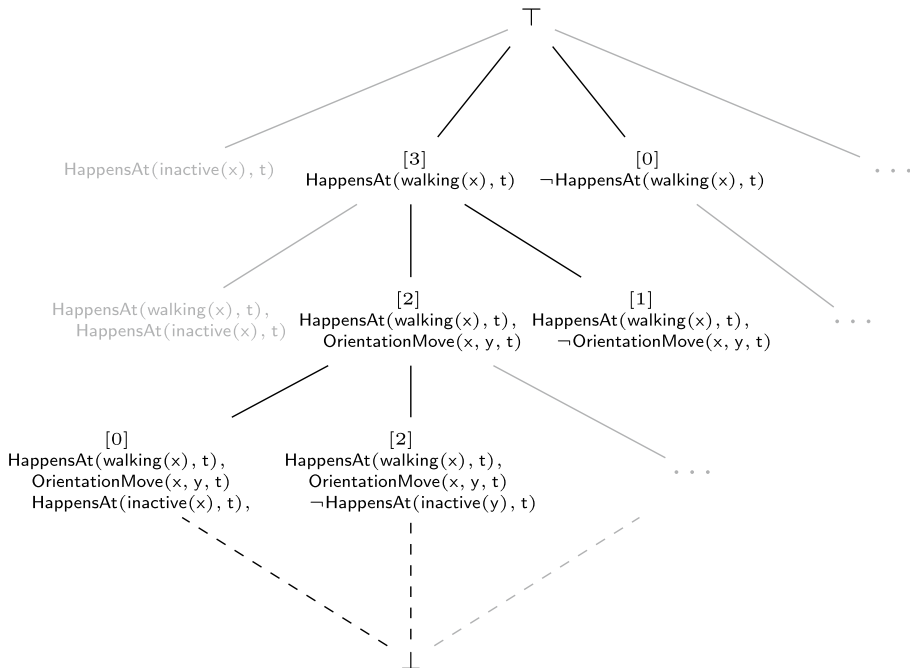


Fig. 4 Path selected by a random tree from the subsumption lattice

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 Sect. 3.2, with the optimised structural distance of Eq. (12) as follows:

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

where α controls the relative importance of each of the two distances (see Table 1 for the notation of the distance measures). Similar to SPLICE, the hybrid distance is turned into a similarity as $1 - d_h^b(v_i, v_j)$.

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 on each neighbourhood. To that end, SPLICE⁺ uses a temporal variant of k NN that connects each unlabelled vertex to its k -nearest (most similar) labelled neighbours, as well as to its temporally adjacent ones. For instance, the bottom example in Fig. 1 would connect to the examples at time-points 49 and 51, since they are temporally adjacent to time-point 50, and would also connect to its k closest labelled examples. If $k = 1$ then it would connect only to the top example since they are almost identical. The intuition behind this extension of k NN is that temporally adjacent vertices should affect the labelling of each other. In terms of label propagation, temporally adjacent neighbours should exchange information about their 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 Sect. 2.4. Given a memory size parameter τ , the synopsis removes

Input: \mathcal{F} : a set of half-space trees, \mathcal{V} : a set of examples, τ : synopsis memory

- 1: Partition examples into labelled and unlabelled $\mathcal{V} = (\mathcal{V}_L, \mathcal{V}_U)$
- 2: $\text{UPDATEFOREST}(\mathcal{F}, \mathcal{V}_L^t \cup \mathcal{V}_U^t) \triangleright$ Update tree sizes using the examples of micro-batch t .
- 3: **if** $\mathcal{V}_L^t = \emptyset$ and \mathcal{V}_L has new labels since the last optimisation **then**
- 4: $\mathbf{b} = \text{LMFS}(\mathcal{V}_L, \mathcal{M})$
- 5: **end if**
- 6: $\mathcal{V}_\tau \leftarrow \mathcal{V}_L \cup \mathcal{V}_U \setminus \mathcal{V}_U^t$
- 7: **for** $v_i \in \mathcal{V}_\tau$ **do**
- 8: **for** $v_j \in \mathcal{V}_U^t$ **do**
- 9: $\mathbf{W}_{v_i, v_j} \leftarrow 1 - d_h^{\mathbf{b}}(v_i, v_j)$
- 10: **end for**
- 11: **end for**
- 12: $V_\tau \leftarrow V_\tau \cup \mathcal{V}_U^t$
- 13: Apply the connection heuristic: $\mathbf{W}' = \text{temporal-}k\text{NN}(\mathbf{W})$
- 14: **while** $|\mathcal{V}_\tau| > (\tau + |\mathcal{V}_L|)$ **do**
- 15: Find oldest unlabelled vertex $v_o \leftarrow \mathcal{V}_\tau \setminus \mathcal{V}_L$
- 16: **for** all vertex pairs $v, v' \in \mathcal{V}_\tau : v \neq v'$ **do**
- 17: $w_{v, v'} \leftarrow w_{v, v'} + \frac{w_{v_o, v} w_{v_o, v'}}{\text{degree}(v_o)}$
- 18: **end for**
- 19: Remove the edges of v_o from \mathbf{W}'
- 20: **end while**
- 21: **return** \mathbf{W}'

Algorithm 4 $\text{GRAPHCONSTRUCTION } \mathcal{F}, \mathcal{V}, \tau$

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 (Wagner et al., 2018). Therefore, the synopsis renders the labelling invariant to different batch sizes. Algorithm 4 presents the graph construction pseudo-code.

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 \mathcal{V}_L^t and unlabelled \mathcal{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 \mathcal{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 \mathcal{V}_τ of stored examples is composed of all the labelled examples \mathcal{V}_L and the τ stored unlabelled ones, $\mathcal{V}_U \setminus \mathcal{V}_U^t$, where τ is the synopsis size. Then, the temporal- $k\text{NN}$ 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).

4 Empirical evaluation

Our experimental hypothesis is that SPlice^+ should outperform SPlice . To that end, we compare SPlice^+ to its predecessor (SPlice) on the task of composite event recognition (CER), using OLED , an open-source software² for online structure learning. We also perform experiments using an Iterative Cross-Training (ICT) technique, proposed by Soonthornphisaj and Kijisirikul (2004). In order to be able to learn EC theories, we replace the PROGOL system (Muggleton, 1995) with ILASP (Law et al., 2016, 2018), a state-of-the-art structure learner. Since ICT combines ILASP with a Naïve Bayes classifier, we refer to the combined system as ILASP-NB . For the experiments, we use the publicly available benchmark activity recognition dataset of the CAVIAR project,³ the publicly available maritime monitoring dataset, concerning the activity of vessels in the Atlantic Ocean, near the port of Brest, France,⁴ and a fleet management dataset, recording the activity of vehicles around Greece and some neighbouring countries.⁵ The experiments were performed on a computer with 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.⁶

4.1 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 inactive. In addition, the coordinates of tracked persons are used to capture spatial relations, e.g. two persons being relatively close to each other. The second level contains CEs, describing activities between multiple persons and/or objects, i.e., people meeting or moving together.

Similar to our earlier work (Michelioudakis et al., 2019), 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. Thus, for both CEs the number of negatives is significantly larger than the number of positives.

The maritime dataset consists of vessel position signals (AIS messages) 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

² <https://github.com/nkatzz/OLED>.

³ <http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1>.

