Online Structure Learning for Traffic Management

✉Evangelos Michelioudakis[1], Alexander Artikis[2,1], and Georgios Paliouras[1]

Institute of Informatics and Telecommunications, NCSR "Demokritos"[1]
Department of Maritime Studies, University of Piraeus[2]
`{vagmcs,a.artikis,paliourg}@iit.demokritos.gr`

Abstract. Most event recognition approaches in sensor environments are based on manually constructed patterns for detecting events, and lack the ability to learn relational structures in the presence of uncertainty. In this paper we describe the application of `OSL`α, a recently proposed online structure learner for Markov Logic Networks that can exploit Event Calculus axiomatizations, to proactive traffic management.

Keywords: Markov Logic Networks, Event Calculus, Uncertainty

1 Introduction

Many real-world applications are characterized by both uncertainty and relational structure. Regularities in these domains are hard to identify manually, and thus automatically learning them from data is desirable. One framework that concerns the induction of probabilistic knowledge by combining the powers of logic and probability is Markov Logic Networks (MLNs) [9]. Structure learning approaches that focus on MLNs have been successfully applied to a variety of applications where uncertainty holds. However, most of these approaches are batch and cannot handle large training sets due to their requirement to load all data in memory and carry out inference in each learning iteration.

Recently, we proposed the `OSL`α [7] online structure learner for MLNs, which extends OSL [5] by exploiting a given background knowledge to effectively constrain the space of possible structures during learning. The space is constrained subject to characteristics imposed by the rules governing a specific task, herein stated as axioms. As a background knowledge we are employing `MLN–EC` [11], a probabilistic variant of the Event Calculus [6, 8] for event recognition.

In event recognition [3, 1] the goal is to recognize *composite events* (CE) of interest given an input stream of *simple derived events* (SDEs). CEs can be defined as relational structures over sub-events, either CEs or SDEs, and capture the knowledge of a target application. Due to the dynamic nature of real-world applications, the CE definitions may need to be refined over time or the current knowledge may need to be enhanced with new definitions. Manual curation of event definitions is a tedious and cumbersome process and thus machine learning techniques to automatically derive the definitions are essential.

We have applied `OSL`α to learning definitions for traffic incidents using real sensor data provided in the context of the SPEEDD project[1]. The goal of SPEEDD is to develop a system for proactive, event-based decision-making, wherein decisions are triggered by forecasting events. In order to forecast and detect traffic incidents, `OSL`α constructs and refines the necessary CE definitions. Due to the high volume of the dataset, the learning process must employ an online strategy.

Section 2 provides a brief description of `OSL`α. Section 3 outlines the main challenges of learning traffic incident definitions, and reports the application of `OSL`α to the traffic domain.

2 `OSL`α: An Online Structure Learner using Background Knowledge Axiomatization

`OSL`α extends the procedure of OSL, by exploiting a given background knowledge. Figure 1 presents the components of `OSL`α. The background knowledge consists of the `MLN–EC` axioms (i.e., domain-independent rules) and an already known (possibly empty) hypothesis (i.e., set of clauses). Each axiom contains *query predicates* `HoldsAt` $\in \mathcal{Q}$ that consist the supervision and *template predicates* `InitiatedAt`, `TerminatedAt` $\in \mathcal{P}$ that specify the conditions under which a CE starts and stops being recognized. The latter form the target CE definitions that we want to learn. `OSL`α exploits these axioms in order to create mappings of supervision predicates into template predicates and search only for explanations of these template predicates. Upon doing so, `OSL`α does not need to search over time sequences, instead only needs to find appropriate bodies over the current time-point for the following definite clauses:

$$\texttt{InitiatedAt}(f,\, t) \Leftarrow \text{body}$$
$$\texttt{TerminatedAt}(f,\, t) \Leftarrow \text{body}$$

