

Nicolas Lachiche  
Christel Vrain (Eds.)

LNAI 10759

# Inductive Logic Programming

27th International Conference, ILP 2017  
Orléans, France, September 4–6, 2017  
Revised Selected Papers

 Springer

# Lecture Notes in Artificial Intelligence

10759

Subseries of Lecture Notes in Computer Science

LNAI Series Editors

Randy Goebel

*University of Alberta, Edmonton, Canada*

Yuzuru Tanaka

*Hokkaido University, Sapporo, Japan*

Wolfgang Wahlster

*DFKI and Saarland University, Saarbrücken, Germany*

LNAI Founding Series Editor

Joerg Siekmann

*DFKI and Saarland University, Saarbrücken, Germany*

More information about this series at <http://www.springer.com/series/1244>

Nicolas Lachiche · Christel Vrain (Eds.)

# Inductive Logic Programming

27th International Conference, ILP 2017  
Orléans, France, September 4–6, 2017  
Revised Selected Papers

*Editors*  
Nicolas Lachiche  
University of Strasbourg  
Strasbourg  
France

Christel Vrain  
University of Orléans  
Orléans  
France

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Artificial Intelligence  
ISBN 978-3-319-78089-4              ISBN 978-3-319-78090-0 (eBook)  
<https://doi.org/10.1007/978-3-319-78090-0>

Library of Congress Control Number: 2018937377

LNCS Sublibrary: SL7 – Artificial Intelligence

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG  
part of Springer Nature  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

This volume contains the revised versions of selected papers presented at the 27th International Conference on Inductive Logic Programming (ILP 2017). ILP 2017 was held in Orléans, France, during September 4–6, 2017.

Inductive logic programming (ILP) is a subfield of machine learning, which originally relied on logic programming as a uniform representation language for expressing examples, background knowledge, and hypotheses. Due to its strong representation formalism, based on first-order logic, ILP provides an excellent means for multi-relational learning and data mining, and more generally for learning from structured data. The ILP conference series, started in 1991, is the premier international forum for learning from structured or semi-structured relational data. Originally focusing on the induction of logic programs, over the years it has expanded its research horizon significantly and welcomes contributions to all aspects of learning in logic, including exploring intersections with probabilistic approaches.

Three kinds of papers were submitted, and the reviewing process was quite complicated:

1. Regular papers describing original mature work representing a self-contained theoretical contribution and/or supported by appropriate experimental evaluation. In all, 17 regular papers were submitted. These papers were reviewed by at least three members of the Program Committee. Seven papers were rejected. Ten papers were accepted for presentation at the conference. Although four were directly accepted for publication in the proceedings, only three are published in these proceedings. Six were invited to submit a revised version. After a second round of reviewing, four were accepted.
2. Late-breaking papers describing original work in progress, brief accounts of original ideas without conclusive experimental evaluation, and other relevant work of potentially high scientific interest but not yet qualifying for the regular paper category. In total, 14 late-breaking papers were accepted/rejected by the PC chairs, on the grounds of relevance, to be presented at the conference. Each late-breaking paper was reviewed by at least three members of the Program Committee taking also into account the oral presentation. This allowed us to nominate candidates for the most promising student late-breaking paper. Ten out of 14 late-breaking papers were invited to submit an extended version, which was evaluated a second time by three reviewers. Five of them were selected to be included in these proceedings.
3. Recently published papers. Five papers relevant to the conference topics and recently published or accepted for publication in a first-class conference were presented at the conference. These papers do not appear in this Springer LNAI conference proceedings.

In these proceedings, the articles are sorted according to the name of the first author. We identified several trends during this conference:

- Extension of the foundations of ILP (Bekker & Davis, Ribeiro et al., and Svatoš et al.)
- Parallelization (Katzouris et al., and Nishiyama & Ohwada)
- Applications of ILP to robotics, breast cancer and vision, respectively (Antanas et al., Côte-Real et al., and Dai et al.)
- A new trend exploring connections with deep learning (Dumančić et al., Kaur et al., Šourek et al., and Vig et al.)

We had the pleasure to welcome four invited speakers at ILP 2017:

- Alan Bundy, Professor at the University of Edinburgh: “Can Computers Change Their Minds?”
- Marc Boullé, Senior Researcher at Orange Labs: “Automatic Feature Construction for Supervised Classification from Large Scale Multi-Relational Data”
- Jennifer Neville, Associate Professor at Purdue University: “Learning from Single Networks—The Impact of Network Structure on Relational Learning and Collective Inference”
- Mathias Niepert, Senior Researcher at NEC Labs Europe in Heidelberg: “Learning Knowledge Base Representations with Relational, Latent, and Numerical Features”

Three prizes were awarded:

- Best paper (supported by Springer): Gustav Šourek, Martin Svatoš, Filip Železný, Steven Schockaert and Ondřej Kuželka. “Stacked Structure Learning for Lifted Relational Neural Networks”
- Best student paper (supported by *Machine Learning Journal*) Sebastijan Dumančić. “Demystifying Relational Latent Representations” (co-author Hendrick Blockeel)
- Most promising “late-breaking” student paper (supported by *Machine Learning Journal*): Laura Antanas. “Relational Affordance Learning for Task-Dependent Robot Grasping” (co-authors Anton Dries, Plinio Moreno, Luc de Raedt)

We would like to thank all the persons who contributed to the success of ILP 2017: the members of the Organizing Committee, the members of the Program Committee, the additional reviewers, and the sponsors.

February 2018

Nicolas Lachiche  
Christel Vrain

# Organization

## Organizing Committee

Christel Vrain (Chair)	University of Orléans, France
Guillaume Cleuziou	University of Orléans, France
Thi-Bich-Hanh Dao	University of Orléans, France
Matthieu Exbrayat	University of Orléans, France
Frédéric Moal	University of Orléans, France
Marcilio Pereira de Souto	University of Orléans, France
Isabelle Renard	University of Orléans, France

## Program Chairs

Nicolas Lachiche	University of Strasbourg, France
Christel Vrain	University of Orléans, France

## Program Committee

Dalal Alrajeh	Imperial College London, UK
Alexander Artikis	University of Piraeus, Greece
Hendrik Blockeel	KU Leuven, Belgium
Ivan Bratko	University of Ljubljana, Slovenia
Agnès Braud	University of Strasbourg, France
Krysia Broda	Imperial College London, UK
Rui Camacho	LIACC/FEUP University of Porto, Portugal
James Cussens	University of York, UK
Thi-Bich-Hanh Dao	University of Orléans, France
Saso Dzeroski	Jozef Stefan Institute, Slovenia
Floriana Esposito	Università degli Studi di Bari, Italy
Matthieu Exbrayat	University of Orléans, France
Nicola Fanizzi	Università degli Studi di Bari, Italy
Stefano Ferilli	Università degli Studi di Bari, Italy
Nuno A. Fonseca	EMBL-EBI, Hinxton, UK
Paolo Frasconi	Università degli Studi di Firenze, Italy
Katsumi Inoue	National Institute of Informatics, Japan
Ross King	University of Manchester, UK
Nada Lavrač	Jozef Stefan Institute, Slovenia
Francesca Alessandra Lisi	Università degli Studi di Bari, Italy
Donato Malerba	Università degli Studi di Bari, Italy
Stephen Muggleton	Imperial College London, UK
Sriiraam Natarajan	Indiana University, Bloomington, USA
Aline Paes	Universidade Federal Fluminense, Brazil



Bernhard Pfahringer	University of Waikato, New Zealand
Oliver Ray	University of Bristol, UK
Fabrizio Riguzzi	University of Ferrara, Italy
Céline Rouveirol	University of Paris 13, France
Alessandra Russo	Imperial College London, UK
Chiaki Sakama	Wakayama University, Japan
Takayoshi Shoudai	Kyushu International University, Japan
Henry Soldano	University of Paris 13, France
Ashwin Srinivasan	Birla Institute of Technology and Science, India
Stefan Wrobel	Fraunhofer IAIS and University of Bonn, Germany
Gerson Zaverucha	Federal University of Rio de Janeiro, Brazil
Filip Zelezny	Czech Technical University, Czech Republic

## Additional Reviewers

Giuseppe Cota	University of Ferrara, Italy
Nicola Di Mauro	Università degli Studi di Bari, Italy
Ana Luisa Duboc	Pedro II School, Brazil
Sebastijan Dumancic	KU Leuven, Belgium
Nikos Katzouris	NCSR Demokritos Athens, Greece
Evangelos Michelioudakis	NCSR Demokritos Athens, Greece
Andrea Pazienza	Università degli Studi di Bari, Italy

## Sponsors

We gratefully thank all the organizations and institutions that have supported this event:

- University of Orléans
- Laboratoire d'Informatique Fondamentale d'Orléans (LIFO)
- ICVL Fédération de recherche Informatique Centre Val de Loire
- Springer
- *Machine Learning* - Springer
- *Artificial Intelligence*
- EGC Association Internationale Francophone d'Extraction et de Gestion des Connaissances
- AFIA Association Française pour l'Intelligence Artificielle
- Région Centre - Val de Loire - French administrative region
- Département du Loiret - Loiret department
- Orléans métropole - Orléans city



# Contents

Relational Affordance Learning for Task-Dependent Robot Grasping. . . . .	1
<i>Laura Antanas, Anton Dries, Plinio Moreno, and Luc De Raedt</i>	
Positive and Unlabeled Relational Classification Through Label Frequency Estimation . . . . .	16
<i>Jessa Bekker and Jesse Davis</i>	
On Applying Probabilistic Logic Programming to Breast Cancer Data . . . . .	31
<i>Joana Côrte-Real, Inês Dutra, and Ricardo Rocha</i>	
Logical Vision: One-Shot Meta-Interpretive Learning from Real Images . . . .	46
<i>Wang-Zhou Dai, Stephen Muggleton, Jing Wen, Alireza Tamaddoni-Nezhad, and Zhi-Hua Zhou</i>	
Demystifying Relational Latent Representations . . . . .	63
<i>Sebastijan Dumančić and Hendrik Blockeel</i>	
Parallel Online Learning of Event Definitions . . . . .	78
<i>Nikos Katzouris, Alexander Artikis, and Georgios Paliouras</i>	
Relational Restricted Boltzmann Machines: A Probabilistic Logic Learning Approach. . . . .	94
<i>Navdeep Kaur, Gautam Kunapuli, Tushar Khot, Kristian Kersting, William Cohen, and Sriraam Natarajan</i>	
Parallel Inductive Logic Programming System for Superlinear Speedup . . . . .	112
<i>Hiroyuki Nishiyama and Hayato Ohwada</i>	
Inductive Learning from State Transitions over Continuous Domains. . . . .	124
<i>Tony Ribeiro, Sophie Touret, Maxime Folschette, Morgan Magnin, Domenico Borzacchiello, Francisco Chinesta, Olivier Roux, and Katsumi Inoue</i>	
Stacked Structure Learning for Lifted Relational Neural Networks . . . . .	140
<i>Gustav Šourek, Martin Svatoš, Filip Železný, Steven Schockaert, and Ondřej Kuželka</i>	
Pruning Hypothesis Spaces Using Learned Domain Theories . . . . .	152
<i>Martin Svatoš, Gustav Šourek, Filip Železný, Steven Schockaert, and Ondřej Kuželka</i>	

An Investigation into the Role of Domain-Knowledge  
on the Use of Embeddings . . . . . 169  
*Lovekesh Vig, Ashwin Srinivasan, Michael Bain,  
and Ankit Verma*

**Author Index** . . . . . 185



# Relational Affordance Learning for Task-Dependent Robot Grasping

Laura Antanas<sup>1</sup>(✉), Anton Dries<sup>1</sup>, Plinio Moreno<sup>2</sup>, and Luc De Raedt<sup>1</sup>

<sup>1</sup> Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium  
laura.antas@cs.kuleuven.be, laura.antas@gmail.com

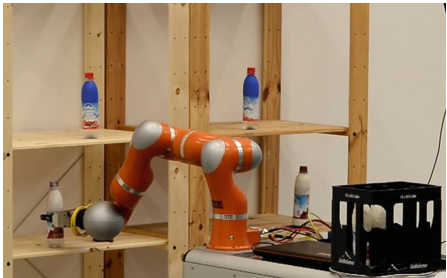
<sup>2</sup> Institute for Systems and Robotics, IST, University of Lisboa, Lisbon, Portugal

**Abstract.** Robot grasping depends on the specific manipulation scenario: the object, its properties, task and grasp constraints. Object-task affordances facilitate semantic reasoning about pre-grasp configurations with respect to the intended tasks, favoring good grasps. We employ probabilistic rule learning to recover such object-task affordances for task-dependent grasping from realistic video data.

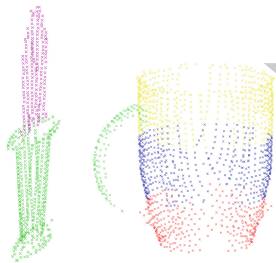
## 1 Introduction

Robot grasping skills are essential for acting in dynamic environments. Objects can be grasped in different ways depending on the specific manipulation scenario: the object, its properties, task and grasp constraints. Inspired by the definition of object affordances – which refers to the properties of an object to allow actions to be performed on it by a human or other entity, we investigate the benefits of object-task affordances for task-dependent grasping in a kitchen environment. Our earlier work on task-dependent grasping [2] shows that, when combined with probabilistic reasoning and object/task ontologies, they facilitate compact grasping models which generalize over object/task categories in a natural way, while showing robustness to uncertainty and missing information. Here we propose, as key contribution, a statistical relational learning approach to learn object affordances for task-dependent grasping. We employ ProbFOIL+ [7] to realize it.

Let us consider the scenario in Fig. 1. A mobile robot with grasping capabilities must grasp a bottle from the shelf and place it on the table. The environment constraints (e.g. narrow spaces) and task constraints (e.g. the most stable pre-grasp gripper pose for grasping the bottle) present a difficult problem which can be solved using semantic reasoning. If we consider the top, middle and bottom as semantic parts of the bottle, the best part to grasp it is from the middle, given that it needs to be placed on the table upright and the top is partially obstructed by the shelf above. Given such semantic object parts (or pre-grasps), object properties, and the intended task, we can learn probabilistic grasp-related rules for our kitchen scenario, e.g., that a bottle affords pick and placing on a surface by grasping it from the middle. The resulting task-dependent affordances give the robot the capability to semantically reason about the best pre-grasp and



**Fig. 1.** Manipulation scenario: grasp the bottle from the shelf and place it upright on the table.



**Fig. 2.** Semantic parts for knife and cup: yellow-top, blue-middle, red-bottom, green-handle, and magenta-usable area. (Color figure online)

thus, help the grasp planner. Our experiments show that we can learn reliable relational affordances from realistic and uncertain video data (Fig. 2).

## 2 Related Work

Much recent work focuses on incorporating task constraints in robot grasping by learning a direct mapping function between good grasps and various constraints (on actions and geometry), action features and object attributes [9–11, 17, 18, 23]. We extend this work by considering either object categorical information as an additional feature to predict suitable task-dependent grasping constraints or a task-dependent setting that uses probabilistic logic and world knowledge to reason about best pre-grasps.

Affordances have been considered before in robot manipulation. While in [24] the authors employ estimated visual-based latent affordances, the work in [4] reasons about grasp selection by modeling affordance relations between objects, actions and effects using either a fully probabilistic setting or a rule-based ontology. In contrast, we employ a SRL approach to learn object affordances which generalize over similar object parts and object/task categories. Closely related is the semantic grasping pipeline in [5]. It employs a semantic affordance map which relates gripper approach directions to particular tasks. We exploit additional world knowledge in form of ontologies. This allows us to experiment with a wide range of object categories. Further, to compute plans comprising sequences of actions and to solve complex manipulation tasks, [1] combines symbolic reasoning and learning from demonstrations. In [14] meaningful symbolic relational representations are used to solve sequential manipulation tasks in a goal-directed manner via active relational reinforcement learning. Relational Markov networks have been extended to build relational object maps for mobile robots in order to enable reasoning about hierarchies of objects and spatial relationships amongst them [16]. Related work for generalizing over doors and handles using SRL has been proposed in [19].

However, none of these frameworks solves the problem of learning affordances for semantic task-dependent grasping. Relational affordance models for robots have been learned in a multi-object manipulation task context [20]. Differently, we propose learning pre-grasp configurations using task-category affordances. Our approach features semantic generalization and can tackle unknown objects. This research topic has great importance in robotics as robots aimed at working in daily environments should be able to manipulate many never-seen-before objects and to deal with increasingly complex scenarios.

The paper is structured as follows. The next section introduces the relational problem of affordance learning. Subsequently, our SRL approach is described. After the experiments section, follow the concluding notes.

### 3 Problem Description and Representation

Each scene contains one object and the task to be executed. Its semantic visual description consists of the task, object parts, category, pose, and containment together with their probabilities. In a kitchen scenario, the perception algorithm proposed in [6] can segment objects, distinguish between upright and sideways poses and label each part with one of the labels: top, middle, bottom, handle or usable area. This reduces the search space for robot grasp generation, prediction and planning. The object category can be obtained using any object classifier. However, due to good results for grasping point prediction, we employ the manifold-based graph kernel approach proposed in [21]. It ensures a good appearance-based predictor for the object category. The prediction has the form of a probability distribution on object categories. Our kitchen setup considers 11 object categories:  $\{pan, pot, cup, glass, bowl, bottle, can, hammer, knife, screwdriver, cooking\_tool\}$ . We pick the category with the highest probability to characterize the object in the grasping scenario.

Further, our kitchen setup includes a set of 7 tasks:  $\{pass, pourOut, pourIn, pickPlaceInUpright, pickPlaceInUpsidedown, pickPlaceInSideways, pickPlaceOn\}$ . The task *pass* refers to grasping and passing the object to a human in the exact same pose, the tasks *pourOut* and *pourIn* to the actions of pouring liquid out of and inside the object, respectively, after grasping it. Tasks *pickPlaceInUpright*, *pickPlaceInUpsidedown* and *pickPlaceInSideways* refer to picking the object from the current pose and placing it inside a shelf in the upright, upside-down and sideways poses, respectively. Finally, the task *pickPlaceOn* is defined as picking and placing the object on a surface in the same initial pose.

The scene is represented as a set of relational visual observations. For the scenario in Fig. 1 they are encoded using probabilistic facts, such as  $1.0 :: \text{object}(o)$ , stating that an object  $o$  is observed with probability 1.0. The observation  $\text{object}(o)$  is a logical atom, while  $\text{object}/1$  is a predicate symbol of arity 1. The object identifier  $o$  is a constant and represents a ground term. Terms can also be variables when denoted in uppercase. Ground atoms or facts, such as  $\text{object}(o)$  and  $\text{part}(o, p1, \text{top})$ , do not contain variables and represent particular relations. They possess truth-values. Relational visual observations for our

scenario are illustrated in Example 1. We consider that the task is given and not observed, thus it has probability 1.0. We represent it as a probabilistic ground term, e.g., `1.0::task(o,t1,pickPlaceOn)`.

*Example 1.* Relational representation for our scenario in Fig. 1:

```

1.0::object(o).
0.8::category(o,bottle).
0.5::pose(o,upright).
0.9::contains(o,full).
0.5::part(o,p1,top).
0.9::part(o,p2,middle).
0.5::part(o,p3,bottom).

1.0::task(o,t1,pourOut).
1.0::task(o,t2,pass).
1.0::task(o,t3,pourIn).
1.0::task(o,t4,pickPlaceInUpsidedown).
1.0::task(o,t5,pickPlaceInUpright).
1.0::task(o,t6,pickPlaceInSideways).
1.0::task(o,t7,pickPlaceOn).

1.0::affords(o,t1).
1.0::affords(o,t2).
0.0::affords(o,t3).
...
1.0::affords(o,t7).

0.0::impossible(o,t1).
0.0::impossible(o,t2).
1.0::impossible(o,t3).
1.0::impossible(o,t4).
...
0.0::impossible(o,t7).

0.1::grasp(o,t1,p1).
1.0::grasp(o,t1,p2).
0.01::grasp(o,t1,p3).
0.5::grasp(o,t2,p1).
1.0::grasp(o,t2,p2).
0.01::grasp(o,t2,p3).
0.01::grasp(o,t3,p1).
0.01::grasp(o,t3,p2).
0.01::grasp(o,t3,p3).
...
0.5::grasp(o,t7,p1).
1.0::grasp(o,t7,p2).
0.01::grasp(o,t7,p3).

```

### 3.1 Object Category-Task (CT) Affordances and Constraints

We define an object-task affordance as the task afforded by an object category considered in our robot grasping setup. We keep in mind the manipulation capabilities of the gripper mounted on a robotic arm, in our case a KUKA LightWeight Robot (LWR) with two fingers [22]. Figure 3 illustrates a set of 46 common sense affordances marked with  $\checkmark$  in the form of a table. They allow us to relate object-task concepts based on human experience and inspired by *AfNet: The Affordance Network* ([www.theaffordances.net](http://www.theaffordances.net)). By looking at the table, we can extract possible object-task affordance pairs which can be encoded as logical rules. For example the rule `affords(X,T) ← bottle(X),task(T,pass)` states that a bottle indicated by variable X affords the passing task indicated by variable T. The set of affordances can be extended to include new object or task categories.

We can further make abstraction of fine-grained object categories by plugging in an object category ontology as in Fig. 4 (top). The super-categories in the ontology are defined based on the object functionality, and are represented by:  $\{kitchenContainer, dish, openContainer, canister, container, tool, object\}$ . For example, the super-category *dish* subsumes the categories *bowl*, *glass* and *cup*. Similarly, tasks can be grouped in super-tasks such as:  $\{pickPlaceIn, pickPlace,$

affordances task/object		container						tool				
		open container				canister						
		dish			kitchen							
		cup	glass	bowl	pan	pot	bottle	can	hammer	knife	screwdr	cooking
pass		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
pour	in	✓	✓	✓	✓	✓	-	-	-	-	-	
	out	✓	✓	✓	-	-	✓	✓	-	-	-	
p&p	in	upright	✓	✓	✓	✓	✓	✓	-	-	-	-
		upside-down	✓	✓	✓	-	-	-	-	-	-	-
		sideways	-	-	-	-	-	-	-	✓	✓	✓
	on	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Fig. 3. Object-task affordances are marked by ✓, constraints by -.

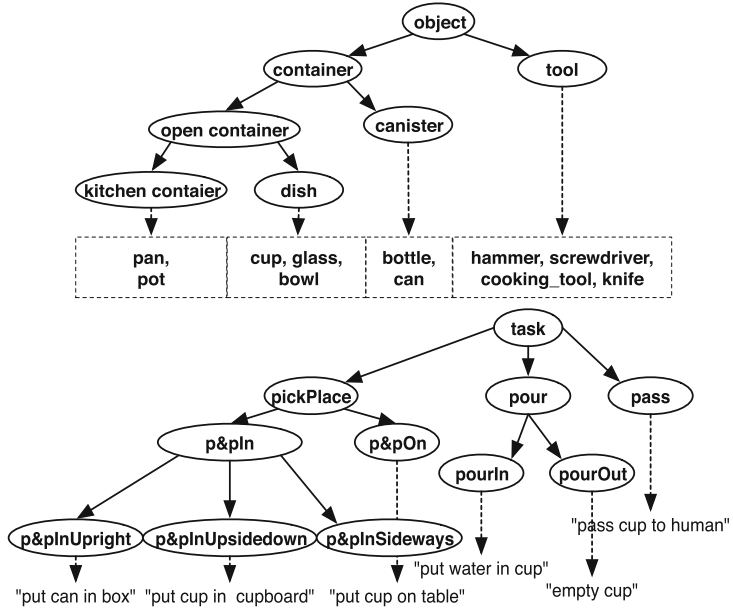


Fig. 4. Object category ontology (top) and task ontology (bottom).

*pour*, *task*} (Fig. 4 bottom). The super-task *pour* refers to the action of pouring the liquid in or out, while the super-task *pickPlaceIn* subsumes the tasks *pickPlaceInUpright*, *pickPlaceInUpside-down* and *pickPlaceInSideways*. The benefit of exploiting ontological structure is that we can make abstraction of the fine-grained object categories and tasks. Ontologies are symbolic high-level knowledge



which allows a compact representation of the affordance model by generalizing over similar object and task categories in a natural and straightforward way. As a result, they allow us to experiment with a wide range of objects and better deal with missing, uncertain or inaccurate information.

Ontologies can be translated into deterministic logical rules and directly used by our learner. For example, `supercategory(X, container) ← category(X, bottle)` states that “any bottle is a container”, `pour(T) ← pourIn(T)` specifies that “any task of pouring liquid in to fill some object is a pouring task”. The arguments `X` and `T` are variables and indicate the object identifier and task, respectively. We can then generally state that any container affords the task of pouring, i.e., `affords(X, T) ← container(X), pour(T)`. However, this is not always true, as pouring liquid in a canister is an almost impossible task, even for a human. We encode such constraints via the *impossible/2* predicate. The rule `impossible(X, T) ← canister(X), pourIn(T)` states that a canister does not afford the task of pouring in. Constraints are marked in Fig. 3 by `-`. Similar to affordances, constraints can be also generalized by making use of the ontological information, as the aforementioned constraint. Another example is `impossible(X, T) ← container(X), pickPlaceInSideways(X, T)` which indicates that a container should not be placed sideways.

A first goal of this work is to improve robot grasping by learning relational object-task affordances and constraints from data. This is done by specifying two separate learning problems. The CT affordance learning problem is indicated by keeping as learning target the `affords(X, T)` predicate, while the CT constraint problem via the target predicate `impossible(X, T)`.

### 3.2 Object Part-Category-Task (PCT) Affordances

Further, depending on the object properties, its parts and task, the object should be grasped in different ways. To reason about good pre-grasp configurations given the intended task, we use semantic object parts. Similar to object categories, pre-grasps can be associated to specific tasks. Each task activates grasping affordances according to associations between object categories, object parts and gripper poses. Besides object category-task associations, the second goal of our work is to learn object part-category-task relations. While the first are general pair-wise affordances, the second are grasp-related triplets that may rely on the first. For example, the rule `grasp(X, T, P) ← affords(X, T), pickPlaceInUpsidedown(T), glass(X), pose(X, upsidedown), part(X, P, bottom)` states that a glass `X` in the upside-down pre-grasp pose affords the task `T` of picking and placing inside the cupboard in an upside-down post-grasp pose by grasping it from the bottom. The PCT affordance learning problem is specified via the target predicate `grasp(X, T, P)`. We can make abstraction of fine-grained object categories and tasks by plugging in the object category and task ontologies here as well. Most of the times we can state that any dish in an initial upside-down pose can be picked and placed inside a cupboard in any pose by grasping it from the bottom, i.e., `grasp(X, T, P) ← affords(X, T), pickPlaceIn(T), dish(X), pose(X, upsidedown), part(X, P, bottom)`. Thus, the introduction of super-categories and super-tasks

reduces considerably the number of rules rendering much more compact model which are easier to interpret.