⁴ <https://zenodo.org/record/1167595>.

⁵ <https://www.vodafoneinnovus.com>.

⁶ Instructions for reproducing all experiments may be found in: https://users.iit.demokritos.gr/~vagmcs/pub/splice_plus.

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 by performing CER 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.⁷ 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 vehicles moving around Greece and neighbouring countries. The SDEs include positional information, such as vehicle speed changes, proximity to points of interest and operational status, such as fuel level. We focus on `nonEconomicDriving` and `dangerousDriving` CEs, since they are the more complex ones. The former expresses a driving activity where the driver is over-speeding despite having limited fuel, while the latter describes various dangerous driving behaviours, including over-speeding on ice, abrupt accelerating or braking and cornering other vehicles. Since the dataset is unlabelled, we produced synthetic annotation similar to the maritime dataset, using the RTEC engine and hand-crafted rules. We have 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 example instances that `nonEconomicDriving` occurs and 639 for `dangerousDriving`.

4.2 Experimental setup

Two learning scenarios have been evaluated, with corresponding experiments. 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 on each supervision percentage. We repeated the random selection 20 times, leading to 20 runs per supervision level, in order to obtain a good estimate of the performance.

This random selection scenario was the one used in SPLICE, making the results directly comparable to our earlier work (Michelioudakis et al., 2019). However, in a typical stream learning situation, usually, the assumption of labels arriving randomly on-stream is unrealistic. A more appropriate assumption is that a 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, 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.

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

⁷ <https://zenodo.org/record/2557290>.

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 in 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 contained different proportions of positive and negative examples, the runs were not stratified.

We present accuracy results for both supervision completion and structure learning using the F_1 -score. All reported statistics are micro-averaged over the recognised instances of CEs. The F_1 -score of supervision completion is measured over both train and test sets. For structure learning, the reported statistics on the CAVIAR dataset, 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, leaving again complete sequences out for testing.

We follow Ting et al. (2019) and use $T = 100$ random trees for computing the mass-based dissimilarity, and $\alpha = 0.5$ in order to equally weight the two distance measures. We only present results for the best performing k values for both SPLICE and SPLICE⁺ in order to avoid clutter. We have experimented using $k \in [1, 5]$, but for larger k values, performance was notably decreasing for both algorithms. It is worth noting that for any combination of k , SPLICE⁺ always outperforms SPLICE. However, the connection strategy used by SPLICE⁺ is not directly comparable to that of SPLICE. A more detailed hyper-parameter selection for SPLICE can be found in Michelioudakis et al. (2019). ILASP was trained using the default parameters.

4.3 Experimental results

4.3.1 Activity recognition

First, we compare the performance of SPLICE⁺ against SPLICE on the activity recognition dataset for both meet and move CEs. Figure 5 shows 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 both SPLICE and ILASP-NB in most cases even at 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 iterations to converge. As a result, it was 12 times slower than SPLICE⁺ in the CAVIAR dataset.

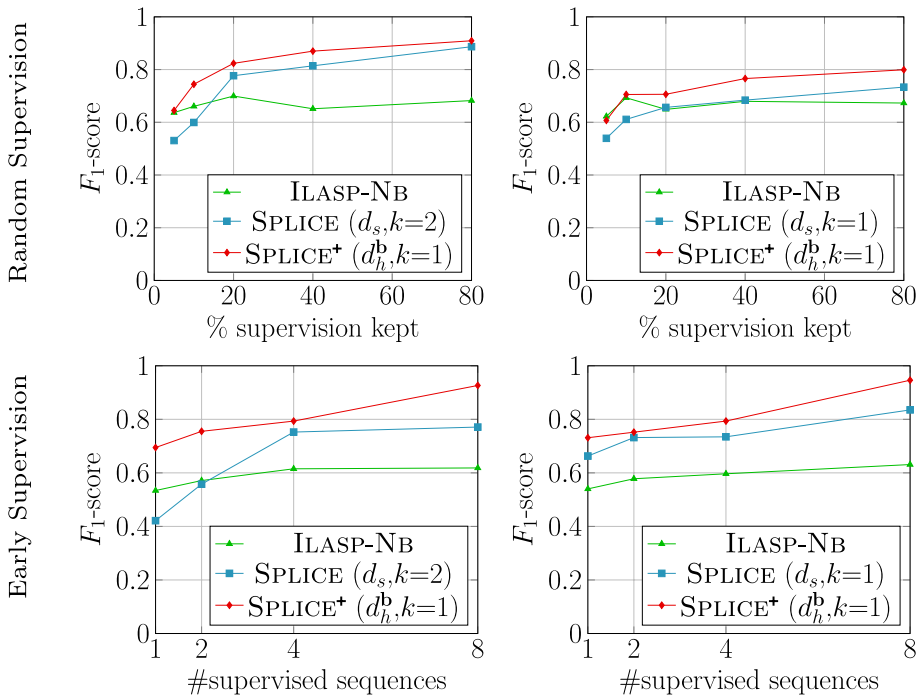


Fig. 5 F_1 -score of supervision completion on meet (left) and move (right) as supervision increases. In the first scenario, supervision arrives uniformly at random (top), while in the second one is provided only at the beginning of the sequence (bottom). The notation d_s and d_h^b refer to the structural and hybrid distances respectively

Improved performance in SPLICE^+ comes at the cost of a decrease in runtime, i.e., the time to process both the train and test set, as shown in Fig. 6. Note that SPLICE^+ is always slower than SPLICE , since it has to update the trees and select appropriate features for every micro-batch. 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 4). 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 labels arrive in the beginning of training.

Table 2 compares the F_1 -score of SPLICE^+ using the default, optimized hybrid distance (d_h^b), as presented in Fig. 5, against the simpler structural distance (d_s). We only present the F_1 -score for low supervision levels because they are the more interesting in a semi-supervised setting. The results suggest that the informed hybrid distance always performs better than the simpler structural distance. In particular, the proposed distance led to 5% and 8% improvement in F_1 -score on average for the random and early supervision scenarios respectively.