At any step t of the online procedure a training example (micro-batch) $\boldsymbol{\mathcal{D}}_t$ arrives containing simple derived events (SDEs), e.g. a fast lane in a highway has average speed less than 25 km/hour and sensor occupancy greater than 55%. $\boldsymbol{\mathcal{D}}_t$ is used together with the already learnt hypothesis to predict the truth values y_t^P of the composite events (CEs) of interest. This is achieved by (maximum a posteriori) MAP inference based on LP-relaxed Integer Linear Programming [4]. Given $\boldsymbol{\mathcal{D}}_t$ `OSL`α constructs a hypergraph that represents the space of possible structures as graph paths. Then for all incorrectly predicted CEs the hypergraph is searched, guided by `MLN–EC` axioms and path mode declarations [5] using relational pathfinding [10] up to a predefined length, for definite clauses explaining these CEs. The paths discovered during the search correspond to conjunctions of true ground atoms and are generalized into first-order clauses by replacing constants in the conjunction with variables. Then, these conjunctions are used

[1] `https://speedd-project.eu`

as a body to form definite clauses using as head the template predicate present in each path. The resulting set of formulas is converted into clausal normal form and evaluated. The weights of the retained clauses are then optimized by the AdaGrad online learner [2]. Finally, the weighted clauses are appended to the hypothesis $\boldsymbol{\mathcal{H}}_t$ and the procedure is repeated for the next training example $\boldsymbol{\mathcal{D}}_{t+1}$.

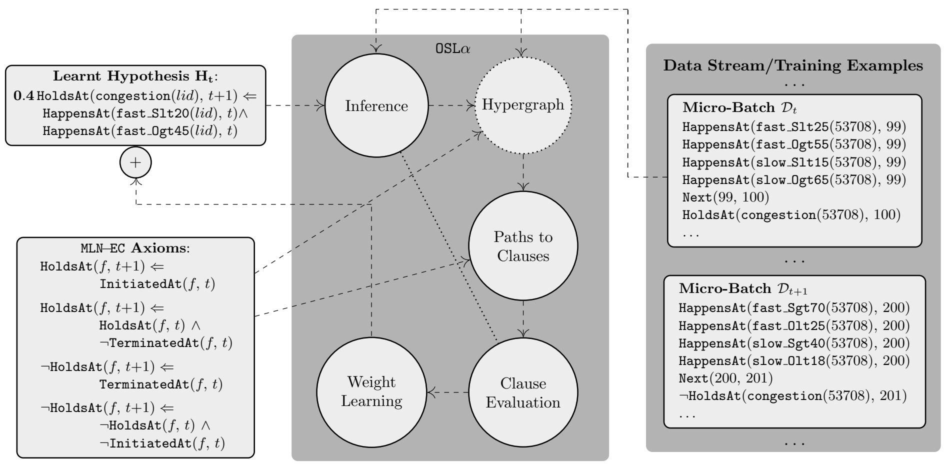
Learnt Hypothesis $\mathbf{H}_t$:
0.4 HoldsAt(congestion(lid), t+1) ⇐ HappensAt(fast_Slt20(lid), t)∧ HappensAt(fast_Ogt45(lid), t)
+
MLN–EC Axioms:
HoldsAt(f, t+1) ⇐ InitiatedAt(f, t)
HoldsAt(f, t+1) ⇐ HoldsAt(f, t) ∧ ¬TerminatedAt(f, t)
¬HoldsAt(f, t+1) ⇐ TerminatedAt(f, t)
¬HoldsAt(f, t+1) ⇐ ¬HoldsAt(f, t) ∧ ¬InitiatedAt(f, t)
OSLα
Inference
Hypergraph
Paths to Clauses
Clause Evaluation
Weight Learning
Data Stream/Training Examples
...
Micro-Batch $\mathcal{D}_t$
HappensAt(fast_Slt25(53708), 99)
HappensAt(fast_Ogt55(53708), 99)
HappensAt(slow_Slt15(53708), 99)
HappensAt(slow_Ogt65(53708), 99)
Next(99, 100)
HoldsAt(congestion(53708), 100)
...
...
Micro-Batch $\mathcal{D}_{t+1}$
HappensAt(fast_Sgt70(53708), 200)
HappensAt(fast_Olt25(53708), 200)
HappensAt(slow_Sgt40(53708), 200)
HappensAt(slow_Olt18(53708), 200)
Next(200, 201)
¬HoldsAt(congestion(53708), 201)
...
...