### 3.3 The Affordance Learning Problem

Using the visual observations introduced in Example 1 one can learn several affordances, e.g., that the bottle affords pick and placing on a surface in the initial upright pose and, in this case, it should be grasped from the middle or from the top (from the bottom the gripper might hit the shelf below, from the top a bit difficult to grasp given that the bottle is full), that it affords pouring out, and, in this case the bottle should be grasped from the middle (from the bottom the gripper might hit the shelf below, from the top it is rather difficult even for a human to pour out). Further, constraints can be also inferred, e.g., the bottle cannot be poured in liquid or placed upside-down. In order to do so, we assume to have labeled examples: object category-task relations specified via the target predicate `affords/2`, object category-task constraints via `impossible/2`, and object part-category-task affordances indicated by the target predicate `grasp/3`.

Ground target predicates or learning examples for each problem are illustrated in Example 1. Each learning problem is tackled in turn. Every learning example is a fact labeled with a target probability. In our scenario, target atom `1.0::affords(o,t1)` states that bottle `o` allows pouring out with maximum probability and is a learning example for the CT learning problem. Target label `1.0::grasp(o,t1,p2)` is a learning example for the PCT problem and asserts that the bottle can be grasped by the middle part `p2` with probability 1.0. The resulting set of probabilistic ground facts from all the scenarios corresponding to one learning problem represent input data for our probabilistic rule learner. For example, for the CT affordance problem, the learner takes as input features all grounded object, category, pose, containment and task predicates and the target `affords/2` predicates, while for the PCT affordance problem it takes, in addition as input features all grounded parts and as targets the `grasp/3` predicates. In our learning from entailment setting, probabilistic ground targets are positive learning examples. Using ProbFOIL+, we obtain negative examples automatically by taking combinations of possible values for the target arguments. A sampling step is performed such that the number of negatives balances the number of positives.

## 4 Approach: Probabilistic Rule Learning

Given the representation of our input and output we employ probabilistic rule learning for affordance learning. The learned probabilistic rules would have the form  $x :: \text{target} \leftarrow \text{body}$ , where the target is represented, for example, by the predicate `affords(X,T)` in the CT affordance learning problem and the body is represented by the set of pre-grasp configurations w.r.t the object category, pose, containment and task. The set of rules obtained are used to predict target predicates. We proceed with learning from entailment since our examples are facts probabilistically entailed by the theory. This setting is incorporated

in the probabilistic rule learner ProbFOIL+. It combines the principles of the rule learner FOIL with the probabilistic Prolog called ProbLog [12] and is capable of learning probabilistic rules from probabilistic ground input facts. Because ProbFOIL+ is a natural probabilistic extension of ILP and rule learning with respect to probabilistic data, we employ it to learn relational affordances. ProbFOIL+ generalizes FOIL, nFOIL [15], mFOIL [13] and ProbFoil [8]. Its output is a probabilistic classifier in the form of a set of generalized rules that return a probabilistic target atom.

The ProbFOIL+ algorithm directly generalizes the mFOIL rule learner. It follows a typical sequential covering approach where the outer loop of the algorithm starts from an empty set of clauses and repeatedly adds clauses to the hypothesis until no more improvements are observed with respect to some global scoring function (e.g. accuracy, recall or F1-measure). The clause to be added is obtained in a greedy manner by performing beam search using  $m$ -estimate such that it maximizes a local scoring function using a refinement operator. Each clause is learned in a greedy manner by performing beam search using  $m$ -estimate as a local scoring function. What sets ProbFOIL+ apart from mFOIL is its support for probabilistic data by generalizing the concepts of true/false positive/negative to a probabilistic context. In addition, it performs an additional step of parameter learning that allows it to learn rules that express probabilistic relationships.

While ProbLog and Prolog assume that the rules are definite clauses, in ProbFOIL+ we use probabilistic rules. We note that all facts for such rules are independent of one another, and the probability is determined by the rule learning algorithm. ProbFOIL+ uses versions of standard scoring functions for rule learning. As the global scoring function, which determines the stopping criterion of the outer loop, we use F1 measure. The local scoring function is based on the  $m$ -estimate, a variant of precision that is more robust against noise in the training data. Both metrics are based on the number of examples correctly classified as positive (true positives) and the number of examples incorrectly classified as positive (false positives) which are upgraded for use in a probabilistic setting. While in a deterministic setting, each example  $e_i$  has a 1/0 target classification, in ProbFOIL+ it has a probability value  $p_i$ . This means that every example contributes  $p_i$  to the positive part of the dataset and  $(1 - p_i)$  to the negative part of the dataset, which generalizes the deterministic setting with  $p_i = 1$  for positive and  $p_i = 0$  for negative examples. ProbFOIL+ defines the positive part of a dataset of size  $M$  as  $P = \sum_{i=0}^M p_i$  and the negative part as  $N = \sum_{i=0}^M 1 - p_i$ . The same approach generalizes the predictions of a model to the probabilistic setting where a hypothesis  $H$  will predict a value  $p_{H,i}$  for example  $e_i$  instead of 0 or 1. In this way the rule learner uses a probabilistic version of the true positive and false positive rates of the predictive model. If  $H$  overestimates the target value of  $e_i$ , that is,  $p_{H,i} > p_i$  then the true positive part will be maximal, that is, equal to  $p_i$ . The remaining part  $p_{H,i} - p_i$ , is part of the false positives. If  $H$  underestimates the target value of  $e_i$  then the true positive part is only  $p_{H,i}$

and the remaining part  $p_i - p_{H,i}$  contributes to the false negative part of the prediction.

In order to avoid learning large hypotheses with many clauses that only have limited contributions, ProbFOIL+ uses a significance test, used also by mFOIL. It is a variant of the likelihood ratio statistics. As a local stopping criteria for finding a viable next candidate, the clause must have a refinement that has a higher local score than the current best rule, has a significance that is high enough (according to a preset threshold), and has a better global score than the current rule set without the additional clause.

As input we provide ProbFOIL+ with the target predicate to be learned (e.g. `affords/2`) and a description of the refinement operator in terms of mode declarations. For example, `task(+, +, c)` indicates that the first two arguments should be variables that already exist in the clause, and the third argument is to be replaced by a constant. ProbFOIL+ then proceeds by iteratively extending the clause with one literal, pruning the least promising candidates at each step, until no more improvement can be made. We refer to the ProbFOIL+ paper for more details.

## 5 Experiments

We experiment on task-dependent robotic grasping datasets for kitchen-related scenarios introduced in [2]. We consider two datasets to quantitatively investigate the robustness and power of generalization of ProbFOIL+ for learning grasping affordances. The synthetic dataset denoted  $S_{SYN}$  considers flawless detection of objects from 3D meshes. The object points are distributed uniformly on the object surface according to their size by applying the midpoint surface subdivision technique. The object pose, its parts and object containment are manually labeled, while the object category is estimated using the global similarity classifier in [2]. The dataset is synthetic and actual grasps are not executed. It contains 41 objects belonging to all categories in our ontology and 102 grasping scenarios. This synthetic dataset serves as an upper-bound comparison scenario to the other realistic scenarios and allows an extensive evaluation of the generalization capabilities of the affordance learner.

The other dataset is obtained with the ORCA simulator [3] which provides sensors (laser range camera Asus Xtion PRO and the Universal Gripper WSG 50 force sensor), the robotic arm (KUKA LightWeight Robot (LWR)), objects and interface to a physics engine (Newton Game Dynamics library) for robot grasping simulation. The other modules that we use on top of ORCA, i.e., object completion, part and pose detection, category recognition and the tree-based motion planner (available in the Open Motion Planning Library), are external to ORCA and interfaced with the simulated robot. The datasets contain 25 objects belonging to all categories, except pot and cooking tool, and 134 grasping scenarios. We assume all containers empty. Each object is placed on top of a table. We obtain the dataset  $S_{REAL}$  by estimating object pose, category and its parts after the point cloud completion. It may have missing parts, when they

are occluded or not detected, or extra parts according to the limitations of the detection algorithm. The pose and parts have associated probabilities according to the limitations of the detection algorithms.

Our goal is to investigate if we can recover affordances from labeled data and to evaluate if these rules are good. In order to do so, we manually inspect the learned rules for the 3 datasets and compare them against the affordance table. Besides the number of correct rules recovered, we report recall, F1 measure, accuracy which are calculated based on classified probabilistic examples. The goal of this work is to focus on a more qualitative evaluation, and thus, we use all available data for training. Reported evaluation results are obtained by optimizing F1 measure on this data.

### 5.1 Results for Object Category-Task (CT) Affordances

We obtain CT affordances by specifying `affords/2` as the learning target. This gives a dataset of 714 examples for  $S_{SYN}$ , and 882 examples for  $S_{REAL}$ . As input information we consider two settings. In a first setting we use only object category, parts and task in order to assess the importance of the object category for affordances. In the second setting we add the pose and containment as well. Figure 5 shows part of learned rules on  $S_{SYN}$  for the first input setting by employing object fine-grained categories. A learned rule in the discovered set is `0.78 :: affords(A,B) ← category(A, cup), task(A, B, pourIn)`. We obtain 40 rules out of which 38 are fine-grained affordance rules (from 44 possible cf. Fig. 3) and an accuracy of 98% as Table 1 shows. We note that our dataset did not contain

<code>affords(A,B) ← task(A,B,pickPlaceOn), category(A,cup).</code>	0.73
<code>affords(A,B) ← task(A,B,pickPlaceOn), category(A,glass).</code>	0.54
<code>affords(A,B) ← task(A,B,pickPlaceOn), category(A,bowl).</code>	0.67
<code>affords(A,B) ← task(A,B,pickPlaceOn), category(A,pot).</code>	0.73
<code>affords(A,B) ← task(A,B,pickPlaceOn), category(A,bottle).</code>	0.58
<code>affords(A,B) ← task(A,B,pickPlaceOn), category(A,can).</code>	0.63
<code>affords(A,B) ← task(A,B,pickPlaceOn), category(A,knife).</code>	0.69
...	
<code>affords(A,B) ← category(A,cup), task(A,B,pickPlaceInUpsidedown).</code>	0.78
<code>affords(A,B) ← category(A,bowl), task(A,B,pickPlaceInUpsidedown).</code>	0.75
<code>affords(A,B) ← category(A,glass), task(A,B,pickPlaceInUpsidedown).</code>	0.82
<code>affords(A,B) ← category(A,cup), task(A,B,pourIn).</code>	0.78
<code>affords(A,B) ← category(A,bowl), task(A,B,pourIn).</code>	0.75
<code>affords(A,B) ← category(A,glass), task(A,B,pourIn).</code>	0.81
...	
<code>affords(A,B) ← supercategory(A,dish), task(A,B,pickPlaceOn).</code>	0.73
<code>affords(A,B) ← supercategory(A,canister), task(A,B,pickPlaceOn).</code>	0.54
<code>affords(A,B) ← supercategory(A,tool), task(A,B,pickPlaceOn).</code>	0.62
...	
<code>affords(A,B) ← supercategory(A,dish), task(A,B,pickPlaceInUpsidedown).</code>	0.82
<code>affords(A,B) ← supercategory(A,dish), task(A,B,pourIn).</code>	0.88

**Fig. 5.** Examples of CT affordances learned for  $S_{SYN}$  using object fine-grained categories (top) and super-categories (bottom).

**Table 1.** Number of learned rules, accuracy, F1 and recall for fine-grained categories and tasks. Input features used are object parts, category and executed task (without pose and containment).

Dataset	CT affordances		CT constraints		PCT affordances	
	$S_{SYN}$	$S_{REAL}$	$S_{SYN}$	$S_{REAL}$	$S_{SYN}$	$S_{REAL}$
Number of rules	40	15	42	41	33	22
Accuracy %	98	98	97	85	95	93
F1	0.86	0.86	0.82	0.83	0.52	0.54
Recall	0.87	0.77	0.77	0.76	0.37	0.58

**Table 2.** Input information, besides object parts, category and executed task, includes object pose and containment. Number of learned rules and accuracy are reported for fine-grained categories and tasks.

Dataset	CT affordances		CT constraints		PCT affordances	
	$S_{SYN}$	$S_{REAL}$	$S_{SYN}$	$S_{REAL}$	$S_{SYN}$	$S_{REAL}$
Number of rules	51	16	39	52	81	21
Accuracy %	98	98	98	98	97	96
F1	0.89	0.89	0.86	0.84	0.83	0.62
Recall	0.79	0.80	0.77	0.89	0.76	0.52

positive targets for pot-pourIn and pan-pourIn. The other affordances were not recovered because they depend also on the object initial pose and its containment. When we include them, the number of meaningful affordances learned does not increase. The learner discovers more category-task-pose (specialized CT affordances with the pose refinement), task-pose dependencies, but not new category-task affordances. This indicates that the pose is not so relevant for the CT pairs. The extra 2 input features do not notably impact the evaluation measures, which also proves, that the dataset, even synthetic, is not perfect. Next, using super-categories, we can summarize the set of 38 fine-grained affordances with 15 rules, while keeping the same accuracy, recall and F1-measure. Learned supercategory-task affordances are more general, specifically cup, glass, bowl are replaced by dish, bottle and can by canister, and hammer, screwdriver, cooking tool and knife by tool. Examples of more general rules obtained by ProbFOIL+ are illustrated in Fig. 5.

For the realistic dataset we can recover 35 fine-grained affordances out of 39 possible (cf. Fig. 3 without pot and cooking tool which are not included in the datasets) for  $S_{REAL}$ . We obtain 2 affordance rules for category pot which is not in the dataset (according to Fig. 3 one is correct, the other not), but none for *pickPlaceInUpside-down*. This is due to object misclassification. We note that 22 of these affordances are summarized by 2 rules, i.e.,  $0.44 :: \text{affords}(X, T) \leftarrow \text{task}(X, T, \text{pickPlaceOn})$

<code>affords(A,B) ← task(A,B,pickPlaceOn), supercategory(A,object).</code>	0.63
<code>affords(A,B) ← task(A,B,pass), supercategory(A,object).</code>	0.77
<code>affords(A,B) ← task(A,B,pickPlaceInsideSideways), supercategory(A,tool).</code>	0.81
<code>affords(A,B) ← task(A,B,pickPlaceInsideUpright), supercategory(A,dish).</code>	0.84
<code>affords(A,B) ← task(A,B,pickPlaceInsideUpright), supercategory(A,canister).</code>	0.86
<code>affords(A,B) ← task(A,B,pickPlaceInsideUpright), supercategory(A,openContainer).</code>	0.87
<code>affords(A,B) ← task(A,B,pourOut), supercategory(A,canister).</code>	0.87

**Fig. 6.** Examples of learned CT affordances for  $S_{REAL}$  using super-categories.

and  $0.72 :: \text{affords}(X, T) \leftarrow \text{task}(X, T, \text{pass})$ . The body has only the task predicate and no explicit object category predicate as the rule applies to all 11 object categories. This is due to lack of negative examples.

By using super-categories, we can replace the set of fine-grained rule with 7 generalized rules keeping similar evaluation values. The obtained rules are more general and can apply to new object categories. ProbFOIL+ does not learn again any rules for task `pickPlaceInUpsidedown` and `pourIn`. Examples of general rules using super-categories for  $S_{REAL}$  are illustrated in Fig. 6.

## 5.2 Results for Object Category-Task (CT) Constraints

To obtain CT constraints we give as target predicate `impossible/2`. It represents the opposite of `affords/2` probabilistically in the sense that what is affordable with a very small probability it is impossible with a high probability. We note that we obtain 42 constraint rules for  $S_{SYN}$  and 41 for  $S_{REAL}$  using fine-grained categories, without pose and containment. When we include the later 2 features, the model slightly improves in terms of rules, but also accuracy. A mistake that the learner returns is the constraint  $0.8 :: \text{impossible}(X, T) \leftarrow \text{task}(X, T, \text{pickPlaceInUpsidedown}), \text{category}(X, \text{cup})$ . This constraint holds only when the cup is full. If we include pose and containment, this constraint is removed. Looking at the evaluation results as well, we note that for CT constraints the initial pose of the object and its containment play a fairly important role. By using super-categories the learned affordance model reduces from 42 rules to 13 rules for  $S_{SYN}$  while keeping similar accuracy, F1 and recall. For  $S_{REAL}$  we obtain 18 rules instead of 41 with better evaluation values.

## 5.3 Results for Object Part-Category-Task (PCT) Affordances

The target to be learned is `grasp/3`. This gives us a dataset of 2093 examples for  $S_{SYN}$  and 2674 for  $S_{REAL}$ . Our setting considers as input information the task, object category, parts and pose, since the part from which to grasp an object for a given task highly depends on the pose as well, as Tables 1 and 2 show. Experiments using ProbFOIL+ give us a grasping model of 81 affordance rules for  $S_{SYN}$  and 21 rules for  $S_{REAL}$ . By introducing super-categories the grasp-based models are generalized from 81 rules to 31 and from 21 to 17, respectively, while keeping a close accuracy. A part-category-task affordance learned

```

grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,canister), part(A,C,middle). 0.78
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,canister), part(A,C,bottom), pose(A,upsideDown). 0.86
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,canister), part(A,C,top), pose(A,upright). 0.88
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,canister), part(A,C,top), pose(A,sideWAYS). 0.91
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,dish), part(A,C,middle). 0.85
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,dish), pose(A,upsideDown), part(A,C,bottom). 0.85
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,dish), pose(A,upsideDown), part(A,C,handle). 0.86
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,dish), pose(A,upright), part(A,C,top). 0.88
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,dish), pose(A,upright), part(A,C,middle). 0.81
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,dish), pose(A,sideWAYS), part(A,C,middle). 0.83
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,kitchenContainer), pose(A,sideWAYS), part(A,C,handle). 0.84
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,kitchenContainer), pose(A,upright), part(A,C,handle). 0.85
grasp(A,B,C)← stask(A,B,pickPlaceOn), scategory(A,tool), part(A,C,handle). 0.83
grasp(A,B,C)← stask(A,B,pickPlaceInsideUpsideDown), scategory(A,dish), pose(A,sideWAYS), part(A,C,middle). 0.95
grasp(A,B,C)← stask(A,B,pickPlaceInsideUpsideDown), scategory(A,dish), pose(A,upsideDown), part(A,C,middle).0.94
grasp(A,B,C)← stask(A,B,pourIn), scategory(A,dish), pose(A,upsideDown), part(A,C,middle). 0.92
grasp(A,B,C)← stask(A,B,pourIn), scategory(A,dish), pose(A,upsideDown), part(A,C,bottom). 0.94
grasp(A,B,C)← stask(A,B,pourIn), scategory(A,dish), pose(A,sideWAYS), part(A,C,middle). 0.94

```

**Fig. 7.** Examples of PCT affordances learned for  $S_{SYN}$  using super-categories and super-tasks.

from  $S_{SYN}$  is for example  $0.8:: \text{grasp}(A, B, C) \leftarrow \text{part}(A, C, \text{usable\_area}), \text{supercategory}(A, \text{tool}), \text{task}(A, B, \text{pass})$ . More learned rules are illustrated in Fig. 7.

## 5.4 Discussion

We note that learning PCT affordances is harder than learning CT affordances. First, this is due to the fact that the problem considers an extra input feature, i.e., the part, which adds additional noise. Although defined by hand, the  $S_{SYN}$  dataset is not perfect. There are many possible scenarios and constraints, and it is difficult, even by hand, to define learning instances without noise. For example, the containment, an input feature which is challenging to estimate and highly influences object grasping is not thoroughly considered across scenarios.  $S_{REAL}$ , much noisier than  $S_{SYN}$ , did not include the containment as input feature at all, although it is taken into account when defining the ground-truth. This is one of the reasons why the recall and F1 drop in Table 2 for  $S_{REAL}$  compared to  $S_{SYN}$ , and this holds for Table 1 as well.

Second, it becomes harder for  $S_{REAL}$  in the cases when the category was wrongly predicted (we used as input only the most likely object category although it is probably better to include the full object category distribution) and when the pose is difficult to estimate (it was mostly assigned by chance). It becomes difficult for  $S_{SYN}$  in Table 1 as well, when the pose is not considered. The object category and pose, as containment, play an important role.

## 6 Conclusions

Our previous experiments on robot grasping with respect to the intended high-level task confirm the importance of high-level reasoning and world knowledge



as opposed to using solely local shape information for robot grasping. The use of affordances and object/task ontologies plays a key role in obtaining better robot grasping. In this paper we propose a probabilistic rule learning approach to learn rule-based affordances that generalize over similar object parts and object/task categories and can be used to semantically reason in task-dependent robot grasping. Our experiments show that we can learn different reliable relational affordances from realistic data.

**Acknowledgements.** Partial support from the CHIST-ERA ReGROUND project on relational symbol grounding through affordance learning.

## References

1. Abdo, N., Kretzschmar, H., Stachniss, C.: From low-level trajectory demonstrations to symbolic actions for planning. In: ICAPS Workshop on Combining Task and Motion Planning for Real-World Applications, pp. 1–8 (2012)
2. Antanas, L., Moreno, P., Neumann, M., Pimentel de Figueiredo, R., Kersting, K., Santos-Victor, J., De Raedt, L.: High-level reasoning and low-level learning for grasping: a probabilistic logic pipeline. CoRR, abs/1411.1108 (2014)
3. Baltzakis, H.: Orca simulator
4. Barck-Holst, C., Ralph, M., Holmar, F., Kragic, D.: Learning grasping affordance using probabilistic and ontological approaches. In: ICRA, pp. 1–6 (2009)
5. Dang, H., Allen, P.K.: Semantic grasping: planning robotic grasps functionally suitable for an object manipulation task. In: IEEE/RSJ ICIRS, pp. 1311–1317 (2012)
6. de Figueiredo, R., Moreno, P., Bernardino, A.: Automatic object shape completion from 3D point clouds for object manipulation. In: Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications VISIGRAPP, pp. 565–570 (2017)
7. De Raedt, L., Dries, A., Thon, I., Van Den Broeck, G., Verbeke, M.: Inducing probabilistic relational rules from probabilistic examples. In: IJCAI, pp. 1835–1843. AAAI Press (2015)
8. De Raedt, L., Thon, I.: Probabilistic rule learning. In: Frasconi, P., Lisi, F.A. (eds.) ILP 2010. LNCS (LNAI), vol. 6489, pp. 47–58. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-21295-6\\_9](https://doi.org/10.1007/978-3-642-21295-6_9)
9. Detry, R., Ek, C.H., Madry, M., Kragic, D.: Compressing grasping experience into a dictionary of prototypical grasp-predicting parts. In: The 5th International Workshop on Human-Friendly Robotics, October 2012
10. Detry, R., Ek, C.H., Madry, M., Kragic, D.: Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In: ICRA, pp. 601–608 (2013)
11. Detry, R., Ek, C.H., Madry, M., Piater, J.H., Kragic, D.: Generalizing grasps across partly similar objects. In: ICRA, pp. 3791–3797 (2012)
12. Dries, A., Kimmig, A., Meert, W., Renkens, J., Van den Broeck, G., Vlasselaer, J., De Raedt, L.: ProbLog2: probabilistic logic programming. In: Bifet, A., May, M., Zadrozny, B., Gavalda, R., Pedreschi, D., Bonchi, F., Cardoso, J., Spiliopoulou, M. (eds.) ECML PKDD 2015. LNCS (LNAI), vol. 9286, pp. 312–315. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-23461-8\\_37](https://doi.org/10.1007/978-3-319-23461-8_37)
13. Džeroski, S.: Handling imperfect data in inductive logic programming. In: SCAI, pp. 111–125 (1993)

14. Kulick, J., Toussaint, M., Lang, T., Lopes, M.: Active learning for teaching a robot grounded relational symbols. In: IJCAI, pp. 1451–1457. AAAI Press (2013)
15. Landwehr, N., Kersting, K., De Raedt, L.: nFOIL: Integrating Naïve Bayes and FOIL. In: Proceedings of the 20th National Conference on Artificial Intelligence, AAAI 2005, vol. 2, pp. 795–800. AAAI Press (2005)
16. Limketkai, B., Liao, L., Fox, D.: Relational object maps for mobile robots. In: IJCAI, pp. 1471–1476 (2005)
17. Madry, M., Song, D., Ek, C.H., Kragic, D.: “Robot bring me something to drink from”: object representation for transferring task specific grasps. In: ICRA Workshop on Semantic Perception, Mapping and Exploration, pp. 1–6 (2012)
18. Madry, M., Song, D., Kragic, D.: From object categories to grasp transfer using probabilistic reasoning. In: ICRA, pp. 1716–1723 (2012)
19. Moldovan, B., Antanas, L., Hoffmann, M.E.: Opening doors: an initial SRL approach. In: Riguzzi, F., Železný, F. (eds.) ILP 2012. LNCS (LNAI), vol. 7842, pp. 178–192. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38812-5\\_13](https://doi.org/10.1007/978-3-642-38812-5_13)
20. Moldovan, B., Moreno, P., Nitti, D., Santos-Victor, J., De Raedt, L.: Relational affordances for multiple-object manipulation. *Autonomous Robots*, May 2017
21. Neumann, M., Moreno, P., Antanas, L., Garnett, R., Kersting, K.: Graph Kernels for object category prediction in task-dependent robot grasping. In: MLG, pp. 1–6 (2013)
22. KUKA robotics. Kuka lightweight robot (LWR)
23. Song, D., Huebner, K., Kyrki, V., Kragic, D.: Learning task constraints for robot grasping using graphical models. In: IEEE/RSJ ICIRS, pp. 1579–1585 (2010)
24. Sweeney, J., Grupen, R.A.: A model of shared grasp affordances from demonstration. In: *Humanoids*, pp. 27–35 (2007)



# Positive and Unlabeled Relational Classification Through Label Frequency Estimation

Jessa Bekker<sup>(✉)</sup> and Jesse Davis<sup></sup>

Computer Science Department, KU Leuven, Leuven, Belgium  
{jessa.bekker, jesse.davis}@cs.kuleuven.be

**Abstract.** Many applications, such as knowledge base completion and automated diagnosis of patients, only have access to positive examples but lack negative examples which are required by standard relational learning techniques and suffer under the closed-world assumption. The corresponding propositional problem is known as Positive and Unlabeled (PU) learning. In this field, it is known that using the label frequency (the fraction of true positive examples that are labeled) makes learning easier. This notion has not been explored yet in the relational domain. The goal of this work is twofold: (1) to explore if using the label frequency would also be useful when working with relational data and (2) to propose a method for estimating the label frequency from relational positive and unlabeled data. Our experiments confirm the usefulness of knowing the label frequency and of our estimate.