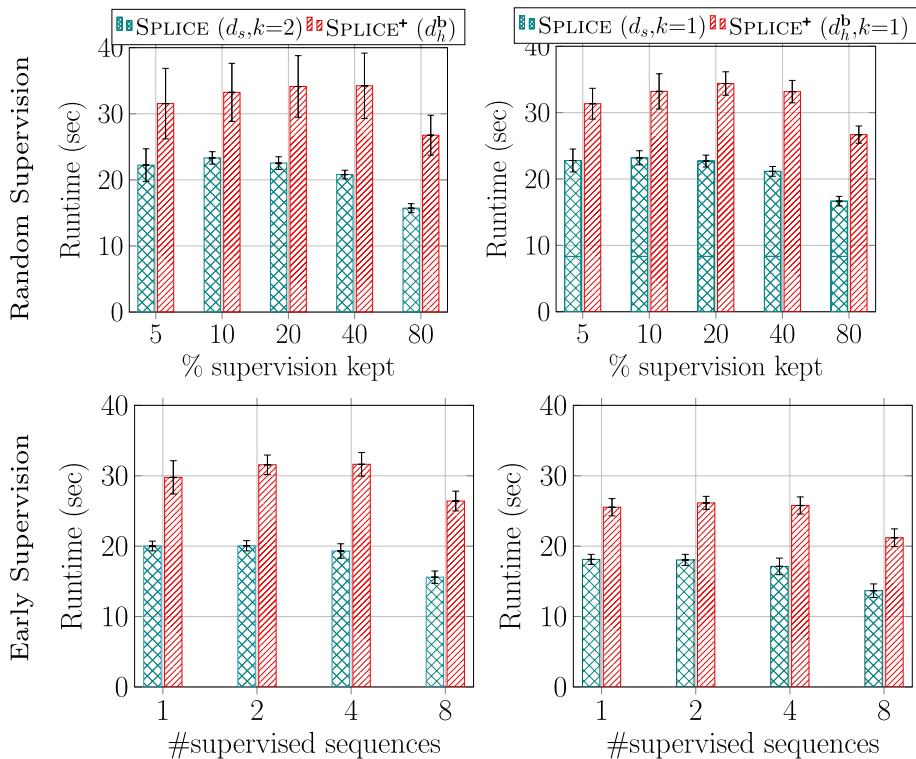


Fig. 6 Runtime performance of supervision completion on meet (left) and move (right) as supervision increases. The runtime is macro-averaged over all samples. In the first scenario, supervision arrives uniformly at random (top), while in the second one is provided only at the beginning of training (bottom). 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

Figure 7 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. 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 its predecessor SPLICE -OLED the only noticeable difference is in the meet CE, when little supervision is available. 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.

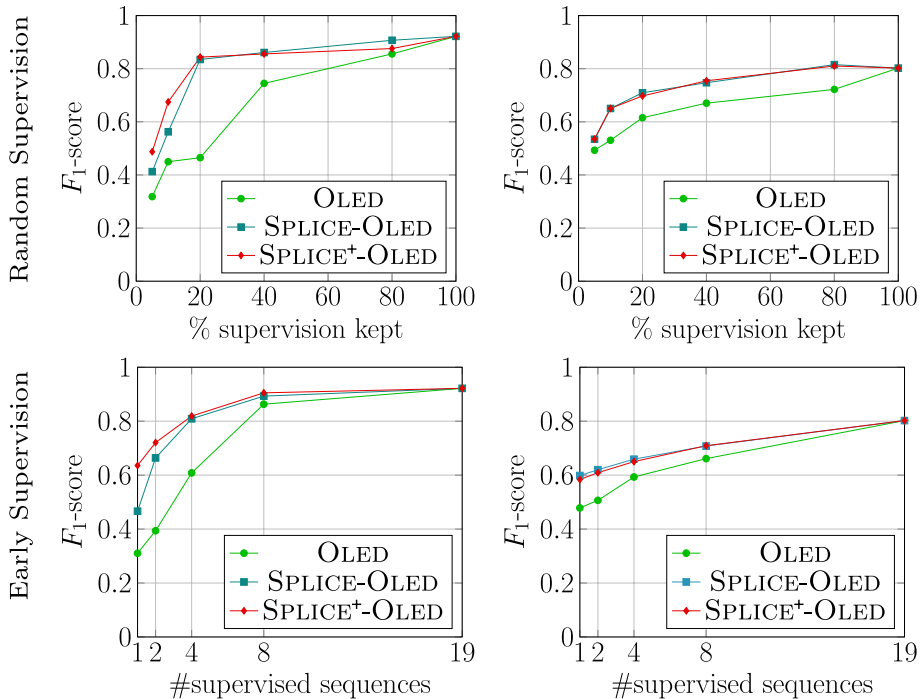
4.3.2 Maritime monitoring

For the maritime monitoring dataset, we performed the same evaluation, as for human activity recognition, for both *pilotOps* and *rendezvous* CEs. The F_1 -score of supervision

Table 2 Comparison of SPlice^+ using the structural distance (d_s) against the optimized hybrid distance (d_h^b) on meet and move

CE	Distance	Random supervision		Early supervision	
		5%	10%	1	2
meet	d_s	0.62	0.70	0.56	0.69
	d_h^b	0.67	0.77	0.70	0.76
move	d_s	0.57	0.64	0.67	0.71
	d_h^b	0.58	0.69	0.73	0.75

Bold font highlights the best performing distance measure

**Fig. 7** 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 is provided at the beginning of the training sequence (bottom)

completion on both scenarios, using the same notation, is presented in Fig. 8. The results suggest that SPlice^+ effectively infers the missing labels, significantly outperforms SPlice in all cases, even for high supervision levels (80% uniform supervision or 8 sequences). The difference is again larger in the early supervision scenario, where supervision appears only in the beginning of training. The results of ILASP-NB are not shown here, due to very high training times of the algorithm. In particular, a single run of ILASP-NB requires more than 3 hours to complete, while SPlice^+ achieves better results in 30 seconds.

An interesting observation is that in the early supervision scenario the F_1 -score of 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

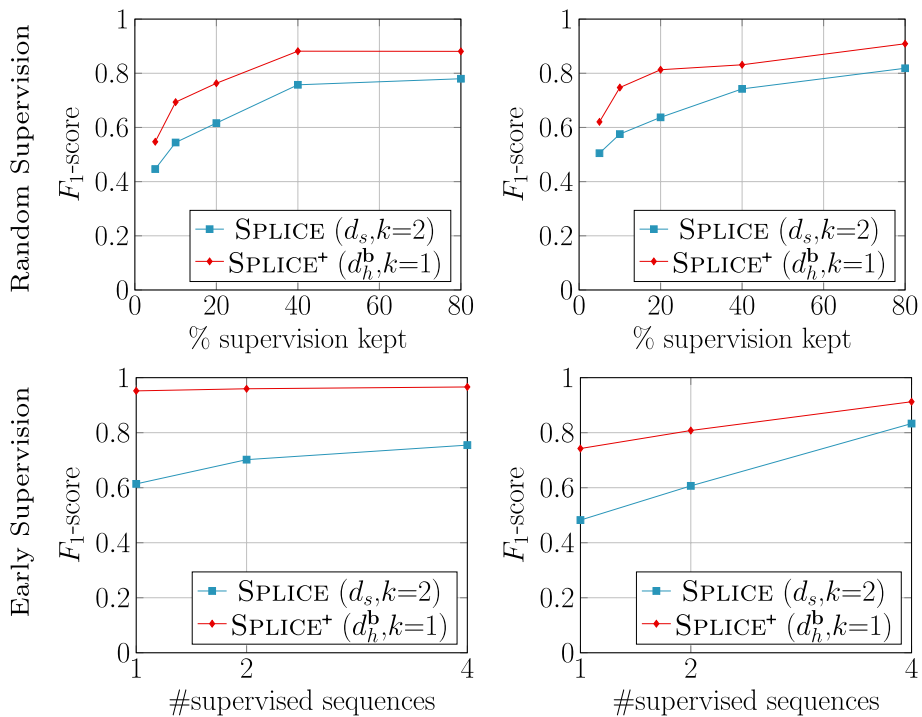


Fig. 8 F_1 -score of supervision completion on pilotOps (left) and rendezvous (right) as supervision increases. In the first scenario, supervision arrives uniformly at random (top), while in the second one, supervision is provided only at the beginning of the training sequence (bottom)

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 and select features for each micro-batch. However, the computational penalty is still tolerable since it is typically below 25 seconds. Full runtime results are included in the online Appendix.⁸

Similar to the task of human activity recognition, in Table 3, we present the F_1 -score of SPLICE + using the default, optimized hybrid distance (d_h^b), as presented in Fig. 8, against the simpler structural distance (d_s). The results confirm the improved performance of the informed hybrid distance. In particular, the proposed distance led to 6% and 5% improvement in F_1 -score on average for the random and early supervision scenarios respectively.