Fig. 1: The procedure of OSLα.

OSLα, AdaGrad and MAP inference based on LP-relaxed ILP are contributed to LoMRF[2], an open-source implementation of MLNs written in the Scala programming language. LoMRF is a tool that enables knowledge base compilation, grounding, inference and learning. It provides various features such as predicate completion, CNF compilation, function elimination and introduction, and a parallel grounding algorithm which constructs the minimal Markov Random Field (MRF) required for inference. The latter feature, in particular, is essential in temporal domains, such as event recognition.

3 Empirical Evaluation

We have applied OSLα to traffic management using real data from sensors mounted on the southern part of the Grenoble ring road (Rocade Sud). The dataset was made available by CNRS, our partner in the SPEEDD project, and consists of one month of sensor readings ($\approx$ 3.3GiB), where each day is annotated by human traffic controllers for traffic congestion. Sensors are placed in 19 collection points along a 12km stretch of the highway. Each collection point has a sensor per lane. Sensor data are collected every 15 seconds, containing the total number of vehicles passing through a lane, the average speed and sensor occupancy. These

[2] `https://github.com/anskarl/LoMRF`

readings constitute the simple derived events (SDEs), while traffic congestion is the target CE.

3.1 Learning Challenges

The first challenge of the problem at hand is the size of the dataset, that makes the use of batch learners, as well as online learners such as OSL that cannot make use of background knowledge, prohibitive. Second, traffic congestion annotation is largely incomplete, leading to the incorrect penalization of good rules. This issue is illustrated in Figure 2.

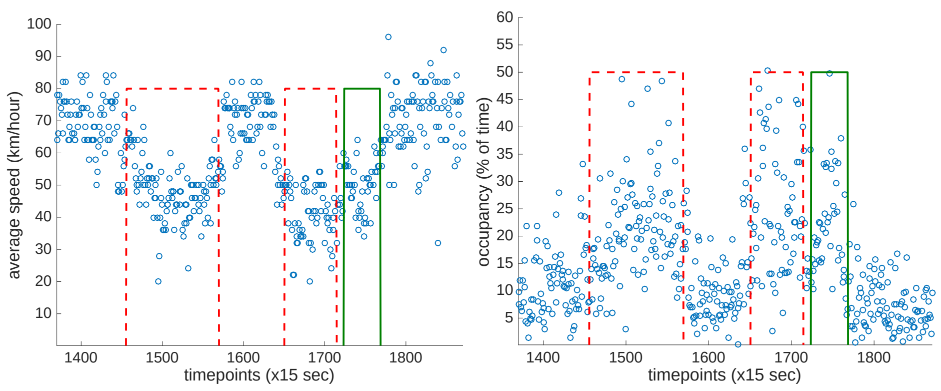

Fig. 2: Location 353708, fast lane: average speed (left) and occupancy (right). The blue points indicate the average speed (occupancy), the green windows indicate the congestion annotated by human experts, and red (dashed) windows the potentially missing annotations.

Third, the quality of information of each sensor differs considerably. This issue is illustrated in Figure 3, that displays the average speed of the fast lane and the queue lane of in same location, as well as the congestion annotation. In this case, the information provided by the sensors of the queue lane is largely uninformative.

Fourth, generic, location- and lane-agnostic rules are not sufficient. Consider, for example, a simple rule defining traffic congestion for any possible location regardless of the lane type:

$$\begin{aligned}&\texttt{InitiatedAt}(\texttt{congestion}(\mathit{lid}), t) \Leftarrow\\ &\quad \texttt{HappensAt}(\texttt{aggr}(\mathit{lid}, \mathit{occupancy}, \mathit{avgspd}), t) \land\\ &\quad \mathit{avgspd} < 50 \land \mathit{occupancy} > 25\end{aligned}$$

According to the above rule, a congestion in some location is said to be initiated if the average speed is below 50 km/hour and the occupancy is greater than 25%. Similar rules, not shown here to save space, terminate the recognition of congestion. The optimization of the weights of these rules had large fluctuations along the learning steps, leading to zero crossings, indicating that the rules correctly capture the concept of traffic congestion in a few locations, and completely fail in others. To deal with this issue, location- and lane-specific rules must be constructed.