## 1 Introduction

Relational classification traditionally requires positive and negative examples to learn a theory. However, in many applications, it is only possible to acquire positive examples. A common solution to this issue is to make the closed-world assumption and assume that unlabeled examples belong to the negative class. In reality, this assumption is often incorrect, for example: knowledge bases are incomplete [1], diabetics often go undiagnosed [2] and people do not bookmark all interesting pages. Considering unlabeled cases as negative is therefore sub-optimal. To cope with this, several score functions have been proposed that use only positive examples [3–5].

In propositional settings, having training data with only positive and unlabeled examples is known as Positive and Unlabeled (PU) learning. It has been noted that if the class prior is known, then learning in this setting is greatly simplified. Specifically, knowing the class prior allows calculating the label frequency, which is the probability of a positive example being labeled. The label frequency is crucial as it enables converting standard score functions into PU score functions that can incorporate information about the unlabeled data [6]. Following this insight, several methods have been proposed to estimate the label frequency from PU data [7–10]. To the best of our knowledge, this notion has not been exploited in relational settings. We propose a method to estimate the

label frequency from relational PU data and a way to use this frequency when learning a relational classifier.

Our main contributions are: (1) Investigating the helpfulness of the label frequency in relational positive and unlabeled learning by extending two common relational classifiers to incorporate the label frequency; (2) Modifying TlCE, a method for estimating the label frequency in PU data, to operate with relational data, and (3) Evaluating our approach experimentally.

The paper is organized as follows: Sect. 2 gives an overview of related work. A general background on PU learning and how the label frequency simplifies this problem is presented in Sect. 3. Section 4 discusses how the label frequency can be estimated from PU data. The helpfulness of the label frequency and our method to estimate it from the data are empirically evaluated in Sect. 5. Finally, we conclude in Sect. 6.

## 2 Related Work

Positive and Unlabeled Learning has been studied mostly in propositional contexts. The proposed methods fall into four categories of approaches. The first and most straightforward approach is to use standard machine learning techniques and just assuming all the unlabeled examples to be negative. The second approach is to look for examples that are very different from the labeled ones and label these as negative. Subsequently, semi-supervised learning methods can be applied [11–15]. The third approach is to formulate an evaluation metric that has only positive or positive and unlabeled data as input and use this for tuning class weights or regularization settings [16–19]. The fourth approach is the approach considered in this paper. It explicitly uses the label frequency to modify traditional classification algorithms [6, 20–23].

In order to use the label frequency, it needs to be given as input. Note that the label frequency can be calculated from the class prior, because the proportion of labeled data is directly proportional to the class prior with the label frequency as the proportionality constant. The class prior could be known from domain knowledge or it could be estimated from a smaller fully labeled dataset. Because this knowledge or data is often not available, several methods have been proposed to estimate it directly from the positive and unlabeled data [6–10, 24].

A few relational positive and unlabeled learning methods exist. The method proposed by Muggleton follows the third positive and unlabeled learning approach and searches for the smallest hypothesis which covers all the positive examples [3]. If the underlying concept is complicated, this is likely to overgeneralize. RelOCC is a positive and unlabeled classification method that incrementally learns a tree-based distance measure which measures the distance to the positive class [25]. The SHERLOCK system is also related because it learns rules from positive examples by evaluating the rules on their statistical relevance and significance, however, it does not utilize the unlabeled data [5].

Knowledge base completion is inherently a positive and unlabeled problem: all the examples that are already in the knowledge base are positive and all the

additional facts that could be included are unlabeled [26]. However, many methods make a closed world assumption when learning models from the original knowledge base and assume everything that is not present to be false [27–30]. Recently, a new score function for evaluating knowledge base completion rules was proposed [1]. It approximates the true precision of a rule by assuming that the coverage of a rule is equal for labeled and unlabeled positives and by estimating the functionality of the relation.

### 3 PU Learning and the Label Frequency

This section gives some background on PU learning and how the label frequency is used to simplify learning. The methods for using the label frequency are then applied to popular relational classifiers. Table 1 presents the terminology used throughout the paper.

#### 3.1 Positive Unlabeled (PU) Learning

In traditional binary classification, learners are supplied with two types of examples: positive and negative ones. When learning from positive and unlabeled data, commonly referred to as PU learning, there are also two types of examples: positively labeled and unlabeled ones, where the latter can be either positive or negative.

In PU Learning, the labeled positive examples are commonly assumed to be ‘selected completely at random’ [6, 20–22, 31]. This means that the probability  $c = \Pr(s = 1|y = 1)$  for a positive example to be labeled is constant, which is the same for every positive example. The constant  $c$  is called the *label frequency*. Implicit techniques to employ the ‘selected completely at random’ property are to give more weight to the positive class or to model more noise in the negative class [16–18]. It can also be used explicitly by taking the label frequency into account when training a model [6, 20, 22, 31], which greatly simplifies learning

**Table 1.** Description of terminology used in the paper.

Term	Description
$y$	Indicator variable for an example to be positive
$s$	Indicator variable for an example to be labeled
$c$	Label frequency $\Pr(s = 1 y = 1)$
$\hat{c}$	Estimate of the label frequency
$P$	(Estimated) number of positive examples
$N$	(Estimated) number of negative examples
$L$	Number of labeled examples
$U$	Number of unlabeled examples
$T$	Total number of examples

because traditional classifiers can be adjusted to incorporate it in a straightforward manner. This is well-established knowledge for propositional PU learning, however, to the best of our knowledge, using the label frequency has not been investigated yet for relational PU learning. We briefly review some of the propositional methods and discuss how they can be adjusted to the relational setting.

### 3.2 Using the Label Frequency to Simplify PU Learning

Elkan and Noto propose to use the label frequency directly to modify traditional classifiers for PU learning [6]. Concretely, they proposed the following two methods:

1. **Probabilistic classifier modification:** This method trains a probabilistic classifier to predict the probability for instances to be labeled. To this end, during training, it considers unlabeled examples as negative. The label frequency is then employed to modify the output probabilities. It transforms the probability that an instance is labeled  $\Pr(s = 1|x)$  into the probability that an instance is positive:  $\Pr(y = 1|x) = \frac{1}{c} \Pr(s = 1|x)$  [22]. This modified classifier can be used directly or to transform the PU dataset into a probabilistically weighted PN dataset.
2. **Score function modification:** Learning algorithms that make decisions based on counts of positive and negative examples data subsets  $i$  can be modified to use counts of labeled and unlabeled examples. The positive and negative counts  $P_i$  and  $N_i$  can be obtained with  $P_i = L_i/c$  and  $N_i = T_i - P_i$ . Decision trees, for example, assign classes to leaves and score splits based on the positive/negative counts in the potential subsets and can, therefore, be transformed to PU learners [23].

In this paper, we demonstrate how to use these two methods in the relational domain. The proposed solutions  $c$ -adjusted TILDE and  $c$ -adjusted Aleph are described below.

#### **Relational Probabilistic Classifier Modification: $c$ -adjusted TILDE.**

The first method for using the label frequency requires a probabilistic classifier which predicts the probability that an instance is labeled. The first-order logical decision tree learner TILDE can easily be made probabilistic. Doing so simply requires counting for each leaf  $i$  the number of labeled  $L_i$  and unlabeled examples  $U_i$  that reach  $i$  setting the leaf's probability to  $\frac{L_i}{U_i+L_i}$  [32]. The tree that predicts the probability for instances to be positive has the same structure as the tree for distinguishing between labeled and unlabeled example, but requires altering the probability in each of its leaves. The new probability in each leaf  $i$  is  $\frac{1}{c} \frac{L_i}{U_i+L_i}$ , where  $L_i$  and  $U_i$  are defined as above.

**Relational Score Function Modification:  $c$ -adjusted Aleph.** The second method for using the label frequency requires a classifier that makes decisions based on the counts  $N_i$  and  $P_i$  in a subset of the data  $i$ . TILDE satisfies this criterion, and so does the rule learner Aleph [33]. The default evaluation function

of Aleph is coverage, which is defined as  $P_i - N_i$ , where  $i$  is the subset of examples that satisfy the rule. To modify Aleph to use the label frequency  $c$ , the coverage for each rule  $r$  should be calculated as follows:

$$\text{PU coverage} = P_i - N_i = 2P_i - T_i = 2\frac{L_i}{c} - T_i \quad (1)$$

where  $L_i$  is the number of labeled (i.e., positive) examples covered by the rule and  $T_i$  is the total number of examples covered by the rule.

## 4 Label Frequency Estimation

To estimate the label frequency in relational PU data, we will use the insights of a propositional label frequency estimator. We first review the original method and then propose a relational version.

### 4.1 Label Frequency Estimation in Propositional PU Data

The propositional estimator is TlCE [10]. It is based on two main insights: (1) a subset of the data naturally provides a lower bound on the label frequency, and (2) the lower bound of a large enough positive subset approximates the real label frequency. TlCE uses decision tree induction to find likely positive subsets and estimates the label frequency by taking the maximum of the lower bounds implied by all the subsets in the tree.

The label frequency is the same in subsets of the data because of the ‘selected completely at random’ assumption, therefore it can be estimated in a subset of the data. Clearly, the true number of positive examples  $P_i$  in a subset  $i$  cannot exceed the total number of examples in that subset  $T_i$ . This naively implies a lower bound:  $c = L_i/P_i \geq L_i/T_i$ . To take stochasticity into account, this bound is corrected with confidence  $1 - \delta$  using the one-sided Chebyshev inequality which introduces an error term based on the subset size:

$$\Pr \left( c \leq \frac{L_i}{T_i} - \frac{1}{2} \sqrt{\frac{1 - \delta}{\delta T_i}} \right) \leq \delta \quad (2)$$

The higher the ratio of positive examples in the subset, the closer the bound gets to the actual label frequency. The ratio of positive examples is unknown, but directly proportional to the ratio of labeled examples. Therefore, TlCE aims to find subsets of the data with a high proportion of labeled examples using decision tree induction. To avoid overfitting, i.e. finding subsets  $i$  where  $L_i/P_i > c$ ,  $k$  folds are used to induce the tree and estimate the label frequency on different datasets.

The parameter  $\delta$  is set such that at least one tenth of the data or 1000 examples are needed to estimate the label frequency with an error term of 0.1:  $1/2\sqrt{(1 - \delta)/(\delta T_R)} = 0.1$ , with  $T_R = \min[T/10, 1000]$ . This imposed by

$$\delta = \max \left[ 0.025, \frac{1}{1 + 0.004T} \right] \quad (3)$$

## 4.2 Label Frequency Estimation in Relational PU Data

We propose TlCER (Tree Induction for  $c$  Estimation in Relational data). The main difference with TlCE is that it learns a first-order logical decision tree using TILDE [32]. Each internal node splits on the formula which locally optimizes the gain ratio, considering the unlabeled examples as negative. The examples that satisfy the formula go to the left, the others to the right. Each node in the tree, therefore, specifies a subset of the data, and each subset implies a lower bound on the label frequency through (2). The estimate for the label frequency is the maximal lower bound implied by the subsets:

$$\hat{c} = \max_{i \in \text{subsets}} \left[ \frac{L_i}{T_i} - \frac{1}{2} \sqrt{\frac{1 - \delta}{\delta T_i}} \right] \quad (4)$$

To prevent overfitting,  $k$  folds are used to induce the tree and estimate the label frequency on different datasets. With relational data, extra care should be taken that the data in different folds are not related to each other. The final estimate is the average of the estimates made in the different folds.

## 5 Experiments

Our goal is to evaluate if knowing the label frequency makes learning from relational PU data easier and if TlCER provides a good estimate of the label frequency. More specifically, we will answer the following questions:

- Q1:** Does  $c$ -adjusted TILDE, the proposed relational probabilistic classifier modification method, improve over classic TILDE when faced with PU data and how sensitive is it to the correctness of  $\hat{c}$ ?
- Q2:** Does  $c$ -adjusted Aleph, the proposed relational score function modification method, improve over classic Aleph when faced with PU data and how sensitive is it to the correctness of  $\hat{c}$ ?
- Q3:** How well does TlCER estimate the label frequency? In which cases does it perform better or worse?
- Q4:** How do label frequency adapted methods compare with Muggleton’s PosOnly method?

### 5.1 Datasets

We evaluate our approach on four commonly used datasets for relational classification (Table 2). All datasets are available on Alchemy<sup>1</sup>, except for Mutagenesis.<sup>2</sup> The classes of WebKB are disjunctive concepts. Person contains web pages from students, faculty and staff and Other contains web pages from departments, courses, and research projects. To get an intuition of the complexity

<sup>1</sup> <http://alchemy.cs.washington.edu/data/>.

<sup>2</sup> <http://www.cs.ox.ac.uk/activities/machlearn/mutagenesis.html>.



**Table 2.** Characteristics of the Datasets

Datasets	#Examples	Class 1 (#)	Class 2 (#)	# Folds
IMDB	268	Actor (236)	Director (32)	5
Mutagenesis	230	Yes (138)	No (92)	5
UW-CSE	278	Student (216)	Professor (62)	5
WebKB	922	Person (590)	Other (332)	4

**Table 3. Dataset complexities:** The complexities of the models that are trained on the complete and fully labeled datasets. For TILDE, the number of splits in the tree is shown. For Aleph, the number of rules is reported and the average rule length is given in parentheses.

Dataset	TILDE	Aleph Class1	Aleph Class2
IMDB	1	1 (1)	1 (1)
Mutagenesis	6	7 (2.29)	8 (2.25)
UW-CSE	3	5 (1)	5 (1.4)
WebKB	33	32 (3.19)	38 (2.08)

of the concepts to be learned, Table 3 shows how big the TILDE and Aleph models are if they are trained on the complete dataset with labels for all examples. The datasets were converted to PU datasets by selecting some of the positive examples at random to be labeled. The labeling was done with frequencies  $c \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ . Each has five different random labelings.

## 5.2 Methods

For our experiments we used the following PU classifiers:

- $c$ -adjusted TILDE, as described in Sect. 3.2.
- $c$ -adjusted Aleph, as described in Sect. 3.2.
- Aleph, taking unlabeled examples as negative ( $\hat{c} = 1$ )
- TILDE, taking unlabeled examples as negative ( $\hat{c} = 1$ )
- PosOnly: Muggleton’s approach, implemented in Aleph [3].<sup>3</sup>

All classifiers, including TILDE when used for TICER, use standard settings, with the exceptions of requiring PosOnly rules to cover at least two example and allowing infinite noise and exploration in Aleph.

For the  $c$ -adjusted methods, an estimate of the label frequency  $c$  is required. This estimate  $\hat{c}$  can be the correct label frequency  $c$  or the estimate obtained by our method TICER. For the sensitivity experiments, the  $\hat{c}$  is varied in  $c \pm \Delta$  with  $\Delta \in \{0, 0.05, 0.15, 0.25\}$ . The naive baseline where unlabeled examples are

<sup>3</sup> <http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/aleph>.

considered to be negative can be seen as a special case of the  $c$ -adjusted methods with  $\hat{c} = 1$ .

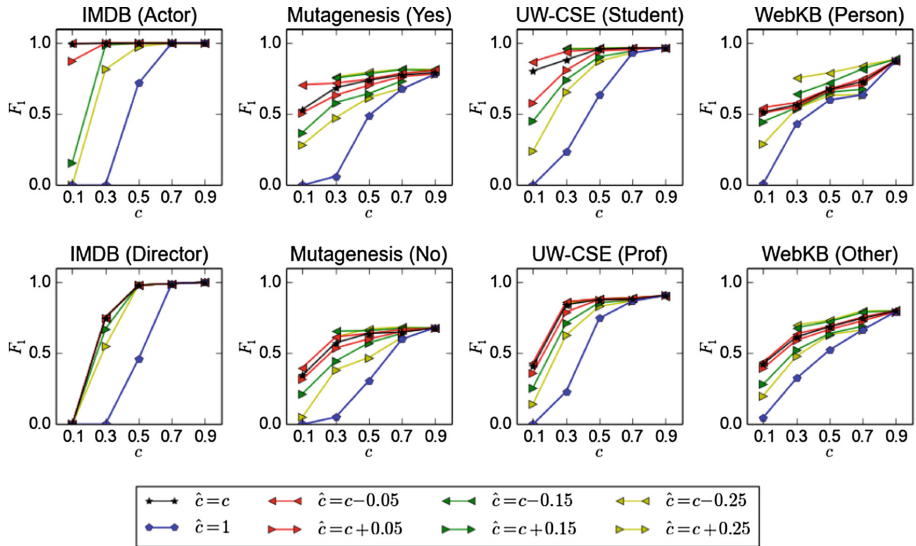
$k$ -fold cross-validation was applied for validation, i.e.,  $k - 1$  folds were used for learning the classifier and the other fold to evaluate it. TICER also needs folds for estimation, it used 1 fold for inducing a tree and the other  $k - 2$  folds for bounding the label frequency.

The classifiers are compared using the  $F_1$  score and the average absolute error of the estimated  $c$ s are reported.

### 5.3 $c$ -adjusted TILDE: Performance and Sensitivity to $\hat{c}$

This section aims to answer **Q1**: Does  $c$ -adjusted TILDE, the proposed relational probabilistic classifier modification method, improve over classic TILDE when faced with PU data and how sensitive is it to the correctness of  $\hat{c}$ ? To this end, TILDE was adjusted with different estimates for the label frequency that deviate from the true label frequencies with fixed values  $\Delta$ . The adjusted versions are compared to the naive method which considers unlabeled examples as negative, i.e.,  $\hat{c} = 1$ . The results are presented in Fig. 1.

As expected, taking the label frequency into account improves the classifier. A striking observation is that overestimates of the label frequency  $c$  can severely



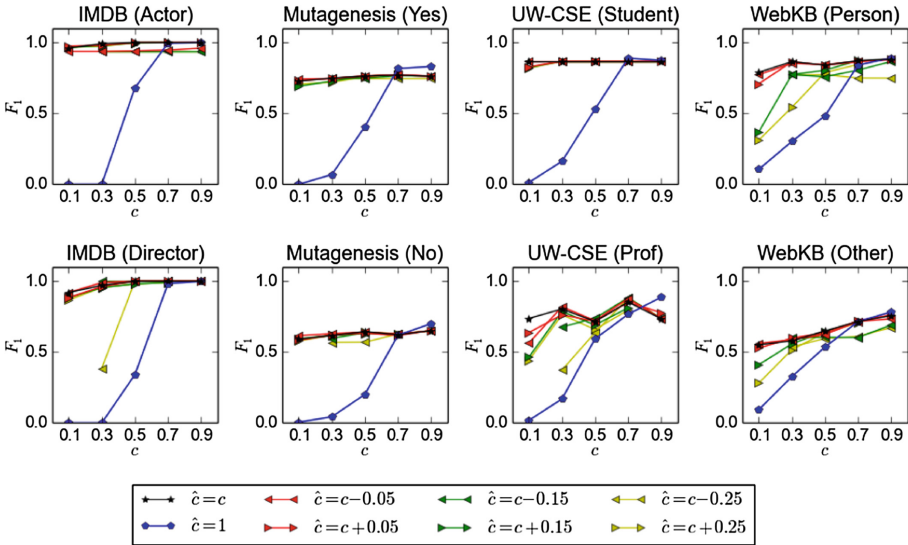
**Fig. 1. TILDE sensitivity to  $c$ .** Taking the label frequency into account clearly improves the classifier. The  $F_1$  does decrease as fewer labeled examples are provided. It is striking that underestimates and overestimates of the label frequency have very different effects on the performance.  $c$ -adjusted TILDE is very sensitive to overestimates, but not to underestimates. In fact, in some cases it even benefits from underestimates!

degrade performance, while underestimates may even improve performance. This is because of the modification method: only the leaf probabilities are altered. Therefore, an underestimate makes leaves with at least one labeled example more likely to classify instances as positive, while leaves without any labeled examples will always classify instances as negative.

#### 5.4 $c$ -adjusted Aleph: Performance and Sensitivity to $\hat{c}$

This section aims to answer **Q2**: Does  $c$ -adjusted Aleph, the proposed relational score function modification method, improve over classic Aleph when faced with PU data and how sensitive is it to the correctness of  $\hat{c}$ ? To this end, Aleph was adjusted with different estimates for the label frequency that deviate from the true label frequencies with fixed values  $\Delta$ . The adjusted versions are compared to the naive method which considers unlabeled examples as negative, i.e.,  $\hat{c} = 1$ . The results are presented in Fig. 2.

Taking the label frequency into account drastically improves the classifier: the  $F_1$  score barely drops when the label frequency decreases. In most cases, a reasonable approximation of the label frequency yields an equivalent performance to using the true label frequency. Two exceptions are (1) when there are few positive examples in the fully labeled dataset, and (2) when the target



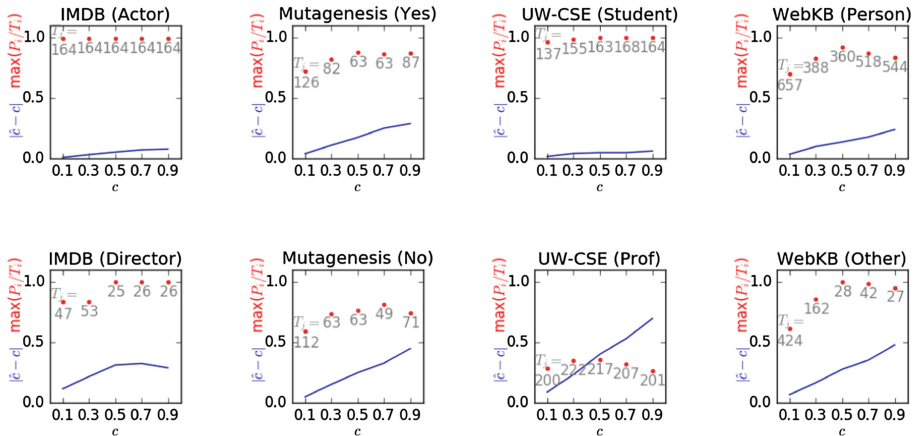
**Fig. 2. Aleph sensitivity to  $c$ .** Considering the label frequency clearly substantially improves the classifier: as the number of labeled examples decreases, the  $F_1$  score barely drops. Aleph is not very sensitive to the label frequency  $c$ , except when there are few positive examples to start with (IMDB-director and UW-CSE-Prof) or when the target concept is complex (WebKB). Even in these cases, a bad estimate for the label frequency is better than taking the unlabeled examples as negative ( $\hat{c} = 1$ ).

concept is very complex. But even in these cases, the performance does not suffer that much, especially when compared to simply assuming that all unlabeled examples belong to the negative class.

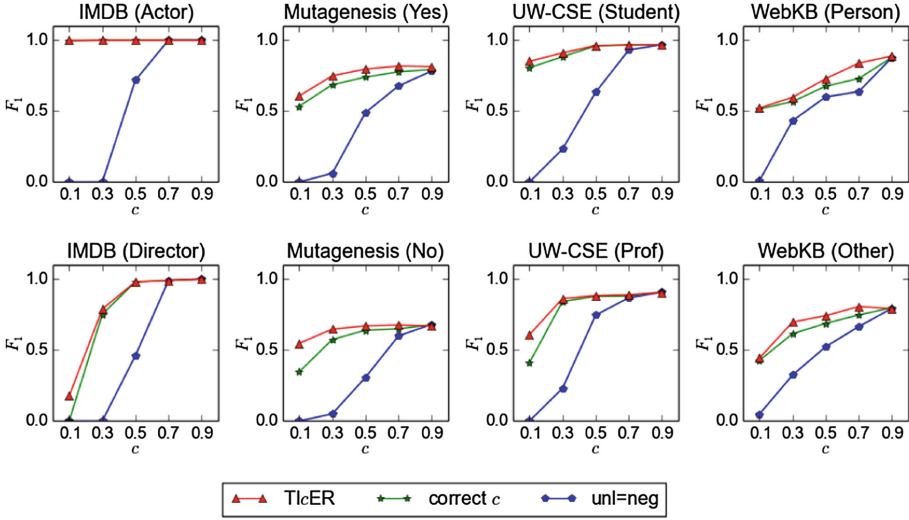
### 5.5 TICeR Evaluation

This section aims to answer **Q3**: How well does TICeR estimate the label frequency? In which cases does it perform better or worse? To this end, TICeR was used to estimate the label frequency  $c$  and compared to the true label frequency in all the training folds of all the datasets. Based on the theory, it is expected that TICeR works well when it can find subsets in the dataset that are purely positive and contain a sufficient number of examples. To check this, the maximal proportion of true positives over all the used subsets was recorded for each setting. We could look at the size of this purest subset to check if a large subset is found. However, the purest subset could be very small and another subset that is almost as pure could be very big. Therefore, we recorded the largest subset that is at least 90% as pure as the purest subset. Figure 3 shows the averaged absolute error  $|\hat{c} - c|$ , purity  $\max(P_i/T_i)$  of the purest subset  $i$  and size  $T_j$  of the largest subset  $j$  with purity close to that of the purest subset, for different label frequencies  $c$ . Figures 4 and 5 compare the performance of TILDE and Aleph respectively when adjusted with the TICeR estimate, the true label frequency and without adjusting it.