In Table 4, we present the change in F_1 -score as the batch size increases. SPLICE + is more robust than SPLICE, and this is more apparent in the maritime dataset, as opposed to the activity recognition one, the results for which are presented in the online Appendix. 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 0.01. For 2 supervised sequences the variation is even greater, since the F_1 -score of SPLICE varies from 0.03–0.18, in contrast to that

⁸ https://users.iit.demokritos.gr/~vagmcs/pub/splice_plus/appendix.pdf.

Table 3 Comparison of SPICE^+ on `pilotOps` and `rendezvous` using the simple structural distance (d_s) and the hybrid distance (d_h^b)

CE	Distance	Random supervision		Early supervision	
		5%	10%	1	2
rendezvous	d_s	0.59	0.70	0.69	0.79
	d_h^b	0.62	0.75	0.74	0.81
pilotOps	d_s	0.47	0.63	0.78	0.90
	d_h^b	0.56	0.69	0.95	0.96

Bold font highlights the best performing distance measure

Table 4 F_1 -score of `pilotOps`, `rendezvous` for varying batch sizes: $\text{SPICE}/\text{SPICE}^+$

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

of SPICE^+ where the variation remains 0.01. The same holds for the `rendezvous` CE, where for 1 supervised sequence, SPICE varies from 0.08–0.15, while SPICE^+ varies only 0.01.

In Fig. 9, we present the structure learning results of SPICE -OLED against SPICE^+ -OLED, and OLED alone using the incomplete dataset (OLED alone uses only the labelled examples). SPICE^+ -OLED clearly outperforms both SPICE -OLED and OLED alone by a large margin, which indicates the usefulness of the proposed approach.

4.3.3 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 Fig. 10. The results appear to be consistent with the previous tasks, since SPICE^+ yields the best overall performance. However, in this dataset the difference with SPICE is smaller, due to the fact that fleet management does not have 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 SPICE^+ and SPICE is similar to that observed in the activity recognition dataset (see online Appendix for a full comparison). Briefly, the computational penalty is typically below 20 seconds, due to the updates of the

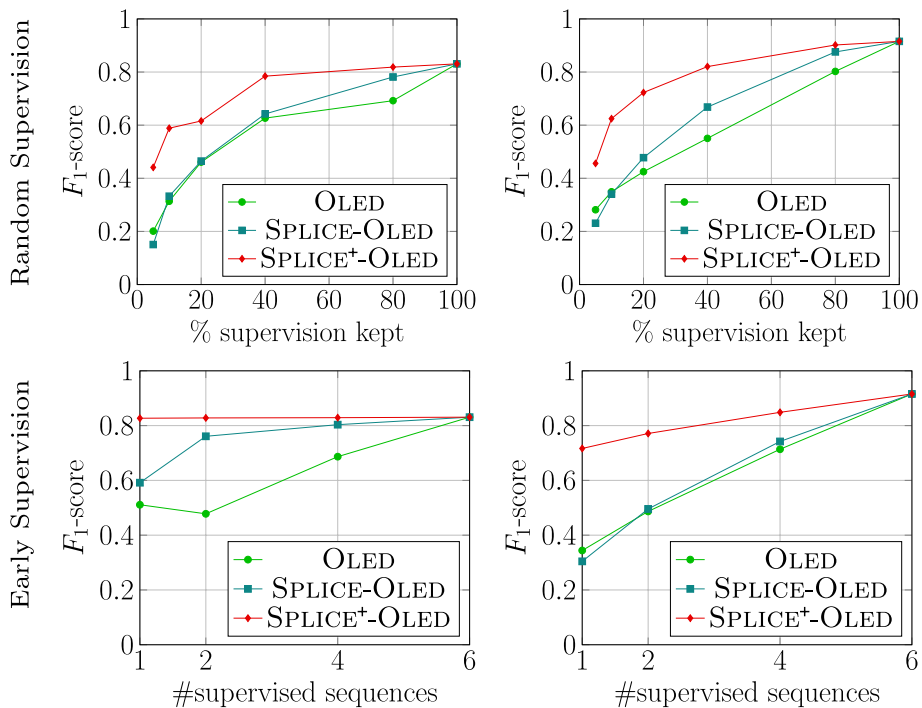


Fig. 9 Structure learning on *pilotOps* (left) and *rendezvous* (right) as supervision increases. In the first scenario, supervision arrives uniformly at random (top), while in the second one is provided at the beginning of the training sequence (bottom)

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.

Finally, in Fig. 11, 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 sub-figures, namely *nonEconomicDriving* under random supervision. This is in agreement with the supervision completion results, shown in Fig. 10.

4.4 Discussion

The experimental results on the three real-life datasets demonstrated 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. Moreover, the findings suggest that the graph synopsis and the temporal connection of the unlabelled examples enable SPLICE⁺ to achieve robust labelling regardless of the batch size.

In the presence of noisy or irrelevant features, each component of the proposed hybrid distance function contributes to an improved overall performance. The feature selection component works well when the labels are sufficient to select a good subset of logical atoms. However, in a semi-supervised learning task, this is not always the case, as in

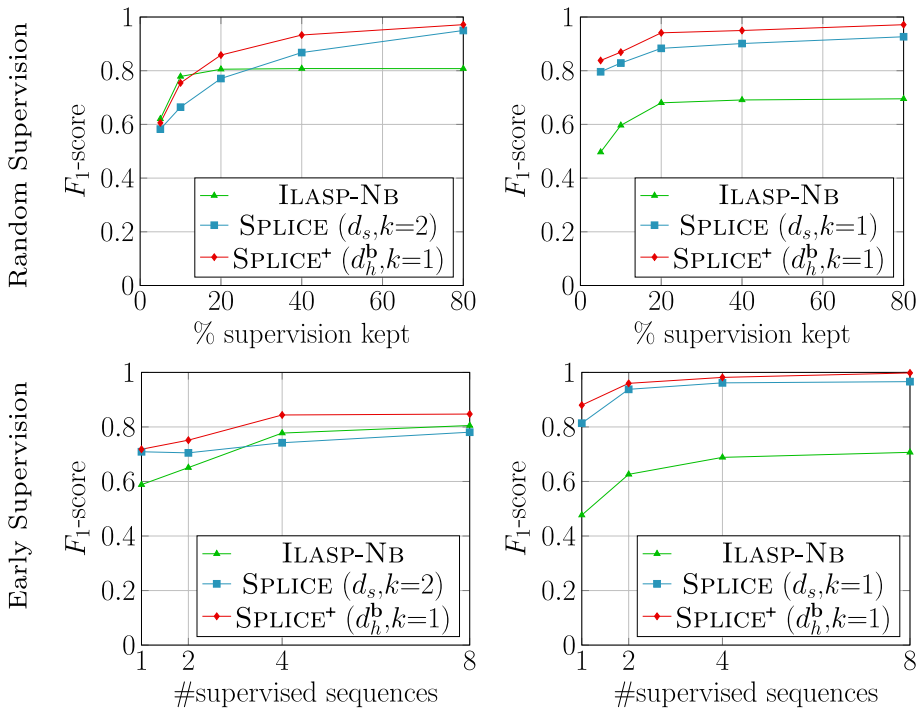


Fig. 10 F_1 -score of supervision completion on nonEconomicDriving (left) and dangerousDriving (right) as supervision increases. In the first scenario, supervision arrives uniformly at random (top), while in the second one is provided only at the beginning of the training sequence (bottom)