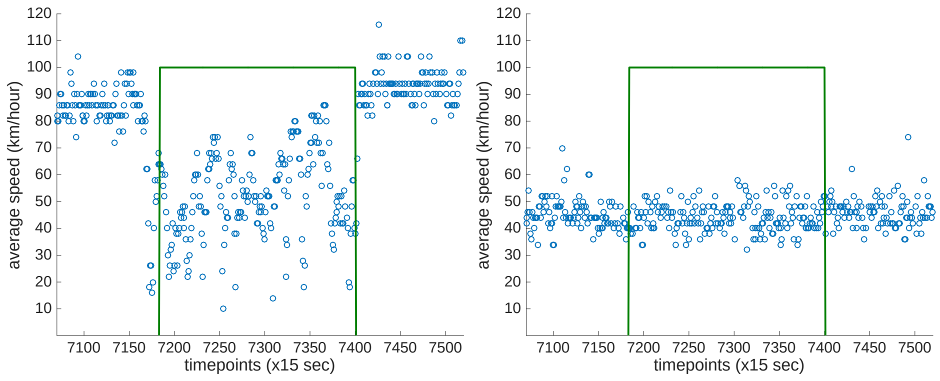

Fig. 3: Location 347549: fast lane (left) vs queue lane (right). The blue points indicate the average speed while green windows indicate the congestion annotated by human experts.

3.2 Experimental Setup and Results

The data are stored in a PostgreSQL database and the training sequence for each micro-batch, as shown in Figure 1, is constructed dynamically by querying the database. A set of first-order logic (FOL) functions is used to discretize the numerical data (speed, occupancy) and produce input events such as, for instance, `HappensAt(fast_Slt55(53708), 100)`, representing that the speed in the fast lane of location 53708 is less than 55 km/hour at time 100.

The CE supervision indicates when a traffic congestion holds in a specific location. Each training sequence is composed of input SDEs (`HappensAt`) over the FOL functions and the corresponding CE annotations (`HoldsAt`). The total length of the training sequence consists of 172799 timepoints. We consider only SDEs from fast lanes. We compare `OSL`α starting from an empty hypothesis against the AdaGrad [2] online weight learner, operating on manually constructed traffic congestion definitions developed by domain experts.

The evaluation results were obtained using MAP inference, as per [4] and are presented in terms of F_1 score. All reported statistics are micro-averaged over the instances of recognized CEs using 10-fold cross validation over the entire dataset using varying batch sizes. At each fold, an interval of 17280 timepoints was left out and used for testing. The experiments were performed in a computer with an Intel i7 4790@3.6GHz processor (4 cores and 8 threads) and 16GiB of RAM, running Ubuntu 16.04.

Figure 4 presents the evaluation results for `OSL`α and AdaGrad. In `OSL`α, the predictive accuracy of the learned model increases initially, due to the increase in the number of learning iterations, and then decreases, due to the decreasing batch size. On the contrary, the accuracy of AdaGrad increases (almost) monotonically as the number of learning iterations increase. `OSL`α achieves comparable predictive accuracy to the weighted manually curated rules (AdaGrad), which is encouraging. Moreover, it can process data batches efficiently. For example, `OSL`α takes $\approx$ 11 seconds to process a 50 minute batch (1400 SDEs). As expected, AdaGrad is faster than `OSL`α. The predictive accuracy of the learned model, both for `OSL`α and AdaGrad, is low. This arises mainly from the semi-supervised nature of the problem (see Section 3.1). Our experiments with fully-annotated simulated data from the Grenoble ring road, not shown here due to lack of space, confirmed this observation. Extending `OSL`α to deal with missing supervision is an area of current research.