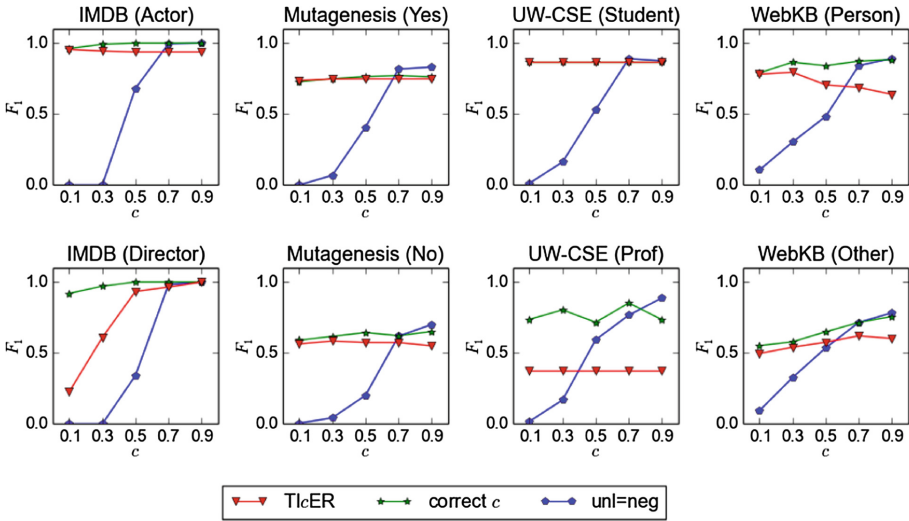
TICeR gives reasonable results most of the time. The experiments confirm our expectations: it performs worse when it fails to find subsets with a high ratio of positive examples or when the subsets contain few examples. Although the



**Fig. 3. Label Frequency Estimates.** The estimate is expected to be good if a large enough subset with a high proportion of positives was found, this is confirmed by the experiments. For example, the worst results, for UW-CSE (Prof), are explained by the low positive proportions. Subset  $i$  is the subset with the maximum purity and subset  $j$  is the largest subset that is at least 90% as pure.



**Fig. 4. TlcER-adjusted TILDE.** Adjusting TILDE with the TlcER estimate gives very similar results to adjusting it with the true label frequency, sometimes even better. This is explained by TlcER giving underestimates.



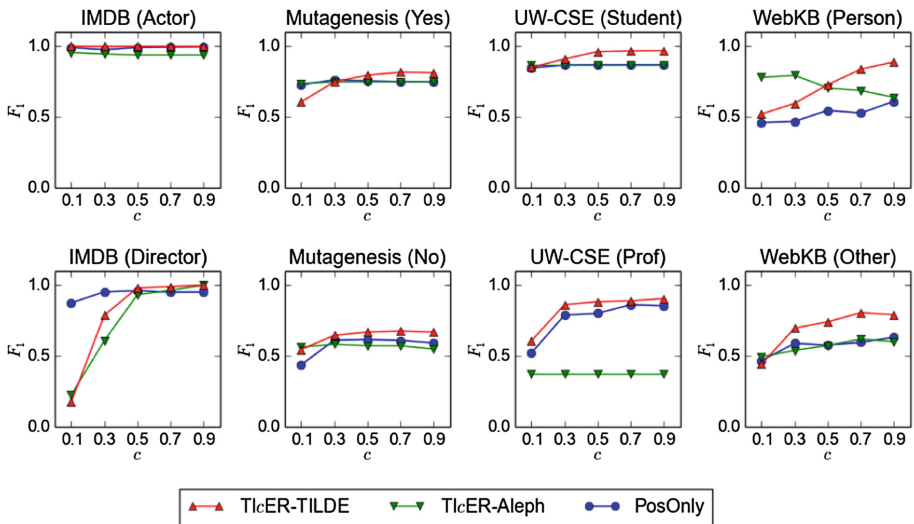
**Fig. 5. TlcER-adjusted Aleph.** Adjusting Aleph with the TlcER estimate gives for most cases similar results to adjusting it with the true label frequency. It performs worse for the datasets where Aleph is very sensitive to  $c$ . UW-CSE-Prof is doubly problematic because it is both sensitive to the label frequency and the most difficult dataset for TlcER.

estimates are not perfect, they still can improve the performance of TILDE and Aleph. Most of the time, the performance using the estimated label frequency is close to the performance of using the true label frequency. TILDE even gives better results with the estimate than with the true label frequency, this is because TILDE estimates the label frequency by looking for the maximum lower bound and hence tends to give underestimates. Aleph performs worse for the cases where it is sensitive to the label frequency. This is notably the case for UW-CSE.

## 5.6 Method Comparison

This section aims to answer **Q4**: How do label frequency adapted methods compare with Muggleton’s PosOnly method? To this end, we compare TILDE-adjusted TILDE and Aleph with PosOnly. The results are presented in Fig. 6.

The label frequency indeed makes learning from PU data easier, as it gives similar or better results than PosOnly. It is especially interesting that for the most complex dataset (WebKB) PosOnly is outperformed. The label frequency-based methods are only outperformed when there are exceptionally few labeled examples or no close-to-pure subsets in the data. Because using the label



**Fig. 6. Comparison of methods.** In most cases, the methods give similar results, which supports the claim that using the label frequency simplifies PU learning, also in the relational domain. It is interesting to see that for the most complex dataset (WebKB) Muggleton’s PosOnly is outperformed. The only situations where any of the  $c$ -adjusted methods perform significantly worse than PosOnly are those with an extremely small number of labeled examples (IMDB-Director with low label frequency) or when the estimate is extremely bad because of the lack of pure positive subsets (UW-CSE)

frequency adjust existing methods, it can benefit from any advancements and optimizations made to traditional classifiers.

## 6 Conclusions

For propositional PU classification tasks, it has long been known that knowing the label frequency greatly simplifies the problem. We transferred this idea to the relational classification tasks and make the same conclusion here. Adjusting established classifiers such as TILDE and Aleph in very simple ways perform equally well as Muggleton’s PosOnly method. For the most complex dataset, PosOnly is even outperformed. Because only small adjustments in traditional classifiers are needed, this PU classification method will improve as traditional classifiers improve.

When the label frequency is unknown, it needs to be estimated from the positive and unlabeled data. We propose a Tl<sub>c</sub>ER, a relational version of Tl<sub>c</sub>E, which employs decision trees to find pure positive subsets in the data and uses these to estimate the label frequency. This method works well when it can find highly positive subsets of the data that contain enough examples.

**Acknowledgements.** JB is supported by IWT (SB/141744). JD is partially supported by the KU Leuven Research Fund (C14/17/070, C32/17/036), FWO-Vlaanderen (SBO-150033, G066818N, EOS-30992574, T004716N), Chist-Era ReGround project, and EU VA project Nano4Sports.

## References

1. Zupanc, K., Davis, J.: Estimating rule quality for knowledge base completion with the relationship between coverage assumption. In: Proceedings of the 27th International Conference on World Wide Web (WWW 2018) (2018)
2. Claesen, M., De Smet, F., Gillard, P., Mathieu, C., De Moor, B.: Building classifiers to predict the start of glucose-lowering pharmacotherapy using Belgian health expenditure data. arXiv preprint [arXiv:1504.07389](https://arxiv.org/abs/1504.07389) (2015)
3. Muggleton, S.: Learning from positive data. In: Selected Papers from the 6th International Workshop on Inductive Logic Programming, pp. 358–376 (1996)
4. McCreath, E., Sharma, A.: ILP with noise and fixed example size: a Bayesian approach. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, pp. 1310–1315 (1997)
5. Schoenmackers, S., Davis, J., Etzioni, O., Weld, D.S.: Learning first-order Horn clauses from web text. In: Proceedings of Conference on Empirical Methods on Natural Language Processing (2010)
6. Elkan, C., Noto, K.: Learning classifiers from only positive and unlabeled data. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 213–220 (2008)
7. du Plessis, M.C., Niu, G., Sugiyama, M.: Class-prior estimation for learning from positive and unlabeled data. *Mach. Learn.*, 1–30 (2015)
8. Jain, S., White, M., Radivojac, P.: Estimating the class prior and posterior from noisy positives and unlabeled data. In: Advances in Neural Information Processing Systems (2016)

9. Ramaswamy, H.G., Scott, C., Tewari, A.: Mixture proportion estimation via kernel embedding of distributions. In: Proceedings of International Conference on Machine Learning (2016)
10. Bekker, J., Davis, J.: Estimating the class prior in positive and unlabeled data through decision tree induction. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (2018)
11. Liu, B., Lee, W.S., Yu, P.S., Li, X.: Partially supervised classification of text documents. In: Proceedings of the International Conference on Machine Learning, pp. 387–394 (2002)
12. Li, X., Liu, B.: Learning to classify texts using positive and unlabeled data. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 587–592 (2003)
13. Yu, H.: Single-class classification with mapping convergence. *Mach. Learn.* **61**(1–3), 49–69 (2005)
14. Li, X.L., Yu, P.S., Liu, B., Ng, S.K.: Positive unlabeled learning for data stream classification. In: Proceedings of the 2009 SIAM International Conference on Data Mining, pp. 259–270 (2009)
15. Nguyen, M.N., Li, X.L., Ng, S.K.: Positive unlabeled learning for time series classification. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1421–1426 (2011)
16. Lee, W.S., Liu, B.: Learning with positive and unlabeled examples using weighted logistic regression. In: Proceedings of the International Conference on Machine Learning, vol. 3, pp. 448–455 (2003)
17. Liu, Z., Shi, W., Li, D., Qin, Q.: Partially supervised classification-based on weighted unlabeled samples support vector machine. In: International Conference on Advanced Data Mining and Applications, pp. 118–129 (2005)
18. Mordelet, F., Vert, J.P.: A bagging SVM to learn from positive and unlabeled examples. *Pattern Recogn. Lett.* **37**, 201–209 (2014)
19. Claesen, M., De Smet, F., Suykens, J.A., De Moor, B.: A robust ensemble approach to learn from positive and unlabeled data using SVM base models. *Neurocomputing* **160**, 73–84 (2015)
20. Denis, F.Ç.: PAC learning from positive statistical queries. In: Richter, M.M., Smith, C.H., Wiehagen, R., Zeugmann, T. (eds.) ALT 1998. LNCS (LNAI), vol. 1501, pp. 112–126. Springer, Heidelberg (1998). [https://doi.org/10.1007/3-540-49730-7\\_9](https://doi.org/10.1007/3-540-49730-7_9)
21. Liu, B., Dai, Y., Li, X., Lee, W.S., Yu, P.S.: Building text classifiers using positive and unlabeled examples. In: Proceedings of the Third IEEE International Conference on Data Mining, pp. 179–186 (2003)
22. Zhang, D., Lee, W.S.: A simple probabilistic approach to learning from positive and unlabeled examples. In: Proceedings of the 5th Annual UK Workshop on Computational Intelligence, pp. 83–87 (2005)
23. Denis, F., Gilleron, R., Letouzey, F.: Learning from positive and unlabeled examples. *Theoret. Comput. Sci.* **348**(1), 70–83 (2005)
24. du Plessis, M.C., Sugiyama, M.: Class prior estimation from positive and unlabeled data. *IEICE Trans.* **97-D**, 1358–1362 (2014)
25. Khot, T., Natarajan, S., Shavlik, J.W.: Relational one-class classification: a non-parametric approach. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence (2014)
26. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.: Fast rule mining in ontological knowledge bases with amie+. *The VLDB J.* **24**(6), 707–730 (2015)



27. Lao, N., Subramanya, A., Pereira, F., Cohen, W.W.: Reading the web with learned syntactic-semantic inference rules. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1017–1026 (2012)
28. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems 26, pp. 926–934 (2013)
29. Gardner, M., Talukdar, P.P., Krishnamurthy, J., Mitchell, T.M.: Incorporating vector space similarity in random walk inference over knowledge bases. In: EMNLP (2014)
30. Neelakantan, A., Roth, B., McCallum, A.: Compositional vector space models for knowledge base completion. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (2015)
31. De Comit e, F., Denis, F., Gilleron, R., Letouzey, F.: Positive and unlabeled examples help learning. In: Watanabe, O., Yokomori, T. (eds.) ALT 1999. LNCS (LNAI), vol. 1720, pp. 219–230. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-46769-6\\_18](https://doi.org/10.1007/3-540-46769-6_18)
32. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. *Artif. Intell.* **101**, 285–297 (1998)
33. Srinivasan, A.: The Aleph manual (2001)



# On Applying Probabilistic Logic Programming to Breast Cancer Data

Joana Côrte-Real<sup>(✉)</sup>, Inês Dutra, and Ricardo Rocha

Faculty of Sciences and CRACS & INESC TEC, University of Porto,  
Rua do Campo Alegre, 1021/1055, 4169-007 Porto, Portugal  
{jcr,ines,ricroc}@dcc.fc.up.pt

**Abstract.** Medical data is particularly interesting as a subject for relational data mining due to the complex interactions which exist between different entities. Furthermore, the ambiguity of medical imaging causes interpretation to be complex and error-prone, and thus particularly amenable to improvement through automated decision support. Probabilistic Inductive Logic Programming (PILP) is a particularly well-suited tool for this task, since it makes it possible to combine the relational nature of this field with the ambiguity inherent in human interpretation of medical imaging. This work presents a PILP setting for breast cancer data, where several clinical and demographic variables were collected retrospectively, and new probabilistic variables and rules reflecting domain knowledge were introduced. A PILP predictive model was built automatically from this data and experiments show that it can not only match the predictions of a team of experts in the area, but also consistently reduce the error rate of malignancy prediction, when compared to other non-relational techniques.

## 1 Introduction

Probabilistic Inductive Logic Programming (PILP) is a subset of Statistical Relational Learning (SRL) that handles statistical information by using a probabilistic first-order logic language to represent data and their induced models. This technique merges technologies from the SRL and Inductive Logic Programming (ILP) [19] fields in order to automatically compose theories as understandable First Order Logic (FOL) sentences based on data annotated with probabilistic information. PILP manipulates structured representations of data so as to capture the logic relations that lie beyond the low-level features and reason about them by learning the (logical) structure of the data inductively.

The unique ability to combine the expressiveness of FOL rules with a degree of uncertainty makes PILP methods particularly well-suited to be applied in medical domains. Expert knowledge regarding the problem setting can be coded as facts or rules with varying frequencies or degrees of belief [15], and subsequently be used during the knowledge extraction stage to generate the final model. In addition, this final model also consists of a FOL theory which explains the behaviour of the system, and is easily interpretable by human experts (even though it may also be used to perform prediction over new examples).

Breast cancer is one of the most common forms of cancer and mammograms are the most commonly used technique to detect patients at risk. Image-guided core needle biopsy of the breast is then performed to decide on surgery. Biopsy is a necessary, but also aggressive, high-stakes procedure. The assessment of malignancy risk following breast core biopsy is imperfect and biopsies can be *non-definitive* in 5–15% of cases [2]. In particular, the dataset used in this work consists of demographic-related variables and information about the biopsy procedure and BI-RADS (Breast Imaging Reporting and Data System) [12] annotations, as well as domain knowledge annotated both prospectively and retrospectively by experts of three different areas: mammography, biopsy surgery and biopsy pathology. Using an automated decision support system is conducive to rigorous and accurate risk estimation of rare events and has the potential to enhance clinician decision-making and provide the opportunity for shared decision making with patients in order to personalize and strategically target health care interventions.

This work proposes a PILP decision support system targeted to this breast cancer setting. Contrary to other decision support systems, well-known in the literature (for example, Bayesian-based or SVM-based), the model proposed in this work combines probabilistic data with first order logic in order to produce both probabilistic outputs and human interpretable rules. The proposed setting includes experts' domain knowledge as (i) probabilistic rules in the background and (ii) probabilistic target values for examples. Experiments show that incorporating this domain knowledge in the model results in automated predictions which are statistically similar to those of a multidisciplinary team of human experts. Furthermore, the rules produced by the decision support system are human interpretable and relevant to the domain, which can help clinicians assess new cases.

## 2 Probabilistic Inductive Logic Programming

Introducing probabilistic information in a FOL setting allows for modelling facts or rules which are believed to be true to some degree or with a given frequency (as opposed to crisp true or false statements), which results in a closer representation of reality. Probabilities in a logic setting can also be used in cases where all the data were not gathered, since rules containing some information (if available from other sources) can be taken into account when building the final theory model. Additionally, in cases where there are privacy concerns, a similar approach can be used to avoid using the patient instances explicitly, while still considering some of the information contained in the original data.

More formally, PILP is a machine learning technique which learns predictive models from a set of probabilistic logic facts and rules. Like ILP, PILP uses a set of Probabilistic Examples (PE) and additional probabilistic logical information about the domain, the Probabilistic Background Knowledge (PBK), to find a model that explains the probabilistic examples. The PBK is a description of observed data composed of Horn clauses that can be annotated with probabilistic information known a priori. If not annotated, it is assumed that their

probabilistic value is 1. The PE represent the observations that the system is attempting to explain. They also have probabilistic values a priori. Good models will approximate the probabilistic examples values with minimum error.

In this work, probabilities are annotated according to ProbLog’s syntax, using *possible world semantics* [11]. Each fact  $p_j :: c_j$  in the PBK represents an independent binary random variable in ProbLog, meaning that it can either be true with probability  $p_j$  or false with probability  $1 - p_j$ . This means that each probabilistic fact introduces a probabilistic choice in the model. Each set of possible choices over all facts of the PBK represents a possible world  $\omega_i$ , where  $\omega_i^+$  is the set of facts that are true in that particular world, and  $\omega_i^- = \omega_i \setminus \omega_i^+$  is the set of facts that are false. Since these facts have a probabilistic value, a ProbLog program defining a probabilistic distribution over the possible worlds can be formalized as shown in Eq. 1.

$$P(\omega_i) = \prod_{c_j \in \omega_i^+} p_j \prod_{c_j \in \omega_i^-} (1 - p_j) \quad (1)$$

A ProbLog *query*  $q$  is said to be true in all worlds  $w^q$  where  $w^q \models q$ , and false in all other worlds. As such, the *success probability* of a query is given by the sum of the probabilities of all worlds where it is found to be true, as denoted in Eq. 2.

$$P(q) = \sum_{\omega_i \models q} P(\omega_i) \quad (2)$$

PILP systems learn models in the form of probabilistic logic programs.

The theories used to explain examples in PILP are built from the literals that are present in the program’s PBK. The rule (AND) search space is composed by all *Rules* whose body contains one or more of those literals. Rules can be combined using logical conjunction to form longer more *specific* rules. Let *Literals* be the set of distinct literals in the PBK. The AND search space is then the power set of *Literals*,  $\mathcal{P}(\text{Literals})$ .

The theory (OR) search space can be defined in a similar way. Theories are formed by combining a set of distinct rules using logical disjunction. In the same way that literals are the building blocks of rules, rules are the building blocks of theories. Adding a rule to a theory makes it more general. The OR search space is then the set of all theories *Theories* such that  $\text{Theories} = \mathcal{P}(\text{Rules})$ .

Fully exploring the PILP search space is equivalent to evaluating all theories in order to determine the best theory according to a given metric. This can be done in two steps: (i) exploring the AND search space, and (ii) exploring the OR search space. Algorithm 1 presents this procedure.

Algorithm 1 explores the AND search space in a direction of increasing specificity. It starts out by generating rules containing only one literal, using the mode declarations (line 2), and then uses these rules to generate combinations, which are possible according to the language bias, for the next iteration (lines 5–8), and removing the redundant rules. The combination process is repeated until it yields no new rules, or until the number of literals in the rules is greater than

**Algorithm 1.** *PILP\_search\_space*(*PBK*, *PE*, *MaxRuleLen*, *MaxTheoryLen*)

---

```

1:  $R_{all} = \emptyset$ 
2:  $R_1 = \text{generate\_rules\_one\_literal}(PBK, PE)$ 
3:  $R_{new} = R_1$ 
4:  $R_{len} = 1$ 
5: while  $R_{new} \neq \emptyset$  and  $R_{Len} \leq \text{MaxRuleLen}$  do
6:    $R_{all} = R_{all} \cup R_{new}$ 
7:    $R_{new} = \{r_1 \wedge r_{new} \mid (r_1, r_{new}) \in R_1 \times R_{new}\}$ 
8:    $R_{len} = R_{len} + 1$ 
9:  $T_{all} = \emptyset$ 
10:  $T_1 = R_{all}$ 
11:  $T_{new} = T_1$ 
12:  $T_{len} = 1$ 
13: while  $T_{new} \neq \emptyset$  and  $T_{Len} \leq \text{MaxTheoryLen}$  do
14:    $T_{all} = T_{all} \cup T_{new}$ 
15:    $T_{new} = \{t_1 \vee t_{new} \mid (t_1, t_{new}) \in T_1 \times T_{new}\}$ 
16:    $T_{len} = T_{len} + 1$ 
17: return  $T_{all}$ 

```

---

a pre-defined maximum number of literals. The set of initial theories  $T_1$  is then populated with all rules in  $R_{all}$  (line 10). Similarly to the AND search space,  $T_1$  is used to generate new theories  $T_{new}$  through combination using logical disjunction (lines 13–16). This process is analogous to the exploration of the AND search space.

### 3 Methodology

Breast cancer is one of the most common forms of cancer. Mammograms are the most commonly used technique to detect patients at risk. Image-guided core needle biopsy of the breast is then performed to decide on surgery. Biopsy is a necessary, but also aggressive, high-stakes procedure. The assessment of malignancy risk following breast core biopsy is imperfect and biopsies can be *non-definitive* in 5–15% of cases [2–4, 14, 17, 18].

A non-definitive result means that the chance of malignancy remains high due to possible sampling error (i.e., the obtained biopsy is not representative of the suspicious finding), for which surgical excisional biopsy or aggressive radiologic follow-up is proposed. Non-definitive biopsies may therefore result in missed breast cancers (false negatives) and unnecessary interventions (false positives). In the US, the women over the age of 20 years have an annual breast biopsy utilization rate of 62.6 per 10,000 women, translating to over 700,000 women undergoing breast core biopsy in 2012. As a result of non-definitive biopsies, approximately 35,000–105,000 of these women will require additional biopsies or follow-up secondary to judged inadequacy of breast core biopsy.

Interpretation can be complex and error-prone, and thus particularly amenable to improvement through automated decision support, where rigorous and accurate risk estimation of rare events have the potential to enhance clinician decision-making and provide the opportunity for shared decision making with patients in order to personalize and strategically target health care interventions.

The dataset used for this experiment contains anonymised data from 130 biopsies dating from January 2006 to December 2011, collected from the School of Medicine and Public Health of the University of Wisconsin-Madison. The data was prospectively given a non-definitive diagnosis at radiologic-histologic correlation conferences. 21 cases were determined to be malignant after surgery, and the remaining 109 proved to be benign. For all of these cases, several sources of variables were systematically collected including variables related to demographic and historical patient information (age, personal history, family history, etc.), mammographic BI-RADS descriptors (like mass shape, mass margins or calcifications), pathological information after biopsy (type of disease, if it is incidental or not, number of foci, and so on), biopsy procedure information (such as needle gauge, type of procedure), and other relevant facts about the patient.

Probabilistic data was then added to (i) the Probabilistic Examples (PE) and (ii) the Probabilistic Background Knowledge (PBK). In the first instance, the confidence in malignancy for each case (before excision) is associated with the target predicate `is_malignant/1`. The chance of malignancy is an empirical confidence value assigned by a multidisciplinary group of physicians who meet to discuss and reach an agreement about each case. Thus, the target probabilities of examples represent the perceived chance of malignancy for each patient. A high probability indicates the team of physicians thinks the case is most likely malignant, and conversely a low probability indicates the case is most likely benign. This probabilistic value was then added to the probabilistic examples and a sample of the PE is presented next:

---

```
example(is_malignant(case1), 0.10).
example(is_malignant(case2), 0.15).
example(is_malignant(case3), 0.01).
```

---

Each example is a patient case and the three examples above are part of the PE used in this experiment (one per line). Each example has two arguments, the first being the target predicate `is_malignant/1` concerning a particular case (`case1`, `case2`, or `case3`) and the second the chance of malignancy of this case (10% for `case1`, 15% for `case2`, and 1% for `case3`).

Regarding the domain knowledge incorporated in the PBK, breast cancer literature values were used to complement the information on the characteristics of masses, since physicians rely on these values to perform a diagnosis. For example, it is well known among radiology experts in mammography that if a mass has a spiculated margin, the probability that the associated finding is malignant is around 90%. The same kind of information is available in

---

```

0.05::feature_shape(Case) :-
  mass(Case, Mass),
  mass_shape(Mass, oval).

0.50::feature_shape(Case) :-
  mass(Case, Mass),
  mass_shape(Mass, round).

0.50::feature_shape(Case) :-
  mass(Case, Mass),
  mass_shape(Mass, irregular).

```

---

**Fig. 1.** Probabilistic information from the literature regarding mass shape

---

```

0.02::feature_margin(Case) :-
  mass(Case, Mass),
  mass_margin(Mass, circumscribed).

0.20::feature_margin(Case) :-
  mass(Case, Mass),
  mass_margin(Mass, indistinct).

0.70::feature_margin(Case) :-
  mass(Case, Mass),
  mass_margin(Mass, microlobulated).

0.90::feature_margin(Case) :-
  mass(Case, Mass),
  mass_margin(Mass, spiculated).

```

---

**Fig. 2.** Probabilistic information from the literature regarding mass margin

the literature for mass shape or mass density (all part of the BIRADS terms). Figures 1, 2, and 3 show how these variables are encoded in the PBK, (the notation is `probability_value::relation(...)`). Figure 1 encodes the probabilistic information regarding mass shape obtained from the literature. There are three possible rules, each one applicable to a particular kind of shape (oval, round, or irregular). A rule of this type can be read as *IF this Case has a Mass AND the Mass is of type Shape THEN this feature exists with probability P*. The probability value annotated in each rule is the frequency with which a mass whose shape is of that type is malignant. Independent rules such as the ones presented in Fig. 1 are not mutually exclusive. This means that a finding may have simultaneously an oval and round mass shape, for instance. Given that possible world semantics is used to encode these rules, the probability of two rules occurring simultaneously is given by the product of their probabilities. For

---

```

0.05::density(low);
0.10::density(equal);
0.50::density(high).

feature_density(Case) :-
    mass(Case, Mass),
    mass_density(Mass, MassDensity),
    density(MassDensity).

```

---

**Fig. 3.** Probabilistic information from the literature regarding mass density

---

```

is_malignant(Case) :-
    feature_margin(Case).
is_malignant(Case) :-
    feature_shape(Case),
    feature_density(Case).

```

---

**Fig. 4.** A PILP model for the target predicate `is_malignant/1`

instance, the probability that a mass has both an oval and round shape is equal to  $0.05 \times 0.50 = 0.025$ .

Similarly, Fig. 2 also encodes independent rules, each for a characteristic of the mass margin. In this case it becomes obvious that both the microlobulated and spiculated margins have a high correlation with malignancy in the literature, given their high probability of malignancy (70% and 90% respectively).