the case of the maritime monitoring dataset, where no gain is observed. In contrast, we observe on average 2% and 4% improvement for random and early supervision respectively in the activity recognition dataset. On the other hand, the mass-based dissimilarity seems to improve F_1 -score by 4% and 5% on average in activity recognition and by 5% and 8% on maritime monitoring (see online Appendix). By combining both supervised and unsupervised metric learning, SPLICE + manages to refine the distance measurements when the provided CE examples are noisy, similar to negative examples, or have multiple, alternative definitions (Tables 2 and 3). SPLICE fails to infer the labels correctly in these cases, because it relies solely on the structural distance of Eq. (1) without considering the relative importance of different features (logical atoms). Hence, SPLICE performs well only on fleet management and move CE (see Fig. 5 right and 10), since the examples are more distinct compared to their negative counterparts and have a single definition. It is worth noting that ILASP-NB also performs better on move (random supervision) and nonEconomicDriving CEs because they can be learned using fewer examples.

In early supervision scenarios, where labels are provided in a contiguous sequence of events, the disparity in performance between SPLICE and SPLICE + becomes more noticeable (Fig. 8 bottom). This observation reinforces the belief that SPLICE underperform when labeled examples differ significantly from their unlabelled counterparts due to the presence of irrelevant features in other parts of the sequence. As expected, this issue is less

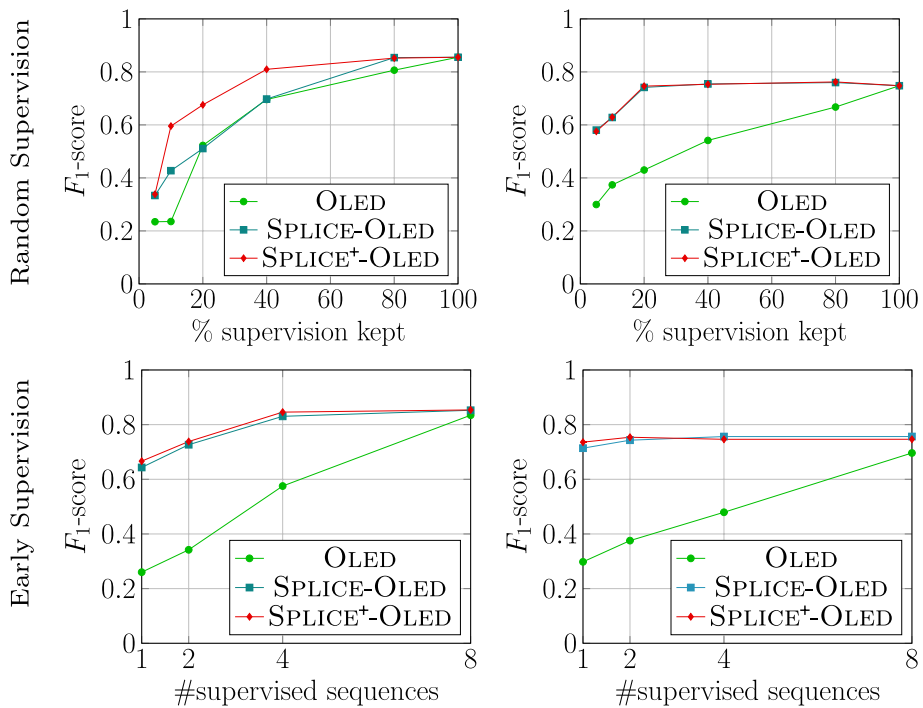


Fig. 11 Structure learning on nonEconomicDriving (left) and dangerousDriving (right) as supervision increases. In the first scenario, supervision arrives uniformly at random (top), while in the second one, supervision is provided only at the beginning of the training sequence (bottom)

prominent when labels are uniformly sampled from the entire training sequence (Fig. 8 top).

The experimental results over different batch sizes confirm that temporal label propagation (see Sect. 3.3) render the predictive accuracy of SPLICE⁺ invariant to batch size (see Table 4). The effect is stronger when the labelled examples are fewer (1 supervised sequence over 4 supervised sequences), which indicates the importance of connecting the unlabelled examples between micro-batches. Therefore, the accuracy of SPLICE⁺ does not fall as the dataset size increases. However, larger micro-batch size hurts runtime performance, since the number of unlabelled examples per micro-batch is higher, increasing the runtime complexity of label propagation (Michelioudakis et al., 2019). Note that size does not affect the time for updating the Half-Space trees, since the update algorithm is linear to the number of examples (see Algorithm 3). The LMFS optimization also remains unaffected (assuming labelled examples are sparse), since it only considers labelled examples.

In contrast to typical supervised learning, semi-supervised performance does not always improve when the dataset size increases, since the provided labelled examples may not be representative of the entire data distribution. In fact, it has been shown that under certain conditions, increasing the number of unlabelled examples does increase predictive accuracy (Singh et al., 2008). When it comes to runtime performance, simply increasing the dataset size (assuming a small batch size) should not negatively impact performance,

unless there is a significant increase in the number of labeled examples, which has an exponential effect on the runtime of Algorithm 1.

Beyond accuracy, the rules discovered using SPlice^+ are interesting and intuitive. In the activity recognition dataset, using random supervision (5%), SPlice^+ -OLED often manages to learn rules such as the following:

$$\begin{aligned} \text{InitiatedAt}(\text{meet}(x, y), t) &\Leftarrow \\ &\quad \text{HappensAt}(\text{inactive}(x), t) \wedge \text{HappensAt}(\text{inactive}(y), t) \wedge \\ &\quad \text{OrientationMove}(x, y, t) \wedge \text{Close}(x, y, 24, t) \\ \text{TerminatedAt}(\text{meet}(x, y), t) &\Leftarrow \text{HappensAt}(\text{walking}(x), t) \end{aligned}$$

The first rule requires both persons to be inactive and close enough (less than 24 pixels), while having similar orientation. The second, more general, rule states that if some person starts walking, then the meeting is over. In the early supervision scenario (1 supervised sequence), the rules may be more error prone, but still very intuitive, such as the following:

$$\begin{aligned} \text{InitiatedAt}(\text{meet}(x, y), t) &\Leftarrow \\ &\quad \text{HappensAt}(\text{active}(x), t) \wedge \text{HappensAt}(\text{active}(y), t) \wedge \\ &\quad \text{OrientationMove}(x, y, t) \\ \text{TerminatedAt}(\text{meet}(x, y), t) &\Leftarrow \\ &\quad \text{HappensAt}(\text{walking}(x), t) \wedge \text{HappensAt}(\text{walking}(y), t) \wedge \\ &\quad \text{Close}(x, y, 24, t) \end{aligned}$$

In the first rule, SPlice^+ -OLED did not manage to learn a spatial constraint, thus it may assume a meeting is initiated regardless of the distance between the two persons. On the other hand, SPlice -OLED learns similar rules in some of the runs (about 30% of them), but in most cases it either learns more general initiation rules leading to many false positives or very specific termination rules yielding false negatives. There are also runs where SPlice -OLED either only learns initiation or termination rules, which indicates that the inferred labels are contradicting. Such cases occur also in SPlice^+ , but they are fewer and usually suggest that the labelled examples in the underlying training sample are outliers.