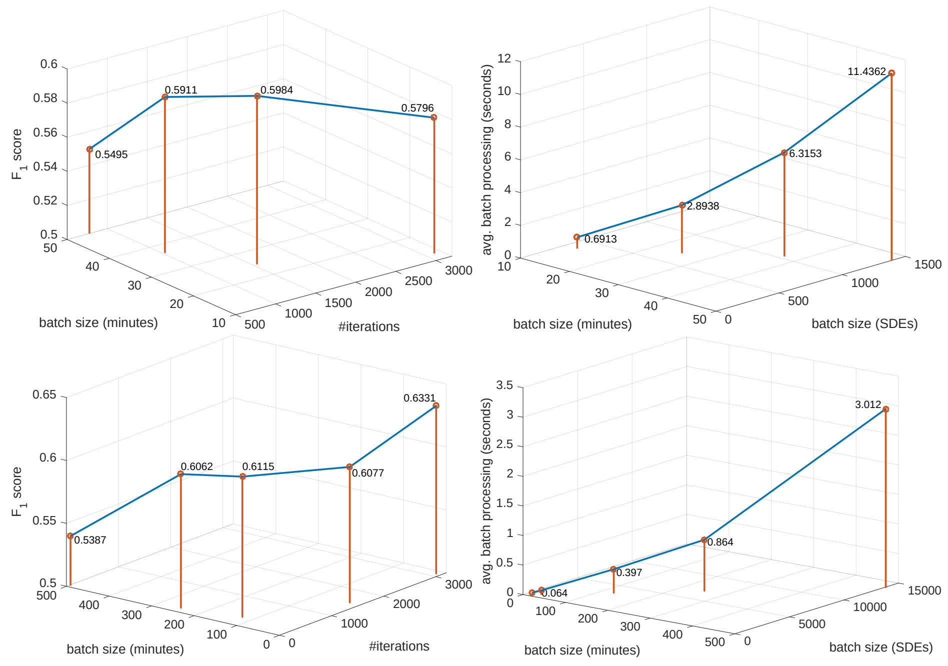

Fig. 4: F_1 score (left) and average batch processing time (right) for `OSL`α (top) and AdaGrad (bottom). In the left figures, the Y axis shows the number of learning iterations.

Acknowledgments. Funded by EU FP7 project SPEEDD (619435).

References

1. Artikis, A., Skarlatidis, A., Portet, F., Paliouras, G.: Logic-based event recognition. Knowledge Eng. Review 27(4): 469–506 (2012)
2. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res., 12, 2121–2159 (2011)
3. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. 44(3): 15 (2012)
4. Huynh, T.N., Mooney, R.J.: Max-Margin Weight Learning for Markov Logic Networks. In: Proc. of the ECML PKDD, 564–579 (2009)
5. Huynh, T.N., Mooney, R.J.: Online Structure Learning for Markov Logic Networks. In: Proc of the ECML PKDD, 81–96 (2011)
6. Kowalski, R.A, Sergot, M.J.: A Logic-based Calculus of Events. New Generation Comput. 4(1): 67–95 (1986)
7. Michelioudakis, E., Skarlatidis, A., Paliouras, G., Artikis, A.: OSLα: Online Structure Learning using Background Knowledge Axiomatization. ECML-PKDD (2016) `http://cer.iit.demokritos.gr/papers/OSLa.pdf`
8. Mueller, E.T.: Event Calculus. In: Handbook of Knowledge Representation, vol. 3 of Foundations of Artificial Intelligence, 671–708, Elsevier (2008)
9. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. (2006)
10. Richards, B.L., Mooney, R.J.: Learning relations by pathfinding. In Proc. of the 10th AAAI, 50–55 (1992)
11. Skarlatidis, A., Paliouras, G., Artikis, A., Vouros, G.A.: Probabilistic Event Calculus for Event Recognition. ACM Trans. on Comput. Logic, 16, 1–37 (2015)