Figure 3 differs from Fig. 1 and Fig. 2 in that it encodes three mutually exclusive possibilities for the mass density: low, equal, or high (note the new operator “;” for disjunction). The probability of malignancy from the literature is encoded in the top three lines, which can be read as *IF the density of Mass is low, the probability of malignancy is 5%; ELSE IF the density of the Mass is equal, the probability of malignancy is 10%; ELSE IF the density of the Mass is high, the probability of malignancy is 50%*. The density rule is then constructed based on the mutual exclusivity introduced by the `density/1` fact above.

PILP models produce classifiers which are composed by a set of FOL rules, learnt automatically from the data, that represent a disjunctive explanation to the target predicate being learned. Figure 4 presents an example of a PILP model for the target predicate `is_malignant/1`, which explains malignancy in terms of margin OR mass shape and density. Since the rules in this explanation are composed of probabilistic literals (`feature_margin/1`, `feature_shape/1`, and `feature_density/1`), the target predicate `is_malignant/1` will also predict a probabilistic value ranging from 0 to 1, even though this is not made explicit in the PILP model. This probability output is computed using the *possible world semantics* [16], and it takes into account the mutual dependency between all the probabilistic literals in the model.



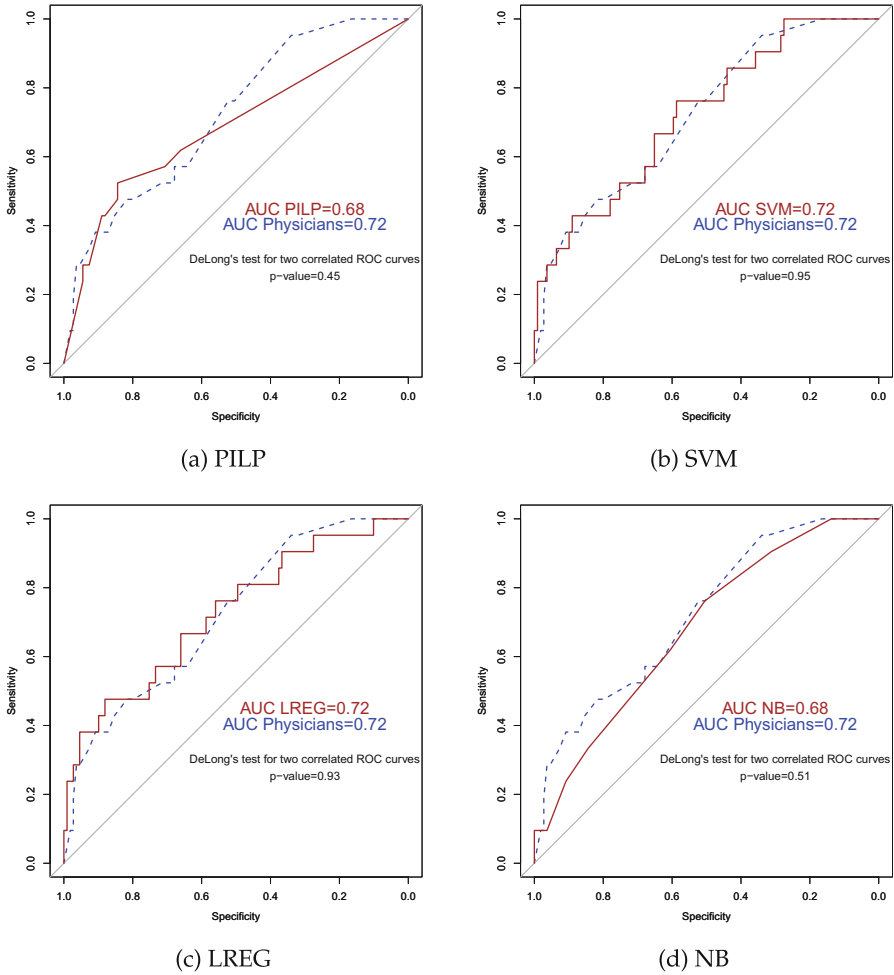
The experiment presented in this work aims at demonstrating that it is possible to use the probabilistic data to build a model that not only obtains good predictive accuracy, but also presents a human-interpretable explanation of the factors that affect the system in study. This model is learnt automatically from the data. In the medical domain it is crucial to represent data in a way that experts can understand and reason about, and as such ILP can successfully be used to produce such models. Furthermore, PILP allows for incorporating in the PBK the confidence of physicians in observations and known values from the literature.

## 4 Experiments

The PILP SkILL system [6] was used for these experiments. It runs on top of the Yap Prolog system [7] and uses TopLog [20] as the basis rules generator and the ProbLog Yap library as its probabilistic inference engine. This system was selected because it can perform exhaustive search over the theory search space. Since this is a small dataset, exhaustive search is possible. However, if the dataset were larger there might be scalability issues in using exhaustive search, and so either SkILL with pruning strategies [5] or another PILP system whose search engine is greedy could be used instead (such as ProbFOIL+ [10] or SLIPCOVER [1]). In this experiment, 130 train and tune sets were used to perform leave-one-out cross validation on the dataset, and the predicted values for the test examples were recorded.

In addition to the PILP model described earlier, three other methods were used to compare against PILP in terms of predictive accuracy, using default parameters: a Support Vector Machine (SVM), a Linear Regression (LREG), and a Naive Bayes classifier (NB). The *scikit-learn* python library [21] was used to perform the preprocessing of these experiments for the three non-relational methods. Since these data contain several categorical features, it was necessary to transform them into numerical features to be able to apply these methods. As such, each possible label was first encoded as an integer. Once this was done, each feature was transformed in several auxiliary features, each one of them binary and regarding only one of the labels. This methodology was used to prevent the integer values corresponding to the labels of a feature from being interpreted as being ordered, which would not represent the independence between the labels accurately. Once these operations were performed over all categorical features, a scaler (standardization) was applied so as to reduce all features to mean 0 and unit variance. The predictions for each method were then obtained.

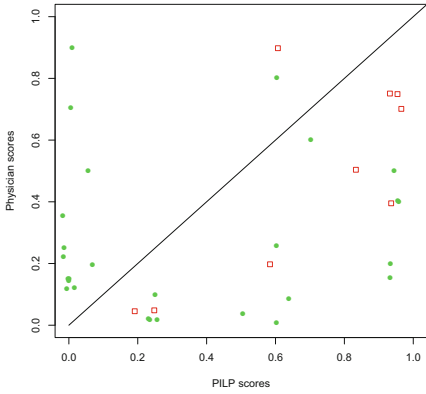
Figure 5 presents the ROC curves for the malignant class and four methods tested: PILP, SVM, LREG and NB. Each sub-figure shows the ROC of the physicians' predictions (blue dashed line) and the ROC of a method (brown solid line), both against the ground truth (confirmed malignancy or benignity of a tumour after excision). Each figure also presents the respective AUCs and the p-value found using DeLong's test for comparing both curves plotted.



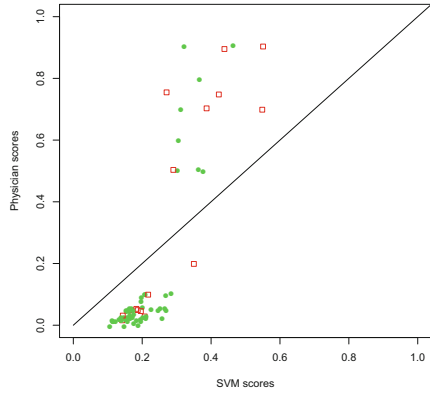
**Fig. 5.** ROC curves, AUCs and p-values for PILP, SVM, LREG and NB methods (Color figure online)

The ROC curves presented in Fig. 5 were compared using DeLong’s test for two correlated ROC curves and the difference between them was found to be statistically not significant, thus implying that all methods are statistically indistinguishable from a physician when predicting the degree of malignancy of a patient in this dataset. This experiment established that both PILP and other non-relational methods can successfully mimic the mental model of physicians in what concerns the probabilities of each case in this dataset.

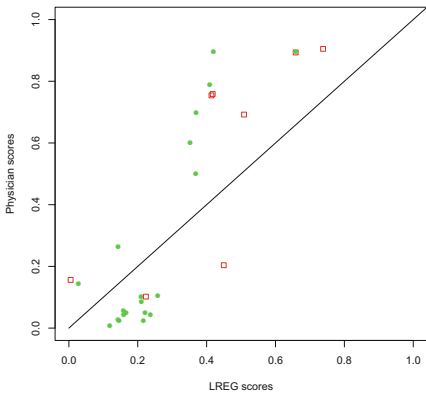
Next, the absolute error of the predictions was analysed. The absolute error is calculated by finding the absolute value of the difference between the prediction and the physicians’ score, for a given case. It is relevant to consider the absolute



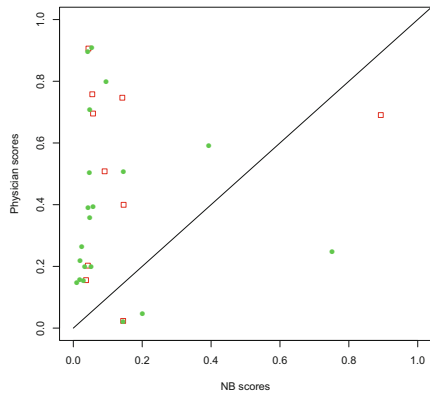
(a) PILP



(b) SVM



(c) LREG



(d) NB

**Fig. 6.** Plot of benign and malignant cases for the PILP, SVM, LREG and NB methods, for errors greater than 0.1, using a negligible amount of jittering (Color figure online)

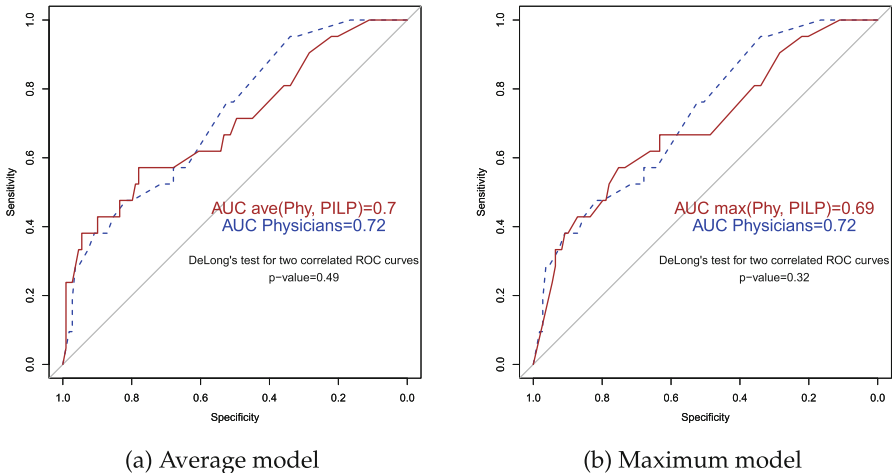
error of predictions because these are the points where the classifiers’ predictions disagree with the physicians’ mental model, and more information about the performance of the classifier can be obtained from them. Figure 6 shows a plot of the classifiers prediction values (x-axis) against the physicians’ prediction values (y-axis), for points where the absolute error was greater than 10%. Points in green (round markers) are cases where the tumour was found to be benign after excision, and conversely points in red (square markers) are cases where the tumour was found to be malignant.

Ideally, malignant prediction by both physician and the classifier should agree and appear on the top right of the plot. Conversely, benign predictions would appear on the bottom left. Points that are plotted below the diagonal line have

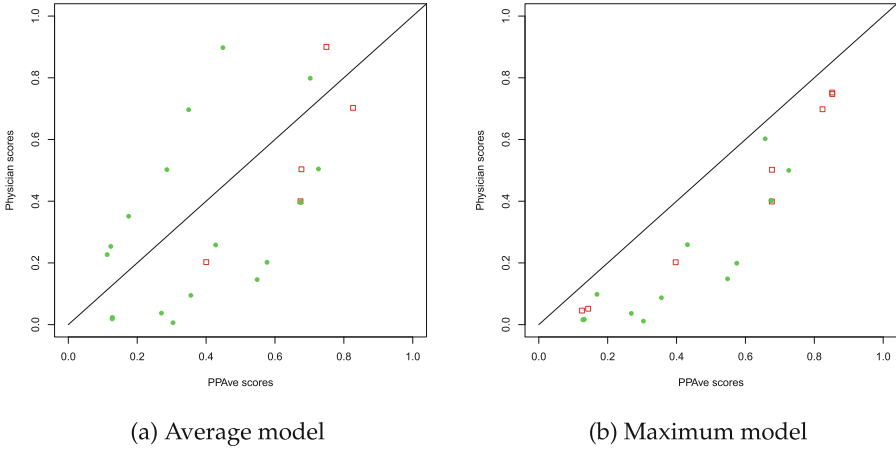
higher classifier scores than physician scores, and conversely points which are plotted above the diagonal line have higher physician scores than classifier scores.

From the plots in Fig. 6, it is clear to see that the PILP classifier assigns higher malignancy values than physicians do to the confirmed malignancy cases (red points under the diagonal line). This is the case for 8 of the 9 malignant cases, and in the single case where this does not happen, PILP still predicts a reasonably high probability of malignancy (60%). Furthermore, for a malignancy threshold of 0.8, PILP still classifies five malignant cases correctly, whilst this only happens for one case using the physicians' scores. When PILP is compared to the other methods tested, it becomes clear that, in most cases, the other methods do not assign higher scores to malignant points than physicians do (few red points beneath the diagonal line), therefore not being of as much use to physicians as PILP, to aid in the diagnosis of malignant tumours. The ability to identify malignant cases is desirable in medical data since a false negative corresponds to assigning a benign label to a patient who in fact has a malignant tumour.

Since the aim of decision support systems is to aid the process of medical diagnoses, two more models were built based on the results obtained previously. These two models are human and machine models, meaning that they take into account both the physicians' and the classifiers scores. The PILP classifier was selected since it proved to be best at identifying malignant cases that the physicians had difficulty with (unlike other methods). For this reason, two models were analysed: calculating the average of physician and the PILP scores, and calculating the maximum of the physician and the PILP scores. Figure 7 presents the ROCs, AUCs and p-values using DeLong's test for both these models.



**Fig. 7.** ROC curves, AUCs and p-values for the average of physician and PILP scores and for the maximum of physicians and PILP scores



**Fig. 8.** Plot of benign and malignant cases for the average and maximum of physician and PILP models, for errors greater than 0.1, using a negligible amount of jittering (Color figure online)

The ROC curves plotted in Fig. 7 show no significant difference to the physicians predictive power, similarly to all other classifiers tested. Figure 8 performs the absolute error analysis, plotting the points where these models' predictions and physician's predictions differ by a value greater than 10%.

The scatter plots in Fig. 8 show that the maximum model can now predict higher scores for all malignant points (all red points below the diagonal). This is to be expected since the model's scores are in effect the maximum score of the PILP and the Physician's model. However, both these models predict higher values for the benign cases as well, which is particularly evident in the case of the maximum model, where there are no points above the diagonal. Whilst a high recall is a desirable feature in a medical decision support system, the ability to discriminate between malignant and benign cases is also important. The PILP model performs better in this area (Fig. 6), since there is a vertical cluster of benign points which are clearly identified by the PILP model as being benign (score of 0.1 or less), and which are no longer present in the combined models analysed here.

Next, the full dataset was used to extract non-trivial knowledge regarding the physician's mental model that is being mimicked and the final theories found are reported in Fig. 9.

From the rules shown in Fig. 9, the first one contains a probabilistic fact related to one mammography descriptor: the shape of a mass. In medical literature, irregular shapes or spiculated margins indicate higher risk of malignancy. This is captured by the system, as well as other features such as no observed increase in mass size and an ultrasound core needle biopsy type. Similarly, the other two rules present features that are evidence of higher risk of malignancy,

---

```

is_malignant(Case):-
    biopsyProcedure(Case,usCore),
    changes_Sizeinc(Case,missing),
    feature_shape(Case).
is_malignant(Case):-
    assoFinding(Case,asymmetry),
    breastDensity(Case,scatteredFDensities),
    vacuumAssisted(Case,yes).
is_malignant(Case):-
    needleGauge(Case,9),
    offset(Case,14),
    vacuumAssisted(Case,yes).

```

---

**Fig. 9.** Theory extracted for physician’s mental models.

such as asymmetry, the gauge of the needle and a possible displacement of the needle (offset) during biopsy which can contribute as a confounding factor.

## 5 Related Work

Relational learning in the form of ILP (without probabilities) has been successfully used in the field of breast cancer. Burnside et al. [8] uncovered rules that showed high breast mass density as an important adjunct predictor of malignancy in mammograms. Later, using a similar dataset, Woods et al. validated these findings [22] performing cross-validation. In another work, Davis et al. used SAYU, an ILP system that could evaluate rules according to their score in a Bayesian network, in order to classify new cases as benign or malignant. Results for a dataset of around 65,000 mammograms consisting of malignant and benign cases showed ROC areas slightly above 70% for Recall values greater than 50% [9]. Dutra et al. showed that the integration of physician’s knowledge in the ILP learning process yielded better results than building models using only raw data [13]. The model we use in this paper was presented in more detail in [6] and [5]. One of the datasets used in those works is the same used in this paper, but only for comparing system’s execution times. To the best of our knowledge, this is the first work that applies PILP to the area of breast cancer, and illustrates how a probabilistic knowledge representation can be linked with a logic representation to learn stronger and more expressive data models.

## 6 Conclusion

This work presented a study conducted over breast cancer data, where a PILP model is learnt from the data. This and other machine learning techniques were used to perform a reasonably accurate estimate of breast cancer risk after image-guided breast biopsy, thus alleviating biopsy sampling error. The PILP model

combines first order logic with probabilistic data in order to obtain interpretable models that predict probabilities for each new case. The results show that a PILP model can achieve similar results to other traditional classifiers and that its predictions on the test sets are quite close to the experts' predictions. Furthermore, the cases where the models and physicians disagree were analysed in greater detail and it was found that the PILP model consistently assigns high malignancy probabilities to malignant cases, unlike the other models tested. Moreover, the PILP model can explicitly explain why some probability is given to a particular case (using the FOL rules generated), unlike non-relational models. Future work includes studying how changing PILP parameters affects the performance of the system on this and other datasets, as well as studying whether other relevant facts and rules from medical literature can be incorporated in the model.

**Acknowledgements.** This work was partially funded by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF) as part of project NanoSTIMA (NORTE-01-0145-FEDER-000016). Joana Côte-Real was funded by the FCT grant SFRH/BD/52235/2013. The authors would like to thank Dr. Elizabeth Burnside for making the dataset used in this paper available to us.

## References


1. Bellodi, E., Riguzzi, F.: Structure learning of probabilistic logic programs by searching the clause space. *Theor. Pract. Log. Program.* **15**(02), 169–212 (2015)
2. Berg, W.A., Hruban, R.H., Kumar, D., Singh, H.R., Brem, R.F., Gatewood, O.M.: Lessons from mammographic histopathologic correlation of large-core needle breast biopsy. *Radiographics* **16**(5), 1111–1130 (1996)
3. Brancato, B., Crocetti, E., Bianchi, S., Catarzi, S., Risso, G.G., Bulgaresi, P., Pisciole, F., Scialpi, M., Ciatto, S., Houssami, N.: Accuracy of needle biopsy of breast lesions visible on ultrasound: audit of fine needle versus core needle biopsy in 3233 consecutive samplings with ascertained outcomes. *Breast* **21**(4), 449–454 (2012)
4. Burbank, F.: Stereotactic breast biopsy: comparison of 14- and 11-gauge mamotome probe performance and complication rates. *Am. Surg.* **63**(11), 988–995 (1997)
5. Côte-Real, J., Dutra, I., Rocha, R.: Estimation-based search space traversal in PILP environments. In: Cussens, J., Russo, A. (eds.) *ILP 2016. LNCS (LNAI)*, vol. 10326, pp. 1–13. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63342-8\\_1](https://doi.org/10.1007/978-3-319-63342-8_1)
6. Côte-Real, J., Mantadelis, T., Dutra, I., Rocha, R., Burnside, E.: SKILL - a stochastic inductive logic learner. In: *International Conference on Machine Learning and Applications*, Miami, Florida, USA, December 2015
7. Santos Costa, V., Rocha, R., Damas, L.: The YAP prolog system. *J. Theor. Pract. Log. Program.* **12**(1 & 2), 5–34 (2012)
8. Davis, J., Burnside, E.S., Dutra, I.C., Page, D., Santos Costa, V.: Knowledge discovery from structured mammography reports using inductive logic programming. In: *American Medical Informatics Association 2005 Annual Symposium*, pp. 86–100 (2005)

9. Davis, J., Burnside, E.S., Dutra, I.C., Page, D., Ramakrishnan, R., Santos Costa, V., Shavlik, J.W.: View learning for statistical relational learning: with an application to mammography. In: Kaelbling, L.P., Saffiotti, A. (eds.) *IJCAI 2005, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, UK, 30 July–5 August 2005, pp. 677–683. Professional Book Center (2005)
10. De Raedt, L., Dries, A., Thon, I., Van den Broeck, G., Verbeke, M.: Inducing probabilistic relational rules from probabilistic examples. In: *International Joint Conference on Artificial Intelligence*, pp. 1835–1843. AAAI Press (2015)
11. De Raedt, L., Kimmig, A.: Probabilistic (logic) programming concepts. *Mach. Learn.* **100**(1), 5–47 (2015)
12. D’Orsi, C.J., Bassett, L.W., Berg, W.A., et al.: *BI-RADS<sup>®</sup>: Mammography*, 4th edn. American College of Radiology Inc., Reston (2003)
13. Dutra, I., Nassif, H., Page, D., et al.: Integrating machine learning and physician knowledge to improve the accuracy of breast biopsy. In: *AMIA Annual Symposium Proceedings*, Washington, DC, pp. 349–355 (2011)
14. Gonçalves, A.V., Thuler, L.C., Kestelman, F.P., Carmo, P.A., Lima, C.F., Cipolotti, R.: Underestimation of malignancy of core needle biopsy for nonpalpable breast lesions. *Rev. Bras. Ginecol. Obstet.* **33**(7), 123–131 (2011)
15. Halpern, J.: An analysis of first-order logics of probability. *Artif. Intell.* **46**(3), 311–350 (1990)
16. Kimmig, A., Demoen, B., De Raedt, L., Santos Costa, V., Rocha, R.: On the implementation of the probabilistic logic programming language ProbLog. *Theor. Pract. Log. Program.* **11**(2 & 3), 235–262 (2011)
17. Liberman, L.: Percutaneous imaging-guided core breast biopsy: state of the art at the millennium. *Am. J. Roentgenol.* **174**(5), 1191–1199 (2000)
18. Liberman, L., Drotman, M., Morris, E.A., et al.: Imaging-histologic discordance at percutaneous breast biopsy. *Cancer* **89**(12), 2538–2546 (2000)
19. Muggleton, S., De Raedt, L.: Inductive logic programming: theory and methods. *J. Log. Program.* **19**(20), 629–679 (1994)
20. Muggleton, S.H., Santos, J.C.A., Tamaddoni-Nezhad, A.: TopLog: ILP using a logic program declarative bias. In: Garcia de la Banda, M., Pontelli, E. (eds.) *ICLP 2008. LNCS*, vol. 5366, pp. 687–692. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-89982-2\\_58](https://doi.org/10.1007/978-3-540-89982-2_58)
21. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
22. Woods, R., Oliphant, L., Shinki, K., Page, D., Shavlik, J., Burnside, E.: Validation of results from knowledge discovery: mass density as a predictor of breast cancer. *J. Digit. Imaging*, 418–419 (2009)





# Logical Vision: One-Shot Meta-Interpretive Learning from Real Images

Wang-Zhou Dai<sup>1</sup> , Stephen Muggleton<sup>2</sup>, Jing Wen<sup>3</sup>,  
Alireza Tamaddoni-Nezhad<sup>4</sup>, and Zhi-Hua Zhou<sup>1</sup>

<sup>1</sup> National Key Laboratory for Novel Software Technology,  
Nanjing University, Nanjing, China  
{daiwz,zhouzh}@lamda.nju.edu.cn

<sup>2</sup> Department of Computing, Imperial College London, London, UK  
s.muggleton@imperial.ac.uk

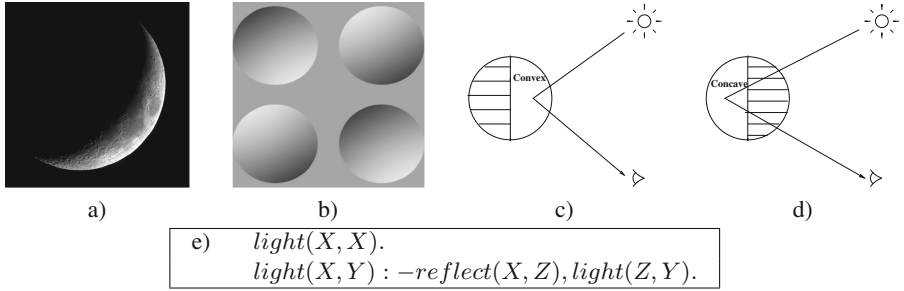
<sup>3</sup> School of Computer and Information Technology,  
Shanxi University, Taiyuan, China  
wjing@sxu.edu.cn

<sup>4</sup> Department of Computer Science, University of Surrey, Guildford, UK  
a.tamaddoni-nezhad@imperial.ac.uk

**Abstract.** Statistical machine learning is widely used in image classification. However, most techniques (1) require many images to achieve high accuracy and (2) do not provide support for reasoning below the level of classification, and so are unable to support secondary reasoning, such as the existence and position of light sources and other objects outside the image. In recent work an Inductive Logic Programming approach called Logical Vision (LV) was shown to overcome some of these limitations. LV uses Meta-Interpretive Learning combined with low-level extraction of high-contrast points sampled from the image to learn recursive logic programs describing the image. This paper extends LV by using (a) richer background knowledge enabling secondary reasoning from raw images, such as light reflection that can itself be learned and used for resolving visual ambiguities, which cannot be easily modelled using statistical approaches, (b) a wider class of background models representing classical 2D shapes such as circles and ellipses, (c) primitive-level statistical estimators to handle noise in real images. Our results indicate that the new noise-robust version of LV is able to handle secondary reasoning task in real images with few data, which is very similar to scientific discovery process of humans. Specifically, it uses a single example (i.e. one-shot LV) converges to an accuracy at least comparable to thirty-shot statistical machine learner on the prediction of hidden light sources. Moreover, we demonstrate that the learned theory can be used to identify ambiguities in the convexity/concavity of objects such as craters.