In the maritime dataset, using either random or early supervision, SPlice^+ -OLED usually manages to learn proper initiation rules for the rendezvous CE, such as the following:

$$\begin{aligned} \text{InitiatedAt}(\text{rendezVous}(x, y), t) &\Leftarrow \\ &\quad \text{HappensAt}(\text{lowSpeed}(x), t) \wedge \text{HappensAt}(\text{lowSpeed}(y), t) \wedge \\ &\quad \text{Proximity}(x, y, t) \\ \text{InitiatedAt}(\text{rendezVous}(x, y), t) &\Leftarrow \\ &\quad \text{HappensAt}(\text{stopped}(x, \text{FarFromPorts}), t) \wedge \\ &\quad \text{HappensAt}(\text{stopped}(y, \text{FarFromPorts}), t) \wedge \\ &\quad \text{Proximity}(x, y, t) \end{aligned}$$

Similarly good rules are discovered in the fleet management dataset. Interestingly, SPlice -OLED usually fails to learn the second initiation rule, which is explained by the fact that there are more examples in the dataset in which vessels are moving at low speed relatively close to each other, than examples where vessels stop moving. Therefore, given very few labelled examples, SPlice fails to capture the important features that highlight

the rendezvous behaviour, leading to contradicting labelling and noisy rules that OLED rejects during structure learning. Note that the interpretability of these rules makes it easy for the end-users to modify and adapt to their needs, e.g., add a missing spatial constraint, in contrast to more complex models that may require expert knowledge in order to tune and reuse the learned model.

In summary, SPlice^+ facilitates learning better rules due to its improved distance measures, that identify useful features among noisy examples, yielding better labelling. Moreover, its labelling is not affected by the batch size, in contrast to SPlice which offers no guarantees. SPlice^+ can be particularly efficient in the more realistic and challenging early supervision setting, in the presence of noisy labels and irrelevant features. Although SPlice^+ is used to learn CE definitions in the Event Calculus, it can be applied to learn any temporal domain formalised in first-order logic, as long as, the examples do not include many long range dependencies (densely connected group of facts) or temporal intervals (CEs defined over interval relationships). Long range dependencies would significantly increase the complexity both of the structural distance and the depth of Half-Space trees. Additionally, temporal intervals necessitate numerical handling to effectively incorporate them.

Additionally, although SPlice^+ is efficient in general, its computational performance can be compromised by a large increase in the number of labelled examples, since the optimization presented in Algorithm 1 is NP-complete. The runtime complexity of the mass-based dissimilarity also increases with the number of atoms in the Hebrand base, since trees have higher depth. Hence, in scenarios where there are limited positive examples or noisy and irrelevant features (logical atoms), SPlice^+ proves to performs best. Conversely, when reliable labelled examples are provided and irrelevant features are minimal, SPlice may be preferred due to its runtime efficiency. However, it is important to carefully select the batch size as it can impact the predictive performance of SPlice .

5 Related work

Learning composite event rules from sensory input is a challenging task that is receiving increasing attention in the literature, since it constitutes a limiting factor to most CER applications. Recent approaches attempt to learn propositional CE rules in the form of a domain-specific language from historical data (Margara et al., 2014; George et al., 2016; Mousheimish et al., 2017; Bruns et al., 2019). On the other hand, online structure learners, employ logic formalisms, such as the Event Calculus, to capture the relational dependencies of complex events (Michelioudakis et al., 2016; Katzouris et al., 2016, 2018). However, these methods, either propositional or relational, assume that a fully-labelled training dataset is available.

Despite the plethora of semi-supervised learning (SSL) methods that have been developed for tackling the problem of missing supervision (Chapelle et al., 2006; Zhu et al., 2009), to our knowledge, only variations of co-training have been adapted to Inductive Logic Programming (Li & Guo, 2012; Soonthornphisaj & Kijirikul, 2004). However, co-training is not suitable for online learning since it assumes that the training data can be separated into distinct views (Nigam & Ghani, 2000), namely disjoint feature sets that provide complementary information about each instance, while each view is sufficient to accurately predict each class. Another approach to rule learning is to learn fuzzy if-then classification rules from partially labelled data (Klose & Kruse, 2005). Nevertheless, these methods cannot be directly used to learn first-order Event Calculus rules. Even though SSL has been extensively studied in static environments (Dyer & Polikar, 2012), online SSL that

operates on data streams remains an open challenge. SPLICE (Michelioudakis et al., 2019), as presented in Sect. 2.1, is a recent online approach to semi-supervised structure learning for CER applications. SPLICE employs graph-based methods (Blum & Chawla, 2001; Zhu et al., 2003; Zhou et al., 2003; Joachims, 2003; Wang et al., 2008, 2013) to infer the missing labels of the training examples.

Only very few online graph-based SSL methods have been proposed to date. (Delalleau et al., 2005) proposed an inductive algorithm that learns a labelling function using the graph similarity matrix constructed given a training set containing labelled and unlabelled data. Subsequent incoming unlabelled examples are labelled using the previously learned inductive function. However, new examples (labelled and unlabelled) are not incorporated into the learned model, and thus, the model can become outdated. (Huang et al., 2015) also follow an inductive approach to graph-based SSL, by updating incrementally the inverse of the graph Laplacian matrix, required for computing the labelling. The update is done using a technique that transforms the matrix inverse computation to matrix multiplications in order to save time. Nonetheless, this method requires all incoming data to be stored, leading to continuously increasing memory requirements. (Valko et al., 2010) designed a transductive technique that quantises the stream into a small number of clusters using an online k -center algorithm. Then, the harmonic solution is computed on the cluster centers. SPLICE follows a similar approach for caching the labelled examples, but instead of clustering, it performs logical unification to group the incoming examples into concepts (bottom clauses). Finally, the Temporal Label Propagation method proposed by Wagner et al. (2018), as presented in Sect. 2.4, stores a constant size graph synopsis of the stream and performs label propagation on the compressed graph.

Naturally, graph construction is an important issue of graph-based SSL methods, and thus, the labelling solution is sensitive to the distance measure used to interconnect examples. Since we are interesting in learning CE rules from interpretations (Blockeel et al., 1999), a distance measure for comparing first-order atoms is desirable. Although various measures exist for first-order logic, either structural (Bisson, 1992b, a; Emde & Wetteschereck, 1996; Nienhuys-Cheng, 1997; Bohnebeck et al., 1998; Ramon & Bruynooghe, 1998; Mavroeidis & Flach, 2003) or semantic (Sebag & Schoenauer, 1993; Sebag, 1997), none of these handle irrelevant or noisy features and thus their credibility may be compromised.

At the same time, numerous methods have been proposed to cope with such noisy and irrelevant features, stemming from feature selection (Guyon et al., 2006; Chandrashekar & Sahin, 2014) or metric learning (Kulis, 2013; Wang & 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 (Vergara & Estévez, 2014; Brown, 2009), consistency measures (Arauzo-Azofra et al., 2008), constraint scores (Zhang et al., 2008; Benabdeslem & Hindawi, 2014), and rough sets (Pawlak et al., 1995; Modrzejewski, 1993). These methods usually provide a ranking of the features according to a specific criterion. Thus, the user needs to select a subset from the ranked list, e.g. the top k features.