## 1 Introduction

Galileo's *Siderius Nuncius* [16] describes the first ever telescopic observations of the moon. Using sketches of shadow patterns Galileo conjectured the existence



**Fig. 1.** Interpretation of light source direction: (a) Waxing crescent moon (Credit: UC Berkeley), (b) Concave/Convex illusion caused by the viewer’s assumption about the light source location, (c) Concave and (d) Convex photon reflection models, (e) Prolog background knowledge of recursive model of photon reflection, where  $light(X, Y)$  denotes that there exists a light path between  $X$  and  $Y$ ,  $reflect(X, Z)$  is background knowledge meaning the photon travels from  $X$  to  $Z$  when  $Z$  is reflecting.

of mountains containing hollow areas (i.e. craters) on a celestial body previously thought perfectly spherical. His reasoned description, derived from a handful of observations, relies on a knowledge of (i) classical geometry, (ii) straight line movement of light and (iii) the Sun as a light source. This paper investigates use of Inductive Logic Programming (ILP) [32] to derive such hypotheses from a small set of real images. Figure 1 illustrates part of the generic background knowledge used by ILP for interpreting object convexity.

Figure 1a shows an image of the crescent moon in the night sky, in which convexity of the overall surface implies the position of the Sun as a hidden light source beyond the lower right corner of the image. Figure 1b shows an illusion caused by the viewer’s assumption about where the light source is. Assuming the light source is above makes the top right and bottom left circles appear *convex* and the other circles *concave*. Assuming the light source is below makes the top left and bottom right circles appear *convex* and the other circles *concave*. Figure 1c shows how interpretation of a *convex* feature, such as a mountain, comes from illumination of the *right* side of a convex object. Figure 1d shows that perception of a *concave* feature, such as a crater, comes from illumination of the *left* side. Figure 1e shows how Prolog background knowledge encodes a recursive definition of the reflected path of a photon.

This paper explores the phenomenon of knowledge-based perception using an extension of *Logical Vision* (LV) [9] based on *Meta-Interpretive Learning* (MIL) [7, 31]. In the previous work *Logical Vision* was shown to accurately learn a variety of polygon classes from artificial images with low sample requirements compared to statistical learners. In this paper we propose a noise-robust version of *Logical Vision* provided with basic generic background knowledge about radiation and reflection of photons to inform the generation of hypotheses in the form of logic programs based on evidence sampled from a single real image. Our experiments on light source position verified that *LV* is effective and accurate for performing secondary reasoning tasks. Empirical comparisons are made between

*LV* and Support Vector Machines (SVMs), because SVMs cannot exploit background knowledge in the form of first-order logic like *LV*, we provide them stronger supervision. Experiments showed SVMs only achieving similar accuracy with *LV* at least after more than 30 training images. Moreover, *LV* discovers a theory that could be used for explaining ambiguity.

The main contributions of this paper are extending *Logical Vision* [9] by using (1) richer background knowledge enabling secondary reasoning from raw images, such as a simple but generic recursive theory of light reflection for resolving visual ambiguities which cannot be easily modelled using pure statistical approaches, (2) a wider class of background models representing classical 2D shapes such as circles and ellipses, (3) primitive-level statistical estimators to handle noise in real images and demonstrating that the extended *LV* can learn well-performed models from only one training example (i.e. one-shot *Logical Vision*).

The paper is organised as follows. Section 2 describes related work. The theoretical framework for *Logical Vision* is provided in Sect. 3. Section 4 describes the implementation of *Logical Vision*, including the recursive background knowledge for describing radiation and reflection of light. Experiments on predicting the light source direction in images of the moon and microscopic images of illuminated micro-organisms are described in Sect. 5. In Sect. 6 we show how the approach perform secondary reasoning and interprets convexity, concavity and visual illusions from raw images. Finally, we conclude and discuss further work in Sect. 7.

## 2 Related Work

Statistical machine learning based on low-level feature extraction has been increasingly successful in image classification [35]. However, high-level vision, involving interpretation of objects and their relations in the external world, is still relatively poorly understood [5]. Since the 1990s *perception-by-induction* [18] has been the dominant model within computer vision, where human perception is viewed as inductive inference of hypotheses from sensory data. The idea originated in the work of the 19th century physiologist Hermann von Helmholtz [20]. The approach described in this paper is in line with *perception-by-induction* in using ILP for generating high-level perceptual hypotheses by combining sensory data with a strong bias in the form of explicitly encoded background knowledge. Whilst Gregory [17] was one of the earliest to demonstrate the power of the Helmholtz’s perception model for explaining human visual illusion, recent experiments [19] show Deep Neural Networks fail to reproduce human-like perception of illusion. This contrasts with results in Sect. 6, in which *Logical Vision* achieves analogous outcomes to human vision.

Shape-from-shading [21, 40] is a key computer vision technology for estimating low-level surface orientation in images. Unlike our approach for identifying concavities and convexities, shape-from-shading generally requires observation of the same object under multiple lighting conditions. By using background knowledge as a bias we reduce the number of images for accurate perception of high-level shape properties such as the identification of convex and concave image areas.

ILP has previously been used for learning concepts from images [2, 8, 11, 13, 15]. For instance, in [1, 4] object recognition is carried out using existing low-level computer vision approaches, with ILP being used for learning general relational concepts from this already symbolised starting point. By contrast, *Logical Vision* [9] uses ILP and abductive perception technique [36] to provide a bridge from very low-level primitives, such as high contrast points, to higher-level interpretation of objects such as shapes. ILP also has been used for 3D scene analysis [14, 30] with 3D point cloud data, however there was no comparison made to statistical learning and image ambiguity is not addressed.

The present paper extends the earlier work on *LV* by implementing a noise-proofing technique, applicable to real images, and extending the use of background knowledge radiation to allow the identification of objects such as light sources, not directly identifiable within the image itself. Moreover, this work shows that by considering generic knowledge about radiation, *LV* can invent generic high-level concepts applicable to many different images including concavity, convexity and light reflection, enabling 2D image analysis to learn a 3D concept with ambiguity handled.

One-shot learning of concepts from images using probabilistic program induction is discussed in [23, 24]. However, unlike the approach in this paper, the images are relatively simple and artificially generated and learning involves parameter estimation for a given program schema, rather than a search through general program space, relative to incrementally generated background knowledge.

Various statistics-based techniques making use of high-level vision have been proposed for one- or even zero-shot learning [34, 37]. They usually start from an existing model pre-trained on a large corpus of instances, and then adapt the model to data with unseen concepts. Approaches can be separated into two categories. The first exploits a mapping from images to a set of semantic attributes, then high-level models are learned based on these attributes [25, 28, 34]. The second approach uses statistics-based methods, pre-trained on a large corpus, to find localised attributes belonging to objects but not the entire image, and then exploits the semantic or spatial relationships between the attributes for scene understanding [12, 22, 26]. Unlike these approaches, we focus on one-shot learning from scratch, i.e. high-level vision based on just *very low-level primitives* such as high contrast points.

### 3 Framework

We present Meta-Interpretive Learning (MIL) first since *Logical Vision* is a special case of MIL.

#### 3.1 Meta-Interpretive Learning

Given background knowledge  $B$  and examples  $E$  the aim of a MIL system is to learn a hypothesis  $H$  such that  $B, H \models E$ , where  $B = B_p \cup M$ ,  $B_p$  is a set of Prolog definitions and  $M$  is a set of *metarules* (see Fig. 2). MIL [6, 7, 29–31] is a form of ILP based

Name	Metarule
PropObj	$P(Obj) \leftarrow$
PropHL	$P(H) \leftarrow$
PropLight	$P(Light) \leftarrow$
Conjunct3	$P(x, y, z) \leftarrow Q(x, y, z), R(x, y, z)$
Chain3	$P(u, x, y) \leftarrow Q(u, x, z), R(u, z, y)$
Chain32	$P(u, x, y) \leftarrow Q(u, x, z), R(z, y)$
PrePost3	$P(x, y, z) \leftarrow Q(x, y), R(x), S(z)$

**Fig. 2.** Metarules used in this paper. Uppercase letters  $P, Q, R, S$  denote existentially quantified variables. Lowercase letters  $u, x, y$ , and  $z$  are universally quantified.  $Obj, H, Light$  are constants representing the main object, highlight on the object and the light source in the domain,  $Prop*$  is the meta-rule for learning predicates describing the property of  $*$ .

on an adapted Prolog meta-interpreter. A standard Prolog meta-interpreter proves goals by repeatedly fetching first-order clauses whose heads unify with the goal. By contrast, a MIL learner proves a set of examples by fetching higher-order metarules (Fig. 2) whose heads unify with the goal. The resulting meta-substitutions are saved, allowing them to be used to generate a hypothesised program which proves the examples by substituting the meta-substitutions into corresponding metarules.

*MIL sample complexity.* Use of metarules and background knowledge helps minimise the number of clauses  $n$  of the minimal consistent hypothesis  $H$  and consequently the number of examples  $m$  required to achieve error below  $\epsilon$  bound. As shown in [7], the error of consistent hypotheses is bounded by  $\epsilon$  with probability at least  $1 - \delta$  once  $m \geq \frac{n \ln |M| + p \ln(3n) + \ln \frac{1}{\delta}}{\epsilon}$ , where  $p$  is the number of predicates and  $M$  is the number of metarules.

### 3.2 Logical Vision

In *Logical Vision* [9], the background knowledge  $B$ , in addition to Prolog definitions, contains a set of one or more named images  $I$ . The examples describe properties associated with  $I$ .

## 4 Implementation

In this section we describe the implementation of *Logical Vision*. The task can be formally defined as follows: The input consists of a set of training images  $D = \{(\mathbf{x}, y)\}_i^n$  with a first-order background knowledge base  $B$ , where  $\mathbf{x}_i$  stands for a raw image in training set,  $y_i \in \{1..12\}$  is a label,  $B$  is the background knowledge that is composed of a set of first-order logical clauses which is provided to  $Metagol_{AI}$  [7] with corresponding compiled background knowledge and metarules. The target is to learn a logic program for predicting light source direction on images.

---

**Algorithm 1.**  $LogVis(I, B)$ 

---

```

Input : Training images  $I$ ; Background knowledge  $B$ .
Output: Hypothesised logic program  $H$ .
1 Candidates =  $\Phi$ ;
2 for each labelled image  $i \in I$  do
3   Angles =  $\Phi$ ;
4   /* Object & highlight detection */
5   for  $t \in [1, T]$  do
6     Obj = objectDetection( $i$ );
7      $\alpha = \operatorname{argmax}_{Angle} \operatorname{contrast}(\operatorname{split}(\text{Obj}, \text{Angle}))$ ;
8     Angles = append(Angles,  $\alpha$ );
9   end
10  /* Highlight angle */
11  HAngle = mode(Angles);
12  /* Light source angle */
13  LAngle = label( $i$ );
14  /* Call MetagolAI to learn a model */
15  Model $_t$  = MetagolAI( $B$ , HAngle, LAngle);
16  Candidates = add(Model $_t$ , Candidates);
17 end
18 Return( $H = \operatorname{best}(\text{Candidates})$ );

```

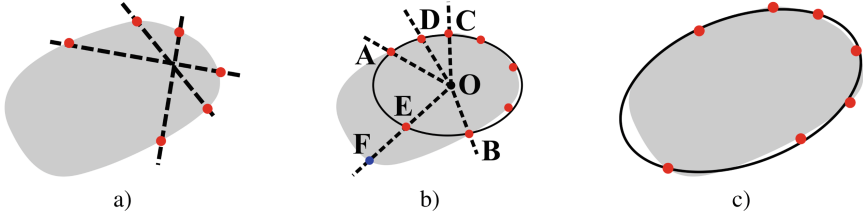
---

Within this paper we narrow the task and background knowledge. The aim is to firstly let *Logical Vision* discover objects with their highlights from the raw image, and then *Metagol<sub>AI</sub>* is applied for learning the target program with background knowledge about radiation and reflection.

Our implementation of *Logical Vision* is shown in Algorithm 1. The procedure of *LogVis* is divided into two stages. The first stage is to extract symbolic background knowledge from images, which is done by the *objectDetection* function. By including abductive theories in  $B$ , *objectDetection* can abduce ground facts about certain visual primitives from images, such as points and lines to forming polygons and ellipses as the discovered objects. The second stage of *LogVis* simply calls the MIL system *Metagol<sub>AI</sub>* to induce a hypothesis for the target concept, such as the light source direction in this paper.

#### 4.1 Meta-Interpretation in Real Images

Because microscopic and telescopic images usually contain a degree of noise, we extend the *Logical Vision* for polygon learning [9] by enhancing it with statistical models. As shown in Fig. 3, the basic process of *objectDetection* is the sampling of *edge points*—which decides if an image pixel belongs to the edge of a target object. Different to the previous version which makes judgement according to the contrast of a pixel’s local area, the new version of *edge\_point/1* is implemented with a statistical model. For example, in both *Protist* and *Moon* experiments of Sect. 5, the *edge\_point/1* is based on a pre-trained statistical image background



**Fig. 3.** Object detection: (a) Sampling lines and detect possible edge points on them; (b) Fitting of initial ellipse centred at  $O$ . Hypothesis tested by sampling new lines halfway between existing adjacent points and check if there exist edge points on the hypothesised ellipse. (c) Revising hypothesis by including the newly obtained edge points and have it tested repeatedly until a hypothesis passes test.

model which can categorise pixels into foreground or background points using Gaussian models or other image segmentation techniques [39].

Moreover, objects of interest in microscopic and telescopic images such as protists and planets are often composed of curves, using polygon to represent them (as for polygon learning [9]) would be inefficient. Consequently, we use ellipse and circle models estimated from sets of edge points (see Fig. 3).

Detected objects take the form  $elps(Centre, Parameter)$  or  $circle(Centre, Radius)$  where  $Centre = [X, Y]$  is the object's centre,  $Parameter = [A, B, Tilt]$  are the axis lengths and tilting angle and  $Radius$  is the circle radius. The computational complexity of estimating ellipse is  $O(n)$ , where  $n$  is the number of edge points.

To estimate light source direction  $LogVis$  (line 6) cuts the object in half at different angles, and returns the angle  $\alpha$  which maximises brightness contrast between the split halves, where  $\alpha \in \{1..12\}$  is a clock face angle. Since the noise brought by edge point sampling may cause object detection to fail, *Logical Vision* repeats the process  $T$  times and returns the mode of  $\{\alpha\}$  as  $HAngle$ , (line 4 to 9). In order to be processed by  $Metagol_{AI}$ , *Logical Vision* finally outputs the discovered maximum contrast angle with a logic fact  $clock\_angle(Obj, H, HAngle)$ , where  $Obj$  is the name of detected ellipse or circle,  $H$  is a constant symbol representing the brighter half of  $Obj$ .

Background knowledge for  $Metagol_{AI}$  is shown in Fig. 4. The *primitives* are used for constructing hypothesis  $H$ , *compiled BK* defines some basic facts that can be called during MIL learning process. Together with the metarules in Fig. 2,  $Metagol_{AI}$  can learn a theory (line 11 in Algorithm 1) which is abductive logic program explaining the observation of highlight on object (discovered by  $LV$ ) with light source direction and object's convexity/concavity. The metarules are 3-ary extensions to the 2-ary metarules defined in [31].

For example, when an image of convex object (such as moon, whose light source is the sun) and an example  $light\_source\_angle(moon, sun, ang)$  is provided,  $LogVis$  work as follows: (1)  $objDetection$  is first called to extract the main object (moon) as a circle; (2) By splitting the circle into two halves and

Primitives	Compiled BK
prim(light_source_angle/3). % supervision prim(highlight/2). % highlight relation prim(opposite_angle/2).	highlight(Obj,H). % H is the highlight on Obj opposite_angle(3,9). opposite_angle(9,3). opposite_angle(12,6). opposite_angle(6,12).

**Fig. 4.** Background knowledge for *Metagol<sub>AI</sub>*. The first primitive *light\_source\_angle(Obj, Light, Ang)* stands for the ground truth of light source *Light* and its angle *Ang* to the main object *Obj*, where *ang* comes from data label, *Obj* is the object abduced from image. *highlight(Obj, H)* is a fact which says *H* is the bright part on *Obj*. *opposite\_angle/2* defines the background knowledge about opposite clock angles.

calculating the contrast, *LogVis* can estimate the highlight position, then output facts *highlight(moon, H)* and *clock\_angle(moon, H, ang)*; (3) The two facts will be input to *Metagol<sub>AI</sub>* as a part of background knowledge for learning an abductive hypothesis to explain the example *light\_source\_angle(moon, sun, ang)*.

When a dataset has more than one example, *LogVis* runs the entire one-shot learning process for a random example, and returns the most accurate hypothesis on the rest of training set (line 14).

## 5 Experiments

This section describes experiments comparing one-shot *LV*<sup>1</sup> with multi-shot statistics-based learning on real image datasets.

### 5.1 Materials

We collected two real image datasets for the experiments: (1) **Protists** drawn from a microscope video of a *Protist* micro-organism, and (2) **Moons** a collection of images of the moon drawn from Google images. The instances in *Protists* are coloured images, while the images in *Moons* come from various sources and some of them are grey-scale. For the purpose of classification, we generated the two datasets by rotating images through 12 clock angles<sup>2</sup>. The datasets consist of 30 images for each angle, providing a total of 360 images. Each image contains one of four labels as follows: *North* = {11, 12, 1} clocks, *East* = {2, 3, 4} clocks, *South* = {5, 6, 7} clocks, and *West* = {8, 9, 10} clocks, as shown in Fig. 5. As we can see from the figure, there is high variance in the image sizes and colours.

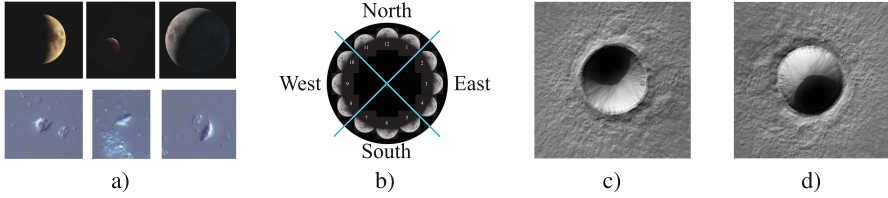
### 5.2 Methods

The aim is to learn a model to predict the correct category of light source angle from real images. For each dataset, we randomly divided the 360 images into training and test sets, with 128 and 232 examples respectively. To evaluate the

<sup>1</sup> Data and code at <https://github.com/haldai/LogicalVision2>.

<sup>2</sup> Clock face angle between 12 and each hour position in {1..12}.





**Fig. 5.** Illustrations of data: (a) Examples of the datasets, (b) Four classes for twelve light source positions, (c) Crater on Mars (Credit: NASA/JPL/University of Arizona), (d)  $180^\circ$  rotated crater.

performance, the models were trained by randomly sampling 1, 2, 4, 8, 16, 32, 64 and 128 images from the training set. The sequences of training and test instances are shared by all compared methods. The random partition of data and learning are repeated 5 times.

**Logical Vision.** In the experiments, we used the grey intensity of both image datasets for *LV*. The hyper-parameter  $T$  in Algorithm 1 was set at 11 by cross-validation. To handle image noise, we used a background model as the statistics-based estimator for predicate  $edge\_point/1$ . When  $edge\_point([X, Y])$  is called, a vector of colour distribution (which is represented by histogram of grey-scale value) of the  $10 \times 10$  region centred at  $(X, Y)$  is calculated, then the background model is applied to determine whether this vector represents an edge point. The parameter of neighbourhood region size 10 is chosen as a compromise between accuracy and efficiency after having tested it ranging from 5 to 20. The background model is trained from 5 randomly sampled images in the training set with supervision.

**Statistics-Based Classification.** The experiments with statistics-based classification were conducted in different colour spaces combined with various features. Firstly, we performed feature extraction to transform images into fixed length vectors. Next SVMs (libSVM [3]) with RBF kernel were applied to learn a multiclass-classifier model. Parameters of the SVM are chosen by cross validation on the training set. We did not choose deep neural networks because the amount of our data is limited.

Like *LV*, we used grey intensity from both image datasets for the *Moons* experiments. For the coloured *Protists* dataset, we also tried to transform the images to **HSV** and **Lab** colour spaces for statistics-based method as they have more natural metrics to describe luminance in images.

Since the image sizes in the dataset are irregular, during the object detection stage of compared statistics-based learner, we used background models and computer graphic techniques (e.g. curve fitting) to extract the main objects and unified them into same sized patches for feature extraction. The sizes of object patches were  $80 \times 80$  and  $401 \times 401$  in *Protists* and *Moons* respectively. For the

feature extraction process of the statistics-based method, we avoided descriptors which are insensitive to scale and rotation, instead choosing the luminance-sensitive features below.

- **HOG:** The Histogram of Oriented Gradient (HOG) [10] is known as its capability of describing the local gradient orientation in an image, and widely used in computer vision and image processing for the purpose of object detection.
- **LBP:** Local binary pattern (LBP) [33] is a powerful feature for texture classification by converting the local texture of an image into a binary number.

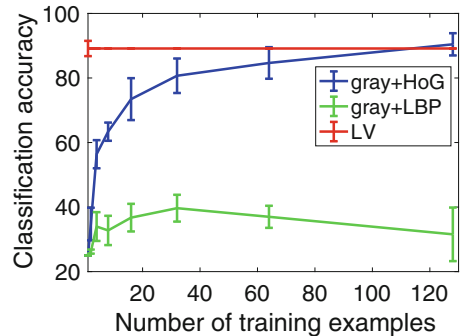
**Remark.** Despite our best efforts it proved impossible to make testing entirely fair. In the *Moons* task, *LV* and the compared statistics-based approach both used geometrical background knowledge for fitting circles (though in different forms) during object extraction. However, in the *Protists* task, the noise in images always caused poor performance in automatic object extraction for the statistics-based method. Therefore, we provided additional supervision to the statistics-based method consisting of manually labelled bounding boxes for the main objects in both training and test images during feature extraction. By comparison *LV* discovers the objects automatically.

### 5.3 Results and Discussion

Figure 6 shows the results for *Moons*. Note that performance of the statistics-based approach only surpasses one-shot *LV* after 100 training examples. In this task, background knowledge involving circle fitting exploited by *LV* and statistics-based approaches are similar, though low-level feature used by statistics-based approach are first-order information (grey-scale gradients), which is stronger than the zero-order information (grey-scale value) used by *LV*.

Results on *Protists* task are shown in Fig. 7. After 30+ training examples only one statistics-based approach outperforms one-shot *LV*. Since the statistics-based approaches have additional supervision (bounding box of main object) in the experiments, improved performance is unsurprising.

The results of *LV* in Figs. 6 and 7 form horizontal lines. When the number of training examples exceeds one, *LV* performs multiple one-shot learning and selects the best output, which we found is always in the same equivalent class in *Metagol<sub>AI</sub>*'s hypothesis space. This suggests *LV* learns the optimal logical model in its hypothesis space from a single example. In fact, the errors are caused by *LV*'s object-detection, which produces noisy inputs to the learned logical model.



**Fig. 6.** Classification accuracy on the *Moon* dataset.

The results in Figs. 6 and 7 demonstrate that *Logical Vision* can learn an accurate model using a single training example. By comparison the statistics-based approaches require 40 or even 100 more training examples to reach similar accuracy. However, the performance of *LV* heavily relies on the accuracy of the statistical estimator of *edge\_point/1*, because the mistakes of edge points detection will harm the shape fitting results and consequently the accuracy of main object extraction. Unless we train a better statistical classifier for *edge\_point/1*, the best performance of *LV* is limited as Figs. 6 and 7 shows.

The learned programs are shown in Fig. 8. They are abductive theories for explaining the observed images. Therefore they invent predicates *clock\_angle2* to represent the property of *Obj*, which can be interpreted as *convexity* (Fig. 8a) or *concavity* (Fig. 8b). For example, the abductive theory in Fig. 8a states that when *Logical Vision* observed an object *A* and its highlight *B* forming an angle *C*, a possible explanation is that the light source is *Light*, *A=Obj* is convex, the angle between light source and *Obj* is also *C*.

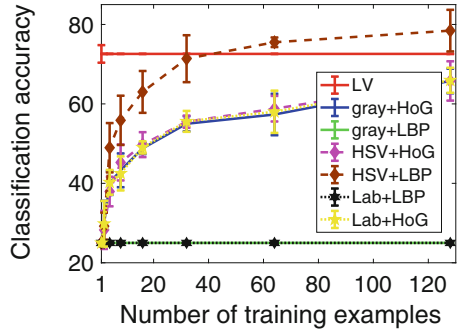


Fig. 7. Classification accuracy on the *Pro-tists* dataset.

<pre> clock_angle(A,B,C):-   clock_angle1(A,B,D),   light_source_angle(A,D,C). clock_angle1(A,B,C):-   highlight(A,B),   clock_angle2(A),clock_angle3(C). clock_angle2(Obj). clock_angle3(Light).                 </pre> <p style="text-align: center;">a)</p>	<pre> clock_angle(A,B,C):-   clock_angle1(A,B,D),   clock_angle4(A,D,C). clock_angle1(A,B,C):-   highlight(A,B),   clock_angle2(A),clock_angle3(C). clock_angle4(A,B,C):-   light_source_angle(A,B,D),   opposite_angle(D,C). clock_angle2(Obj). clock_angle3(Light).                 </pre> <p style="text-align: center;">b)</p>
--	--

Fig. 8. Program learned by *LV*: (a) Hypothesis learned when training data only contains convex objects. (b) Hypothesis learned when training data only contains concave objects. *clock\_angle/3* denotes the clock angle from *B* (highlight) to *A* (object). *highlight/2* is a built-in predicate meaning *B* is the brighter half of *A*. *light\_source\_angle/3* is an abducible predicate and the learning target. With background knowledge about lighting and compare the two programs, we can interpret the invented predicate *clock\_angle2* as *convex*, *clock\_angle3* as *light\_source.name*.

**Learning Concavity.** The *Protists* and *Moons* contain only convex objects, therefore *LV* only learned hypothesis for light radiation reacting with convex objects as shown in Fig. 8a. If instead we provide images with concave objects (such as Fig. 5c and d), *LV* learns a program such as Fig. 8b. Here the invented predicate *clock\_angle2/1* can be interpreted as *concave* because its interpretation is related to the appearance of *opposite\_angle/2*. If both convex and concave examples are provided together, *LV* can learn a more general hypothesis equivalent to Fig. 9.

**Running Time.** We implemented *LV* in SWI-Prolog [38] with multi-thread processing. Experiments were executed on a laptop with Intel i5-3210M CPU (2.50 GHz  $\times$  4), the time costs of object discovery are 9.5 s and 6.4 s per image on *Protists* and *Moons* dataset respectively; the average running time of *Metagol<sub>AI</sub>* procedure is 0.001 s on both datasets.