Metric learning, on the other hand, aims to learn a distance measure on the feature space, so that some given pairs of data points are pulled as close as possible, while others are pushed far apart. There are supervised learning approaches, based on Mahalanobis distance learning (Goldberger et al., 2004), and unsupervised ones, depending on linear reconstruction (Roweis & Saul, 2000; Tenenbaum et al., 2000). Mass-based dissimilarity (Aryal et al., 2014) is also a form of unsupervised metric learning. In contrast to supervised methods (Ting et al. (2019), Section 8), the mass-based approach 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 to combine metric learning and graph-based SSL (Wang & Zhang, 2008; Okada & Nishida, 2010; Pourdamghani et al., 2012), and none of these approaches combines supervised metric learning with mass-based dissimilarity, as in SPlice^+ , in order to exploit both labelled and unlabelled data. Moreover, to our knowledge, no such methods have been applied to logical interpretations.

6 Conclusions and future work

We presented SPlice^+ , a novel 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 $k\text{NN}$ 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. Finally, SPlice^+ constructs a temporal graph and maintains a synopsis from the data stream to achieve robust labelling.

Experimental results using benchmark real-life data from human activity recognition, maritime monitoring, and fleet management, showed that 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 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.

Further extensions of the proposed method are being investigated, including an active learning component that will enable SPlice^+ to enhance its predictions in the presence of noisy labels or concept drift. Additionally, we are considering a distributed implementation that will enhance the scalability of the method.

Author Contributions EM conceived of the presented idea, developed the theory, and performed all the computations required for the experimental analysis. AA and GP supervised the overall research direction and planning, verified the presented methods, and contributed to the interpretation of the results. All authors discussed the results and contributed to the final manuscript.

Funding Open access funding provided by HEAL-Link Greece. This work was funded by the EU H2020 programme, under Grant Agreement No. 825070 (Project INFORE).

Data availability Two out of the three datasets underlying this article are available in the article. The third dataset was provided by Vodafone Innovus under license.

Code Availability Code for experimental analysis is provided as part of the replication package. The link to this package is available in the article.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the

material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References


- Alphonse, É., & Matwin, S. (2002). Feature subset selection and inductive logic programming. In *Proceedings of the 9th international conference on machine learning* (pp. 11–18). Morgan Kaufmann.
- Arauzo-Azofra, A., Benítez, J. M., & Castro, J. L. (2008). Consistency measures for feature selection. *Journal of Intelligent Information Systems*, 30(3), 273–292.
- Artikis, A., Sergot, M. J., & 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., et al. (2012). Logic-based event recognition. *Knowledge Engineering Review*, 27(4), 469–506.
- Aryal, S., Ting, K. M., Haffari, G., et al. (2014). Mp-dissimilarity: A data dependent dissimilarity measure. In *Proceedings of the IEEE international conference on data mining (ICDM)* (pp. 707–712).
- Benabdeslem, K., & Hindawi, M. (2014). Efficient semi-supervised feature selection: Constraint, relevance, and redundancy. *IEEE Transactions on Knowledge and Data Engineering*, 26(5), 1131–1143.
- Bisson, G. (1992a). Conceptual clustering in a first order logic representation. In *Proceedings of the 10th European conference on artificial intelligence* (pp. 458–462). Wiley.
- Bisson, G. (1992b). Learning in FOL with a similarity measure. In *Proceedings of the 10th national conference on artificial intelligence* (pp. 82–87). AAAI Press/MIT Press.
- Blockeel, H., Raedt, L. D., Jacobs, N., et al. (1999). Scaling up inductive logic programming by learning from interpretations. *Data Mining and Knowledge Discovery*, 3(1), 59–93.
- Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the eighteenth international conference on machine learning* (pp. 19–26). Morgan Kaufmann.
- Bohnebeck, U., Horváth, T., & Wrobel, S. (1998). Term comparisons in first-order similarity measures. In *Proceedings of the 8th international workshop on inductive logic programming* (pp. 65–79). Springer.
- Brown, G. (2009). A new perspective for information theoretic feature selection. In *Proceedings of the 12th international conference on artificial intelligence and statistics (AISTATS)* (pp. 49–56).
- Bruns, R., Dunkel, J., & Offel, N. (2019). Learning of complex event processing rules with genetic programming. *Expert Systems with Applications*, 129, 186–199.
- Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1), 16–28.
- Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. MIT Press.
- Chen, B., Liu, H., Chai, J., et al. (2009). Large margin feature weighting method via linear programming. *IEEE Transactions on Knowledge and Data Engineering*, 21(10), 1475–1488.
- Cugola, G., & Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Computing Survey*, 44(3), 1–62.
- Delalleau, O., Bengio, Y., & 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)* (pp. 96–103).
- Deng, Z., & Luo, K. (2015). Cle_Imnn: A novel framework of LMNN based on clustering labeled examples. *Expert Systems with Applications*, 42(14), 5988–5993.
- Dörfler, F., & Bullo, F. (2013). Kron reduction of graphs with applications to electrical networks. *IEEE Transactions on Circuits and Systems*, 60–I(1), 150–163.
- Dries, A., & Raedt, L. D. (2009). Towards clausal discovery for stream mining. In *Proceedings of the 19th international conference on inductive logic programming (ILP)* (pp. 9–16).
- Dyer, K. B., & 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)* (pp. 1–9).
- Emde, W., & Wettschereck, D. (1996). Relational instance-based learning. In *Proceedings of the 13th international conference on machine learning* (pp. 122–130). Morgan Kaufmann.
- Gama, J. (2010). *Knowledge discovery from data streams*. Chapman and Hall/CRC Data Mining and Knowledge Discovery Series, CRC Press.
- George, L., Cadonna, B., & Weidlich, M. (2016). Il-miner: Instance-level discovery of complex event patterns. *Proceedings of the VLDB Endowment*, 10(1), 25–36.
- Gitrakos, N., Alevizos, E., Artikis, A., et al. (2020). Complex event recognition in the big data era: A survey. *VLDB Journal*, 29(1), 313–352.