## 6 Using *LV* for Secondary Reasoning: Interpreting Ambiguity

A major character of science discovery is that human beings can reuse the learned knowledge as a prior to perform efficient secondary reasoning, just like Galileo uses his knowledge about geometry and light for observing the moon. The key to this ability is the common symbolic representation of prior knowledge, learned model and the formulation of human reasoning. In Sect. 5 we have shown that *LV* is able to learn logical theories as explanations to visual observations, in this section we will show that the learned models can be reused in secondary reasoning tasks.

Figure 5c and d shows two images of a crater on Mars, where Fig. 5d is a 180° rotated image of Fig. 5c. Human perception often confuses the convexity of the crater in such images<sup>3</sup>. This phenomenon, called the *crater/mountain illusion*, occurs because human vision usually interprets pictures under the default assumption that the light is from the top of the image.

*Logical Vision* can use MIL to perform abductive inference. We show below that incorporation of generic recursive background knowledge concerning light enables *LV* to generate multiple mutually inconsistent perceptual hypotheses from real images. To the authors' knowledge, such ambiguous prediction has not been demonstrated previously with other forms of machine learning.

Recall the learned programs from Fig. 8 from the previous experiments. If we rename the invented predicates we get the general theory about lighting and convexity shown in Fig. 9.

Now we can use the program as a part of background knowledge for *LV* to perform abduction, where the abducible predicates and the rest of background knowledge are shown in Fig. 10.

<sup>3</sup> <http://www.universetoday.com/118616/do-you-see-a-mountain-or-a-crater-in-this-picture/>.

```

clock_angle(O,H,A):-
    highlight(O,H),convex(O),light_source(L),
    light_source_angle(O,L,A).
clock_angle(O,H,A):-
    highlight(O,H),concave(O),light_source(L),
    light_angle(O,L,A1),opposite(A1,A).
    
```

**Fig. 9.** Interpreted background knowledge for abducing ambiguity, it is a combination of the two hypotheses in Fig. 8 learned by *LV*.

Abducibles	Interpreted BK
prim(convex/1). prim(concave/1). prim(light_source/1). prim(light_angle/3).	highlight(X,Y):- contains(X,Y),brighter(Y,X),light_source(L), light_path(L,R),reflector(R),light_path(R,O), observer(O).
<b>Compiled BK</b>	
% "obj1" is an object discovered from image by LV; % "obj2" is the brighter part of "obj1"; % "observer" is the camera contains(obj1,obj2). brighter(obj2,obj1). observer(camera). reflector(obj2). light_path(X,X). light_path(X,Y):-unobstructed(X,Z), light_path(Z,Y).	

**Fig. 10.** Background knowledge for abducing ambiguity from images. The *abducibles* are open predicates in background knowledge, i.e. they neither have definition or grounding in background knowledge. *Interpreted BK* are the logical rules containing *abducibles* in body. *Compiled BK* consists of the rest part of background knowledge.

	light_source(light). light_angle(obj1,light,south). convex(obj1).		light_source(light). light_angle(obj1,light,north). concave(obj1).
	light_source(obj2). light_angle(obj1,obj2,south). convex(obj1).		light_source(obj2). light_angle(obj1,obj2,north). concave(obj1).

**Fig. 11.** Depiction and output hypotheses abduced from Fig. 5c.

When we input Fig. 5c to *Logical Vision*, it outputs four different abductive hypotheses to explain the image, as shown in Fig. 11<sup>4</sup>. From the first two results we see that, by considering different possibilities of light source direction, *LV* can predict that the main object (which is the crater) is either convex or concave, which shows the power of learning ambiguity. The last two results are even more

<sup>4</sup> Code also at <https://github.com/haldai/LogicalVision2>.

interesting: they suggest that *obj2* (the brighter half of the crater) might be the light source as well, which indeed is possible, though seems unlikely.<sup>5</sup>

Hence, by applying a logic-based learning paradigm, *Logical Vision* is able to reuse the learned models in image processing. This paradigm, to a certain degree, mimics the human reasoning process during scientific discovery and many other tasks which requires the unification of raw data based perception and logic based reasoning.

## 7 Conclusions and Further Work

Human beings learn visual concepts from single image presentations (so-called one-shot-learning) [23]. This phenomenon is hard to explain from a standard Machine Learning perspective, given that it is unclear how to estimate any statistical parameter from a single randomly selected instance drawn from an unknown distribution. In this paper we show that learnable generic logical background knowledge can be used to generate high-accuracy logical hypotheses from single examples. This compares with similar demonstrations concerning one-shot MIL on string transformations [27] as well as previous concept learning in artificial images [9]. The experiments in Sect. 5 show that the *LV* system can accurately identify the position of a light source from a single real image, in a way analogous to scientists such as Galileo, observing the moon for the first time through a telescope or Hook observing micro-organisms for the first time through a microscope. In Sect. 6 we show that logical theories learned by *LV* from labelled images can also be used to predict concavity and convexity predicated on the assumed position of a light source.

As future work, we aim to investigate broader sets of visual phenomena which can naturally be treated using background knowledge. For instance, the effects of object obscuration; the interpretation of shadows in an image to infer the existence of out-of-frame objects; the existence of unseen objects reflected in a mirror found within the image. All these phenomena could possibly be considered in a general way from the point of view of a logical theory describing reflection and absorption of light, where each image pixel is used as evidence of photons arriving at the image plain. We will also investigate the use of universal metarules similar to those used in [6]. Future work also includes the use of probabilistic representation.

The authors believe that *LV* has long-term potential as an AI technology with the potential for unifying the disparate areas of logical based learning with visual perception.

**Acknowledgements.** This research was supported by the National Science Foundation of China (61751306). The second author acknowledges support from his Royal Academy of Engineering/Syngenta Research Chair at the Department of Computing at Imperial College London. Authors want to thank reviewers and ILP'17 attendees for helpful comments.

<sup>5</sup> The result can be reproduced and visualised by the example in Logical Vision 2 GitHub repository.

## References

1. Antanas, L., van Otterlo, M., Oramas Mogrovejo, J., Tuytelaars, T., De Raedt, L.: There are plenty of places like home: using relational representations in hierarchies for distance-based image understanding. *Neurocomputing* **123**, 75–85 (2014)
2. Cecchini, R., Del Bimbo, A.: A programming environment for imaging applications. *Pattern Recogn. Lett.* **14**(10), 817–824 (1993)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2**, 27:1–27:27 (2011)
4. Cohn, A.G., Hogg, D.C., Bennett, B., Devin, V., Galata, A., Magee, D.R., Needham, C., Santos, P.: Cognitive vision: integrating symbolic qualitative representations with computer vision. In: Christensen, H.I., Nagel, H.-H. (eds.) *Cognitive Vision Systems. LNCS*, vol. 3948, pp. 221–246. Springer, Heidelberg (2006). [https://doi.org/10.1007/11414353\\_14](https://doi.org/10.1007/11414353_14)
5. Cox, D.: Do we understand high-level vision? *Curr. Opin. Neurobiol.* **25**, 187–193 (2014)
6. Cropper, A., Muggleton, S.H.: Logical minimisation of meta-rules within meta-interpretive learning. In: Davis, J., Ramon, J. (eds.) *ILP 2014. LNCS (LNAI)*, vol. 9046, pp. 62–75. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-23708-4\\_5](https://doi.org/10.1007/978-3-319-23708-4_5)
7. Cropper, A., Muggleton, S.: Learning higher-order logic programs through abstraction and invention. In: *Proceedings of the 25th International Joint Conference Artificial Intelligence*, pp. 1418–1424 (2016)
8. Cucchiara, R., Piccardi, M., Mello, P.: Image analysis and rule-based reasoning for a traffic monitoring system. *IEEE Trans. Intell. Transp. Syst.* **1**(2), 119–130 (2000)
9. Dai, W.-Z., Muggleton, S.H., Zhou, Z.-H.: Logical vision: meta-interpretive learning for simple geometrical concepts. In: *Late Breaking Paper Proceedings of the 25th International Conference on Inductive Logic Programming*, pp. 1–16. CEUR (2015)
10. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: *Proceedings of the 13rd IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, CA, pp. 886–893. IEEE Computer Society (2005)
11. Del Bimbo, A., Vicario, E., Zingoni, D.: A spatial logic for symbolic description of image contents. *J. Vis. Lang. Comput.* **5**(3), 267–286 (1994)
12. Duan, K., Parikh, D., Crandall, D.J., Grauman, K.: Discovering localized attributes for fine-grained recognition. In: *Proceedings of the 25th IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, pp. 3474–3481. IEEE Computer Society (2012)
13. Esposito, F., Ferilli, S., Basile, T.M.A., Di Mauro, N.: Machine learning for digital document processing: from layout analysis to metadata extraction. *Mach. Learn. Doc. Anal. Recogn.* **90**, 105–138 (2008)
14. Farid, R., Sammut, C.: Plane-based object categorisation using relational learning. *Mach. Learn.* **94**(1), 3–23 (2014)
15. Ferilli, S., Basile, T.M., Esposito, F., Biba, M.: A contour-based progressive technique for shape recognition. In: *Proceedings of 2011 International Conference on Document Analysis and Recognition*, pp. 723–727 (2011)
16. Galilei, G.: *The Herald of the Stars* (1610). English translation by Edward Stafford Carlos, Rivingtons, London, 1880; edited by Peter Barker, Byzantium Press, 2004
17. Gregory, R.: *Concepts and Mechanics of Perception*. Duckworth, London (1974)

18. Gregory, R.: *Eye and Brain: The Psychology of Seeing*. Oxford University Press, Oxford (1998)
19. Heath, D., Ventura, D.: Before a computer can draw, it must first learn to see. In: *Proceedings of the 7th International Conference on Computational Creativity*, pp. 172–179 (2016)
20. von Helmholtz, H.: *Treatise on Physiological Optics*, vol. 3. Dover Publications, New York (1962). Originally published in German in 1825
21. Horn, B.: *Obtaining Shape from Shading Information*. MIT Press, Cambridge (1989)
22. Hu, R., Xu, H., Rohrbach, M., Feng, J., Saenko, K., Darrell, T.: Natural language object retrieval. In: *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV*, pp. 4555–4564. IEEE Computer Society (2016)
23. Lake, B., Salakhutdinov, R., Gross, J., Tenenbaum, J.: One shot learning of simple visual concepts. In: *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, pp. 2568–2573 (2011)
24. Lake, B., Salakhutdinov, R., Tenenbaum, J.: Human-level concept learning through probabilistic program induction. *Science* **350**, 1332–1338 (2015)
25. Lampert, C.H., Nickisch, H., Harmeling, S.: Attribute-based classification for zero-shot visual object categorization. *IEEE Trans. Pattern Anal. Mach. Intell.* **36**(3), 453–465 (2014)
26. Li, Z., Gavves, E., Mensink, T., Snoek, C.G.M.: Attributes make sense on segmented objects. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) *ECCV 2014, Part VI*. LNCS, vol. 8694, pp. 350–365. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10599-4\\_23](https://doi.org/10.1007/978-3-319-10599-4_23)
27. Lin, D., Dechter, E., Ellis, K., Tenenbaum, J., Muggleton, S.: Bias reformulation for one-shot function induction. In: *Proceedings of the 23rd European Conference on Artificial Intelligence (ECAI 2014)*, pp. 525–530. IOS Press, Amsterdam (2014)
28. Mensink, T., Verbeek, J.J., Csurka, G.: Learning structured prediction models for interactive image labeling. In: *The 24th IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO*, pp. 833–840. IEEE Computer Society (2011)
29. Muggleton, S.H., Lin, D., Chen, J., Tamaddoni-Nezhad, A.: MetaBayes: Bayesian meta-interpretive learning using higher-order stochastic refinement. In: Zaverucha, G., Santos Costa, V., Paes, A. (eds.) *ILP 2013*. LNCS (LNAI), vol. 8812, pp. 1–17. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44923-3\\_1](https://doi.org/10.1007/978-3-662-44923-3_1)
30. Muggleton, S., Lin, D., Pahlavi, N., Tamaddoni-Nezhad, A.: Meta-interpretive learning: application to grammatical inference. *Mach. Learn.* **94**, 25–49 (2014)
31. Muggleton, S., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Mach. Learn.* **100**(1), 49–73 (2015)
32. Muggleton, S., Raedt, L.D., Poole, D., Bratko, I., Flach, P., Inoue, K.: ILP turns 20: biography and future challenges. *Mach. Learn.* **86**(1), 3–23 (2011)
33. Ojala, T., Pietikainen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(7), 971–987 (2002)
34. Palatucci, M., Pomerleau, D., Hinton, G., Mitchell, T.M.: Zero-shot learning with semantic output codes. In: *Advances in Neural Information Processing Systems*, vol. 22, pp. 1410–1418. Curran Associates Inc. (2009)



35. Poppe, R.: A survey on vision-based human action recognition. *Image Vis. Comput.* **28**(6), 976–990 (2010)
36. Shanahan, M.: Perception as abduction: turning sensor data into meaningful representation. *Cogn. Sci.* **29**(1), 103–134 (2005)
37. Vinyals, O., Blundell, C., Lillicrap, T.P., Kavukcuoglu, K., Wierstra, D.: Matching networks for one shot learning. *CoRR* abs/1606.04080 (2016)
38. Wielemaker, J., Schrijvers, T., Triska, M., Lager, T.: SWI-prolog. *Theor. Pract. Logic Program.* **12**(1–2), 67–96 (2012)
39. Zhang, H., Fritts, J.E., Goldman, S.A.: Image segmentation evaluation: a survey of unsupervised methods. *Comput. Vis. Image Underst.* **110**(2), 260–280 (2008)
40. Zhang, R., Tai, P., Cryer, J., Shah, M.: Shape-from-shading: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(8), 670–706 (1999)



# Demystifying Relational Latent Representations

Sebastijan Dumančić<sup>(✉)</sup>  and Hendrik Blockeel

Department of Computer Science, KU Leuven, Leuven, Belgium  
{sebastijan.dumancic,hendrik.blockeel}@cs.kuleuven.be

**Abstract.** Latent features learned by deep learning approaches have proven to be a powerful tool for machine learning. They serve as a data abstraction that makes learning easier by capturing regularities in data explicitly. Their benefits motivated their adaptation to the relational learning context. In our previous work, we introduce an approach that learns *relational* latent features by means of clustering instances and their relations. The major drawback of latent representations is that they are often *black-box* and difficult to interpret. This work addresses these issues and shows that (1) latent features created by clustering are interpretable and capture interesting properties of data; (2) they identify local regions of instances that match well with the label, which partially explains their benefit; and (3) although the number of latent features generated by this approach is large, often many of them are highly redundant and can be removed without hurting performance much.

**Keywords:** Relational learning · Deep learning  
Unsupervised representation learning · Clustering

## 1 Introduction

Latent representations created by deep learning approaches [1] have proven to be a powerful tool in machine learning. Traditional machine learning algorithms learn a function that directly maps data to the target concept. In contrast, deep learning creates several layers of latent features between the original data and the target concept. This results in a multi-step procedure that simplifies a given task before solving it.

The progress in learning such latent representations has predominantly focused on vectorized data representations. Likewise, their utility has been recognized in the relational learning community [2] in which models are learned not only from instances but from their relationships as well [3,4]. There the problem is known as *predicate invention* [5,6]. The prevalent latent representations paradigm in that direction are *embeddings to vector spaces* [7–9]. The core idea behind the embeddings is to replace symbols with numbers and logical reasoning with algebra. More precisely, relational entities are transformed to low-dimensional vectors and relations to vectors or matrices. This way of

learning latent features corresponds to learning the low-dimensional representations of relational entities and relations. Many variations of this formalization exist, but they share the same underlying principles. Assuming facts  $\mathbf{p}(\mathbf{a}, \mathbf{b})$  and  $\mathbf{r}(\mathbf{b}, \mathbf{a})$ ,  $\mathbf{a}$  and  $\mathbf{b}$  are entities whereas  $\mathbf{p}$  and  $\mathbf{r}$  are existing relations between them. One major line of research interprets relations as *translations between facts in Euclidean space*. More precisely, given the fact  $\mathbf{p}(\mathbf{a}, \mathbf{b})$  the goal is to find vector representations of  $\mathbf{a}$ ,  $\mathbf{b}$  and  $\mathbf{p}$  (namely  $a$ ,  $b$  and  $p$ ) such that  $a + p \approx b$  and  $b + r \approx a$ . The other major line of research interprets embeddings as a *factorization of a knowledge base*. These approaches represent entities as vectors and relations as matrices; the goal is to find vector representations of entities and relations (vectors  $a$  and  $b$  for  $\mathbf{a}$  and  $\mathbf{b}$ , matrices  $P$  and  $R$  for  $\mathbf{p}$  and  $\mathbf{r}$ ) such that products  $aPb$  and  $bRa$  have *high values*. In contrast, given a false statement  $\mathbf{q}(\mathbf{a}, \mathbf{b})$  product  $aQb$  should have a *low value*.

These embeddings approaches have several drawbacks. First, the latent features created that way have no inherent meaning – they are created to satisfy the aforementioned criteria. This is thus a major obstacle for interpretability of the approach, which is important in many aspects and one of the strengths of relational learning. Second, huge amounts of data are needed in order to extract useful latent features. Knowledge bases used for training often contain millions of facts. Third, it is not clear how these approaches can handle unseen entities (i.e., an entity not present in the training set and whose embedding is therefore not known) without re-training the entire model. Fourth, compared to the full-fledged expressivity of statistical relational learning [3] these approaches have reduced expressivity.

Recently, Dumančić and Blockeel [10] introduced a complementary approach, titled CUR<sup>2</sup>LED, that takes a relational learning stance and focuses on learning relational latent representations in an unsupervised manner. Viewing relational data as a hypergraph in which instances form vertices and relationships among them form hyperedges, the authors rely on clustering to obtain latent features. The core component in this approach is a declarative and intuitive specification of the similarity measure used to cluster both instances and their relationships. This consequently makes entire approach more *transparent* with respect to the meaning of latent features, as the intuitive meaning of similarity is precisely specified.

The benefits of latent representations were clearly shown with respect to both performance and complexity. The complexity of models learned on latent features was consistently lower compared to the models learned on the original data representation. Moreover, the models learned with latent features often resulted in improved performance, by a large margin as well. These two results jointly show that latent representations capture more complex dependencies in a simple manner.

In this work we further investigate the properties of relational latent representations created by CUR<sup>2</sup>LED. We start by asking the question: *what do latent features mean?* We introduce a simple method to extract the meaning of the latent features, and show that they capture interesting properties. We ask next: *what makes latent representations effective?* The initial work showed the benefits of the latent representations, however, no explanation is offered why

that is the case. We hope to shed light behind the scene and offer (at least a partial) answer why that is the case.

In the following section we first briefly introduce neighbourhood trees – a central concept of CUR<sup>2</sup>LED. We then describe an approach used in extracting the knowledge form the latent features, and investigating the properties of such latent representation. The results are presented and discussed next, followed by the conclusion.

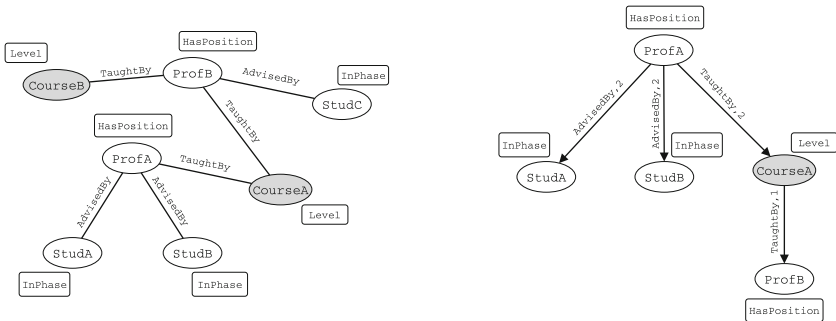
## 2 Background

### 2.1 Neighbourhood Trees

The central concept of CUR<sup>2</sup>LED is a neighbourhood tree [11]. The neighbourhood tree is a rooted directed graph describing an instance, together with instances it relates to and their properties. Viewing relational data as a hypergraph, the neighbourhood tree provides a summary of all path of a pre-defined length that originate in a particular vertex (see Fig. 1).

As instances are represented as neighbourhood trees, two instances are compared by comparing corresponding neighbourhood trees. The authors introduce a versatile and declarative similarity measure [11] that analyses neighbourhood trees over multiple aspects by introducing the following *core similarities*:

- attribute similarity of root vertices
- attribute similarity of neighbouring vertices
- connectivity between root vertices
- similarity of vertex identities in a neighbourhood
- similarity of edge types



**Fig. 1.** A snapshot of a knowledge base (left) and the corresponding neighbourhood trees of ProfA entity (right). The knowledge base describes students, professors and courses they teach. Entities (people and courses) are represented with node, their attributes with rectangles and relationships with edges. Attribute values are left out for brevity.

Continuing the example in Fig. 1, person instances can be clustered based on their own attributes, which yields clusters of professors and students. Clustering person instances based on the vertex identities in their neighbourhood yields clusters of *research groups* – a professor and his students.

In order to obtain the core similarities, a neighbourhood tree is first decomposed into three multisets:

- the multiset of vertices of type  $t$  at depth  $l$ , with the root having depth 0 ( $V_t^l$ )
- the multiset of values of attribute  $a$  observed among the nodes of type  $t$  at depth  $l$  ( $B_{t,a}^l$ )
- the multiset of edge types between depth  $l$  and  $l + 1$  ( $E_l$ ).

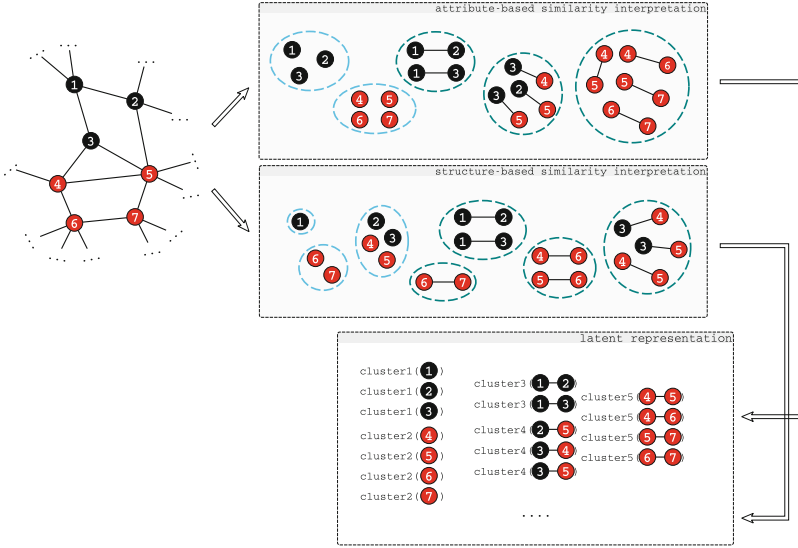
Each of the core similarities is now defined in terms of comparing relative frequencies of elements in the aforementioned multisets. For instance, the attributes similarity of root vertices is achieved by comparing relative frequencies of elements in  $B_{t,a}^0$  (for each attribute  $a$ ), while the attributes similarity of neighbouring vertices is achieved by comparing  $B_{t,a}^{>1}$  (for each vertex type  $t$  and attribute  $a$ ). Similarly, the similarity of edge types is achieved by comparing relative frequencies of elements in  $E_l$  for each level  $l$  of a neighbourhood tree. This comparison of relative frequencies of elements in a neighbourhood tree is the central notion we will exploit when discovering the meaning of latent features.

These core similarities form basic building blocks for a variety of similarity measure, all defined over neighbourhood trees. The final similarity measure is a linear weighted combination of the core similarities. Weights simply define a relative importance of core similarities in the final similarity measure. The value assignments to the weights defines a *similarity interpretation*. For more details of the approach we refer the reader to [11].

## 2.2 CUR<sup>2</sup>LED

Two ideas are central to CUR<sup>2</sup>LED. First, it learns latent features by clustering instances and their relationships. In order to cluster relationships, it makes a straightforward interpretation of relationships as sets of vertices (ordered or not). Second, it uses multiple similarity interpretations (i.e., combinations of core similarities) to obtain a variety of features. This is inspired by the notion of *distributed representation*, one of the pillars of representation learning. Distributed representation refers to a notion of a *reasonable sized representation capturing a huge number of possible configurations* [12]. A common view is that a distributed representation represents concepts with independently manipulated factors, instead of a single one with one-hot representations. Both ideas are realised by means of neighbourhood trees. Instances and relationships are represented as (collections of) neighbourhood trees, while using different similarity interpretations (which consider only certain parts of neighbourhood trees) explicitly defines a distributed representation.

Latent features are thus learned through *repeated clustering of instances and relations and alternating the similarity measure in each iteration* (see Fig. 2).



**Fig. 2.** (Figure and example taken from [10]) Starting from a relational data see as a hypergraph, CUR<sup>2</sup>LED clusters the vertices and hyperedges in the hypergraph according to different similarity interpretations. It first performs clustering based on the vertex attributes (indicated by the colour of vertices): the vertices are clustered into *red* and *black* ones, while the edges are clustered according to the colour of the vertices they connect. It then performs clustering based on the structure of the neighbourhoods (the bottom part). The vertices are clustered into clusters that have (i) only *black* neighbours ( $\{1\}$ ), (ii) only *red* neighbours ( $\{6,7\}$ ), and (iii) neighbours of both colours ( $\{2,3,4,5\}$ ). The edges are clustered into clusters of (i) edges connecting *black* vertices with only *black* neighbours and *black* vertices with *red* neighbours ( $\{1-2,1-3\}$ ), (ii) edges connecting *red* vertices with only *red* neighbours with neighbours of both colour ( $\{6-7\}$ ), and so on. The final step represents the obtained clusterings in the format of first-order logic. (Color figure online)

Each latent feature, corresponding to a cluster of instances, is associated with one latent predicate. Truth instantiations of latent predicates reflect the cluster assignments, i.e., the instantiations of a latent predicate are true for instances that belong to the cluster; therefore, latent features are defined extensionally and lack an immediate interpretable definition.

### 3 Opening the Black Box of Latent Features

The primary focus of this work is on understanding *what* do latent features created by CUR<sup>2</sup>LED mean and *why* do they prove useful. We start by describing the procedure to extract the meaning of the latent features.