- Goldberger, J., Roweis, S. T., Hinton, G. E., et al. (2004). Neighbourhood components analysis. *Advances in Neural Information Processing Systems*, 17, 513–520.
- Guyon, I., Nikravesh, M., Gunn, S. R., et al. (eds). (2006). *Feature extraction—Foundations and applications, studies in fuzziness and soft computing* (vol. 207). Springer.
- Huang, L., Liu, X., Ma, B., et al. (2015). Online semi-supervised annotation via proxy-based local consistency propagation. *Neurocomputing*, 149, 1573–1586.
- Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceedings of the 20th international conference on machine learning (ICML)* (pp. 290–297).
- Katzouris, N., Artikis, A., & Paliouras, G. (2016). Online learning of event definitions. *Theory and Practice of Logic Programming*, 16(5–6), 817–833.
- Katzouris, N., Michelioudakis, E., Artikis, A., et al. (2018). Online learning of weighted relational rules for complex event recognition. In *Proceedings of European conference on machine learning and knowledge discovery in databases* (pp. 396–413).
- Klose, A., & Kruse, R. (2005). Semi-supervised learning in knowledge discovery. *Fuzzy Sets and Systems*, 149(1), 209–233.
- Kowalski, R. A., & Sergot, M. J. (1986). A logic-based calculus of events. *New Generation Computing*, 4(1), 67–95.
- 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.
- Law, M., Russo, A., & Broda, K. (2016). Iterative learning of answer set programs from context dependent examples. *Theory and Practice of Logic Programming*, 16(5–6), 834–848.
- Law, M., Russo, A., & Broda, K. (2018). Inductive learning of answer set programs from noisy examples. *Advances in Cognitive Systems*, 7, 57–76.
- Li, Y., & Guo, M. (2012). A new relational tri-training system with adaptive data editing for inductive logic programming. *Knowledge-Based Systems*, 35, 173–185.
- Liu, F. T., Ting, K. M., & Zhou, Z. (2008). Isolation forest. In *Proceedings of the 8th IEEE international conference on data mining (ICDM)* (pp. 413–422).
- Margara, A., Cugola, G., & 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)* (pp. 47–58).
- Mavroeidis, D., & Flach, P. A. (2003). Improved distances for structured data. In *Proceedings of the 13th international workshop on inductive logic programming (ILP)* (pp. 251–268).
- Michelioudakis, E., Artikis, A., & Paliouras, G. (2019). Semi-supervised online structure learning for composite event recognition. *Machine Learning*, 108(7), 1085–1110.
- Michelioudakis, E., Skarlatidis, A., Paliouras, G., et al. (2016). Online structure learning using background knowledge axiomatization. In *Proceedings of European conference on machine learning and knowledge discovery in databases* (pp. 242–237).
- Modrzejewski, M. (1993). Feature selection using rough sets theory. In *Proceedings of the European conference on machine learning* (pp. 213–226).
- Mousheimish, R., Taher, Y., & 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)* (pp. 158–169).
- Mueller, E. T. (2008). Event calculus. In *Handbook of knowledge representation, foundations of artificial intelligence* (Vol. 3, pp. 671–708). Elsevier.
- Muggleton, S. (1995). Inverse entailment and Prolog. *New Generation Computing*, 13, 245–286.
- 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* (pp. 213–226). Springer.
- Nigam, K., Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 2000 ACM CIKM international conference on information and knowledge management* (pp. 86–93). ACM.
- Okada, S., 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)* (pp. 468–478).
- Patroumpas, K., Alevizos, E., Artikis, A., et al. (2017). Online event recognition from moving vessel trajectories. *GeoInformatica*, 21(2), 389–427.
- Pawlak, Z., Grzymala-Busse, J. W., Slowinski, R., et al. (1995). Rough sets. *Communications of the ACM*, 38(11), 88–95.

- Pitsikalis, M., Artikis, A., Dreo, R., et al. (2019). Composite event recognition for maritime monitoring. In *Proceedings of the 13th ACM international conference on distributed and event-based systems* (pp. 163–174). ACM.
- Plotkin, G. D. (1971). Automatic methods of inductive inference. PhD thesis, Edinburgh University.
- Pourdamghani, N., Rabiee, H. R., & Zolfaghari, M. (2012). Metric learning for graph based semi-supervised human pose estimation. In *Proceedings of the 21st international conference on pattern recognition (ICPR)* (pp. 3386–3389).
- Ramon, J., & Bruynooghe, M. (1998). A framework for defining distances between first-order logic objects. In: *Proceedings of the 8th international workshop on inductive logic programming* (pp. 271–280). Springer.
- Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500), 2323–2326.
- Sebag, M. (1997). Distance induction in first order logic. In *Proceedings of the 7th international workshop on inductive logic programming* (pp. 264–272).
- Sebag, M., & Schoenauer, M. (1993). A rule-based similarity measure. In *1st European workshop on topics in case-based reasoning* (pp. 119–131).
- Singh, A., Nowak, R. D., & Zhu, X. (2008). Unlabeled data: Now it helps, now it doesn't. In *Advances in neural information processing systems 21, proceedings of the twenty-second annual conference on neural information processing systems, Vancouver, British Columbia, Canada, December 8–11, 2008* (pp. 1513–1520). Curran Associates, Inc.
- Song, K., Nie, F., Han, J., et al. (2017). Parameter free large margin nearest neighbor for distance metric learning. In *Proceedings of the 31st AAAI conference on artificial intelligence* (pp. 2555–2561). AAAI Press.
- Soonthornphisaj, N., & Kijisirikul, B. (2004). Combining ILP with semi-supervised learning for web page categorization. In *Proceedings of the international conference on computational intelligence* (pp. 322–325).
- Srinivasan, A., & Bain, M. (2017). An empirical study of on-line models for relational data streams. *Machine Learning*, 106(2), 243–276.
- Tenenbaum, J. B., Silva, Vd., & 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., et al. (2013). Mass estimation. *Machine Learning*, 90(1), 127–160.
- Ting, K. M., Zhu, Y., Carman, M. J., et al. (2019). Lowest probability mass neighbour algorithms: relaxing the metric constraint in distance-based neighbourhood algorithms. *Machine Learning*, 108(2), 331–376.
- Valko, M., Kveton, B., Huang, L., et al. (2010). Online semi-supervised learning on quantized graphs. In *Proceedings of the 26th conference on uncertainty in artificial intelligence* (pp. 606–614).
- Vergara, J. R., & 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., et al. (2018). Semi-supervised learning on data streams via temporal label propagation. In *Proceedings of the 35th international conference on machine learning (ICML)* (pp. 5082–5091).
- Wang, F., & Sun, J. (2015). Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery*, 29(2), 534–564.
- Wang, F., & Zhang, C. (2008). Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1), 55–67.
- Wang, J., Jebara, T., Chang, S. (2008). Graph transduction via alternating minimization. In *Proceedings of the 25th international conference on machine learning (ICML)* (pp. 1144–1151).
- Wang, J., Jebara, T., & Chang, S. (2013). Semi-supervised learning using greedy max-cut. *Journal of Machine Learning Research*, 14(1), 771–800.
- Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10, 207–244.
- Williams, H. P. (2009). *Logic and integer programming, logic and integer programming* (Vol. 130). Springer.
- Zhang, D., Chen, S., & 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., et al. (2003). Learning with local and global consistency. In *Proceedings of advances in neural information processing systems (NIPS)* (Vol. 16, pp. 321–328).
- Zhu, X., Ghahramani, Z., Lafferty, J. D. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th international conference on machine learning* (pp. 912–919). AAAI Press.
- Zhu, X., Goldberg, A. B., Brachman, R., et al. (2009). *Introduction to semi-supervised learning*. Morgan and Claypool Publishers.
- Zucker, J., & Ganasia, J. (1996). Representation changes for efficient learning in structural domains. In *Proceedings of the 13th international conference on machine learning* (pp. 543–551). Morgan Kaufmann.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Evangelos Michelioudakis^{1,2}  · Alexander Artikis^{1,3} · Georgios Paliouras¹

✉ Georgios Paliouras
paliourg@iit.demokritos.gr

Evangelos Michelioudakis
vagmcs@iit.demokritos.gr

Alexander Artikis
a.artikis@unipi.gr

¹ Institute of Informatics and Telecommunications, National Centre for Scientific Research “Demokritos”, Athens, Greece

² Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece

³ Department of Maritime Studies, University of Piraeus, Athens, Greece