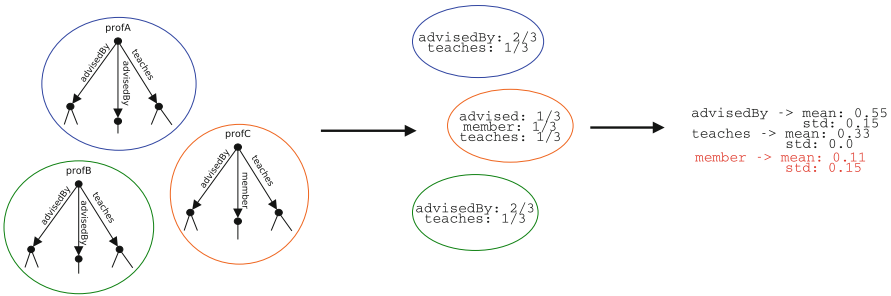
### 3.1 Extracting the Meaning of Latent Features

Although the latent features discovered by CUR<sup>2</sup>LED are defined extensionally, the intuitive specification of the similarity measure (and its core similarities) makes CUR<sup>2</sup>LED a transparent method with a clear description which elements of neighbourhood trees make two instances similar. Consequently, discovering the meaning of latent features is substantially easier than with the embedding approaches (and deep learning in general).

Each latent feature corresponds to a cluster and the meaning of the features is reflected in the *prototype* of the cluster. To approximate the *mean* or *prototypical* neighbourhood tree, we search for the elements common to all neighbourhood trees forming a cluster. These elements can be either attribute values, edge types or vertex identities. The similarity interpretations used to obtain the cluster limits which elements are considered to be a part of a definition. Moreover, neighbourhood trees [11] are compared by the relative frequencies of their elements, not the existence only. Therefore, to find a mean neighbourhood tree and the meaning of a latent feature, we search for *the elements with similar relative frequencies within each neighbourhood tree forming a cluster*.

To identify such elements, we proceed in three steps illustrated in Fig. 3.

1. **Calculate the relative frequencies of all elements within each individual neighbourhood tree, per level and vertex type.** In case of discrete attributes, that corresponds to a distribution of its values. In case of numerical attributes, we consider its mean value. In case of vertex identities and edge types, we simply look into their frequencies with respect to the depth in a neighbourhood tree. In the example in Fig. 3, the neighbourhood tree for *profA* contains two *advisedBy* relations, thus its frequency is  $\frac{2}{3}$ .



**Fig. 3. Discovering the meaning of latent features by analyzing their relations.** Properties that describe latent features are the ones that have similar relative frequency in all neighbourhood trees. Starting from a cluster of instances viewed as neighbourhood trees (left), the relative frequencies of elements are calculated for each neighbourhood tree (middle). Next, the mean and standard deviation of relative frequencies are calculated for each individual element within the cluster (right). Which elements *explain* the latent features is decided with  $\theta$ -confidence. Setting  $\theta$  to 0.3 identifies *advisedBy* and *teaches* as relevant elements (in black).

2. **Calculate the mean and standard deviation of relative frequency for each element within a cluster.** In Fig. 3, the frequencies of the `advisedBy` elements in individual neighbourhood trees are  $\frac{2}{3}$ ,  $\frac{2}{3}$  and  $\frac{1}{3}$ . Thus, its mean is 0.55 with a standard deviation of 0.15.
3. **Select relevant elements.** The final step involves a decision which elements should form a definition of a latent feature. Relevant elements are identified by a notion of  $\theta$ -confidence which captures the allowed amount of variance in order to element to be relevant.

**Definition 1 ( $\theta$ -confidence).** *An element with mean value  $\mu$  and standard deviation  $\sigma$  in a cluster, is said to be  $\theta$ -confident if  $\sigma \in [0, \theta \cdot \mu]$ .*

In Fig. 3, setting  $\theta$  to 0.3 makes `advisedBy` a 0.3-confident element, because its standard deviation of 0.15 is within the range  $[0, 0.3 \cdot 0.55] = [0, 0.165]$  specified by  $\theta$ . In contrast, `member` is not a 0.3-confident elements as its standard deviation is outside the range  $[0, 0.3 \cdot 0.11] = [0, 0.0363]$ .

The above-described procedure explains the latent features in terms of distribution of the elements in the neighbourhood of an instance, which has its pros and cons. On the downside, this type of explanation does not conform to the standard first-order logic syntax common within relational learning. Despite this reduced readability, these explanations are substantially more transparent and interpretable than the ones produced by the embeddings approaches. However, one benefit of this approach is that it increases the expressivity of a relational learner by extensionally defining properties otherwise inexpressible in the first-order logic.

### 3.2 Properties of Latent Spaces

Latent features produced by CUR<sup>2</sup>LED have proven useful in reducing the complexity of models and improving their performance. However, no explanation was offered why that is the case. In the second part of this work, we look into the properties of these latent representations and offer a partial explanation for their usefulness. To answer this question we introduce the following properties: label entropy, sparsity and redundancy.

**Entropy and sparsity.** Label entropy and sparsity serve as a proxy to a quantification of learning difficulty – i.e., how difficult is it to learn a definition of the target concept. Considering a particular predicate, label entropy reflects a *purity* of its true groundings with respect to the provided labels. Intuitively, if true groundings of predicates tend to predominantly focus on one particular label, we expect model learning to be easier.

Sparse representations, one of the cornerstones of deep learning [12], refer to a notion in which concepts are explained based on local (instead of global) properties of instance space. Even though many properties might exist for a particular problem, sparse representations describe instances using only a small subset of those properties. Intuitively, a concept spread across a small number of



local regions is expected to be easier to capture than a concept spread globally over an entire instance space.

Quantifying sparsity in relational data is a challenging task which can be approached from multiple directions – either by analysing the number of true groundings or interaction between entities, for instance. We adopt a simple definition: the number of true groundings of a predicate.

Label entropy and sparsity jointly describe a compelling property of data representation – instances space is divided in many local regions that match labels well and consequently make learning substantially easier.

**Redundancy.** A downside of CUR<sup>2</sup>LED is the high number of created features. Despite their proven usefulness, a high number of latent features enlarges the search space of a relational model and increases the difficulty of learning. As similarity interpretations are provided by the user, it is possible that almost identical clusterings are obtained with different similarity interpretations. Thus, if many of the features are redundant, removing them simplifies learning.

We measure the redundancy with the *adjusted Rand index* (ARI) [13], a standard measure for overlap between clusterings, and study its impact on the performance. To evaluate the influence of redundant features, we modify CUR<sup>2</sup>LED by adding an additional *overlap parameter*  $\alpha$ . Every time a new clustering is obtained, we check its overlap with the previously discovered clusterings using the ARI. If the calculated value is bigger than  $\alpha$ , the clustering is rejected.

## 4 Experiments and Results

We devise the experiments to answer the following questions:

- (Q1) *Are latent features created by CUR<sup>2</sup>LED interpretable and do they capture sensible information?*
- (Q2) *Do latent features that result in models of lower complexity and/or improved performance exhibit a lower label entropy compared to the original data representation?*
- (Q3) *Are latent representation that improve the performance of a model sparser than the original data representations?*
- (Q4) *To which extent are latent features redundant?*

### 4.1 Datasets and Setup

The results obtained in [10] can be divided in three categories. The first category contains the IMDB and UWCSE datasets; these datasets present easy relational learning tasks in which the original data representation is sufficient for almost perfect performance. The main benefit of latent representations for these tasks was the reduction of model complexity. The second category includes the TerroristAttack dataset, in which the main benefit of latent representation was the reduction of complexity, but not the performance. The third category involves the Hepatitis, Mutagenesis and WebKB datasets. These tasks benefited

from latent representations in both performance and reduced model complexity. That is especially true for the Hepatitis and WebKB datasets on which the performance was improved by a large margin.

We take a representative task from each of the categories. Precisely, we use IMDB, UWCSE, Hepatitis and TerroristAttack datasets in our experiments. Both IMDB and UWCSE datasets were included as they are easy to understand without the domain knowledge, and thus useful for analysing the interpretability of relational latent features. As for the parameters of latent representation, we take the best parameters on individual datasets selected by the model selection procedure in [10]. When analysing the interpretability, we set  $\theta$  to 0.3.

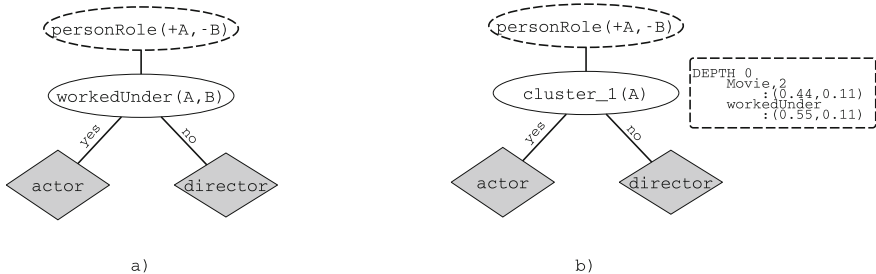
When evaluating the redundancy, we create latent representations by setting the  $\alpha$  to the following values:  $\{0.9, 0.8, 0.7, 0.6, 0.5\}$ . We then learn a relational decision tree TILDE [14] on the obtained representation and compare accuracies, the number of created features and the number of facts.

When analysing the entropy and sparsity of representations, predicates indicating labels (such as **Professor** or **Student**) and entity definitions (such as **Person** or **Course**) are not considered in the analysis.

## 4.2 Interpretability

To illustrate the interpretability of relational features, we show examples of latent features created for two easy to interpret dataset - IMDB and UWCSE. We show that the relational decision trees learned on both original and latent representations. The explanations of latent features are provided as well.

Figure 4 shows the decision trees learned on the IMDB dataset. The task is to distinguish between actors and directors – this is a simple relational learning task and both original and latent decision tree achieve the perfect performance with



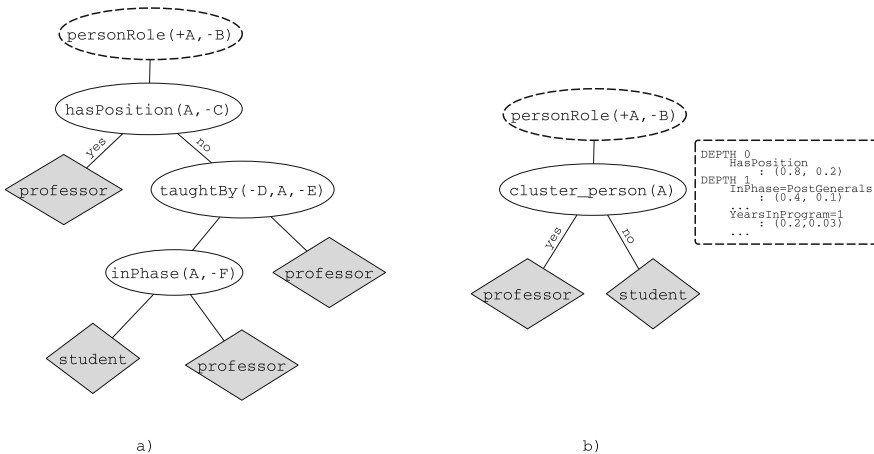
**Fig. 4.** Relational decision trees learned on the original (left) and latent (right) data representation of the IMDB dataset. The dashed ellipse indicates the target predicate and its arguments. The first argument, marked **A** and declared as input (+), denotes a person. The second argument, marked **B** and declared as output (-), states the label of the instance given by **A**. The values in the leaves of the decision trees are assignments to **B**. The dashed rectangle describes the latent feature – for each level of the *mean neighbourhood tree*,  $\theta$ -confident elements are listed with the mean and standard deviation.

only a single node. Even though latent representation does not seem beneficial in this particular case, it is interesting to see that the selected latent feature captures the same information as the decision tree learned on the original data – person instances in `cluster_1` are the ones that have a relationship with movie instances, and have worked under another person (a director).

Figure 5 shows the decision trees for the UWCSE dataset, which benefit from the latent features. Despite the simplicity of distinguishing students from professors, the decision tree learned on the latent features is more compact and has only a single node whereas the decision tree learned on the original features consists of three nodes. The latent feature here again captures similar knowledge as the original decision tree but expressed in a simpler manner – professor is someone who either has a position at the faculty, or is connected to people who are currently in a certain phase of a study program and have been in the program for a certain number of years.

What is particularly interesting about the examples above is that, even though the latent features are created in an unsupervised manner, they match the provided label very well. Moreover, they seem to almost perfectly capture the labelled information as only a few features are needed to outperform the decision tree learned on the original data representation. This observation shows that CUR<sup>2</sup>LED is indeed capturing sensible knowledge in the latent space.

Both aforementioned examples are easy to understand and interpret without an extensive domain knowledge. The other tasks that have benefited more from the latent features are substantially more difficult to understand. For instance, the latent features created from the Mutagenesis dataset reduce the complexity of the relational decision tree from 27 to only 3 nodes, while improving the accuracy for 4%. Similarly, on the Hepatitis dataset the latent features reduced the



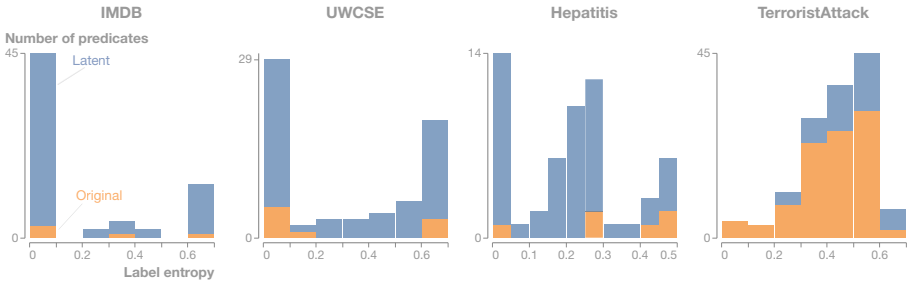
**Fig. 5.** Relational decision trees learned on the original (left) and latent (right) representations of the UWCSE dataset. The elements have the same meanings as in Fig. 4.

complexity of a decision tree from 22 nodes down to 5, improving the accuracy for 11%. Because these examples require an extensive knowledge to interpret them, we leave them out from this work.

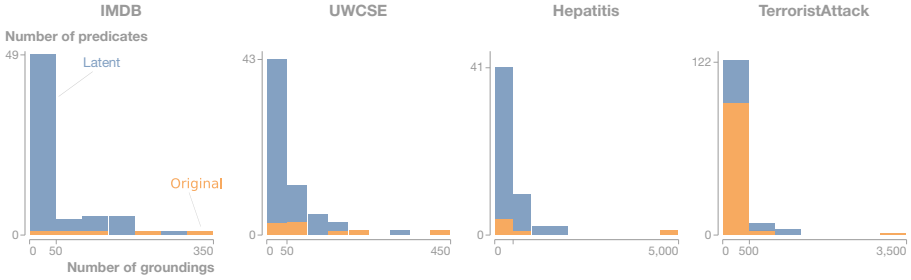
### 4.3 Properties of Latent Spaces

**Label entropy.** Figure 6 summarizes the label entropy for each dataset. In all cases where representation learning proved helpful (i.e., IMDB, UWCSE, Hepatitis), latent representations have a substantially larger number of predicates with low label entropy compared to the original data representation. The latent representation for the TerroristAttack datasets, however, shows a different behaviour in which latent features with high entropy dominate the representation. These results agree with the expectation that a high number of low entropy features makes learning easier. However, not all latent features have low label entropy. This is expected, as the labels are not considered during learning of latent features. It also does not pose a problem – these latent features are less consistent with the one particular task, but it might well be the case that those features are useful for a different task.

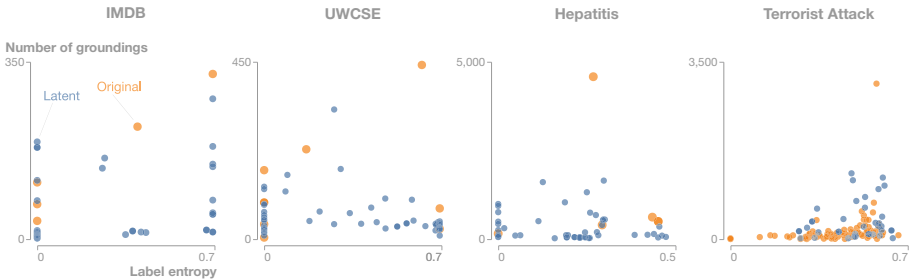
**Sparsity.** Figure 7 summarizes the sparsity results in terms of the number of true instantiations of predicates. The distribution of the number of true groundings in the latent representations (where latent features are beneficial) is heavily skewed towards a small number of groundings, in contrast with the original representation. That is especially the case with the Hepatitis dataset, which profits the most from the latent features. The exception to this behaviour is again the TerroristAttack dataset in which the original representation already is very sparse. These results indicates that latent features indeed describe smaller groups of instances and their local properties, instead of global properties of all instances.



**Fig. 6.** Latent representations for IMDB, UWCSE and Hepatitis datasets contain substantially larger number of predicates (and the corresponding facts) with low label entropy, compared to the original representation of data. On the TerroristAttack dataset, for which the latent representation has not been useful, that is not the case - both original and latent representation demonstrate similar trends in label entropy of the predicates and the corresponding facts.

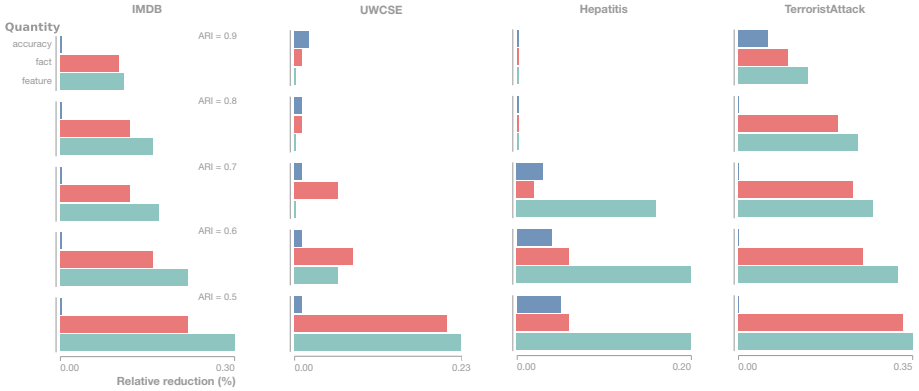


**Fig. 7.** Latent representation tends to be sparser than the original representation on the datasets where it is beneficial (IMDB, UWCSE and Hepatitis). On the TerroristAttack dataset, where the latent representation is not beneficial, both original and latent representation follow the same trend.



**Fig. 8.** Contrasting the label entropy of predicates and the number of true groundings reveals that the many latent predicates with the low label entropy have similar number of groundings as the predicates of the original data representation. This means that the trivial case, in which a large number of low-entropy predicates is obtained due to many predicates that have just a few true groundings, is not explanation for the experimental results. Instead, the latent representation, when beneficial, successfully identifies local regions in the instance space that match well with the provided labels. The exception to this is again the TerroristAttack dataset.

**Connecting label entropy and sparsity.** A potential explanation of the above discussed results might be that many latent features capture a very small number of instances (e.g., 1 or 2) which would lead to a large number of features with low label entropy. Such features would largely be useless as they make generalization very difficult. To verify that this is not the case, Fig. 8 plots the label entropy versus the number of groundings of a predicate. If latent features of low label entropy would indeed capture only a small number of instances, many points would be condensed in the bottom left corner of the plot. However, that is not the case – many latent predicates with low label entropy actually have a number of groundings comparable to the predicates in the original representation. The exception to this is again the TerroristAttacks dataset.



**Fig. 9.** The performance in terms of the accuracy is barely effected by removing overlapping clusterings, while the number of predicates and facts can be reduced up to 30%. The only noticeable reduction in performance happen on the Hepatitis dataset, but only for approximately 5%.

These results jointly point to the following conclusion: *latent features successfully identify local regions in the instance space that match well with the provided labels.* As a consequence, these local regions are easier to capture and represent.

**Redundancy.** Figure 9 summarizes the influence of  $\alpha$  on the accuracy and the number of latent features. The figure shows relative reduction in the number of features (equal to the number of predicates), the number of facts and the accuracy with respect to the latent representation obtained without rejecting the overlapping clusterings. These results show that the performance of the classifier is not affected by removing features based on the overlap of clusterings they define. The performance of TILDE remains approximately the same, whereas the number of latent features is reduced by 20 to 30 %. As the number of features is directly related to the size of the search space of relational model (and thus the complexity of learning), this is an encouraging result indicating that the size of the search space can be naively reduced without sacrificing the performance.

#### 4.4 Looking Forward

The proposed experimental framework is only the first step towards understanding how latent representations can benefit relational learning methods. The interaction between label entropy and sparsity seems to play an important role, indicative of the benefit of a latent representation. On the other hand, the method for extracting the meaning of the latent features and analysis of their redundancy are developed especially for CUR<sup>2</sup>LED and might have a limited benefit for future approaches.

Understanding when learning latent representation is (not) beneficial is an important question for further research. Majority of tasks benefits from learning latent representations, but some, like the TerroristAttack dataset, do not.

Though we cannot definitely explain why that is the case, we suspect that the reason might be that the features of instances contain the most relevant information while the structure is uninformative. In contrast, CUR<sup>2</sup>LED is developed to exploit the rich structure in relational dataset and is thus not suited for the scenario where only the features are relevant.

Another important question is how this kind of insights connects to the embeddings to vector spaces. The analysis done in this work focuses on contrasting the properties of predicates and associated data of original and latent representation obtained by CUR<sup>2</sup>LED. The embeddings to vector spaces replace the logical representation of data with points in the Euclidean space and are thus not amenable to this kind of analysis. However, similar kind of analysis for embedding spaces is currently missing in the literature. Further research towards combining relational and deep learning methods might greatly benefit from understanding up- and downsides of both directions of research, and developing new ideas that combine advantages of both.

## 5 Conclusion

In this work we closely inspect the properties of latent representations for relational data. We focus on relational latent representations created by clustering both instances and relations among them, introduced by CUR<sup>2</sup>LED [10]. The first property we analyse is the interpretability of latent features. We introduce a simple method to explain the meaning of latent features, and show that they capture interesting and sensible properties. Second, we identify two properties of these latent representation that partially explain their usefulness – namely, the label entropy and sparsity. Using these two properties, we show that obtained latent features identify local regions in instance space that match well with the labels. Consequently, this explains why predictive model learned from latent features are less complex and often perform better than the model learned from the original features. Third, we show that latent features tend to be redundant, and that 20 to 30 % of latent features can be discarded without sacrificing the performance of the classifier. This consequently reduces the search space for the relational models, and simplifies learning.

**Acknowledgements.** This research is supported by Research Fund KU Leuven (GOA/13/010) and FWO (G079416N).

## References

1. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). <http://www.deeplearningbook.org>
2. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. *Proc. IEEE* **104**(1), 11–33 (2016)
3. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press, Cambridge (2007)

4. Muggleton, S., De Raedt, L.: Inductive logic programming: theory and methods. *J. Logic Program.* **19**(20), 629–679 (1994)
5. Muggleton, S.: Predicate invention and utilization. *J. Exp. Theor. Artif. Intell.* **6**, 121–130 (1994)
6. Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Mach. Learn.* **100**(1), 49–73 (2015)
7. Bordes, A., Weston, J., Collobert, R., Bengio, R.: Learning structured embeddings of knowledge bases. In: Burgard, W., Roth, D. (eds.) *AAAI*. AAAI Press (2011)
8. Nickel, M., Tresp, V., Kriegel, H.-P.: A three-way model for collective learning on multi-relational data. In: Getoor, L., Scheffer, T. (eds.) *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, pp. 809–816. ACM, New York (2011)
9. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems 26*, pp. 2787–2795. Curran Associates Inc. (2013)
10. Dumančić, S., Blockeel, H.: Clustering-based relational unsupervised representation learning with an explicit distributed representation. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*, pp. 1631–1637 (2017)
11. Dumančić, S., Blockeel, H.: An expressive dissimilarity measure for relational clustering using neighbourhood trees. *Mach. Learn.* **106**, 1523–1545 (2017)
12. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(8), 1798–1828 (2013)
13. Morey, L.C., Agresti, A.: The measurement of classification agreement: an adjustment to the rand statistic for chance agreement. *Educ. Psychol. Measur.* **44**(1), 33–37 (1984)
14. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. *Artif. Intell.* **101**(1–2), 285–297 (1998)





# Parallel Online Learning of Event Definitions

Nikos Katzouris<sup>1</sup>(✉), Alexander Artikis<sup>1,2</sup>, and Georgios Paliouras<sup>1</sup>

<sup>1</sup> National Center for Scientific Research “Demokritos”, Athens, Greece  
{nkatz,a.artikis,paliourg}@iit.demokritos.gr

<sup>2</sup> Department of Maritime Studies, University of Piraeus, Piraeus, Greece

**Abstract.** Logic-based event recognition systems infer occurrences of events in time using a set of event definitions in the form of first-order rules. The Event Calculus is a temporal logic that has been used as a basis in event recognition applications, providing among others, direct connections to machine learning, via Inductive Logic Programming (ILP). OLED is a recently proposed ILP system that learns event definitions in the form of Event Calculus theories, in a single pass over a data stream. In this work we present a version of OLED that allows for parallel, online learning. We evaluate our approach on a benchmark activity recognition dataset and show that we can reduce training times, while achieving super-linear speed-ups on some occasions.

## 1 Introduction

Event recognition systems [9] process sequences of *simple events*, such as sensor data, and recognize *complex events*, i.e. events that satisfy some pattern. Logic-based systems for event recognition [6] typically use a knowledge base of first-order rules to represent complex event patterns and a reasoning engine for pattern matching in the incoming data stream. The Event Calculus (EC) [19] has been used as the basis for event recognition systems [4, 23], offering direct connections to machine learning, via Inductive Logic Programming (ILP) [8].

Event recognition applications deal with noisy data streams [1]. Methods that learn from such streams typically build a decision model by a single pass over the input [13]. OLED (Online Learning of Event Definitions) [18] is an ILP system that learns event definitions in the form of EC theories in a single pass over a relational data stream. OLED learns clauses in top-down manner, by gradually specializing an over-general clause using literals from a bottom clause. Its single-pass strategy is based on the Hoeffding bound [15], a statistical tool that allows to build decision models by approximating their quality on the entire input from a small subset of it. We present an extension of OLED, that allows for learning a theory in an online and *parallel* fashion. Our approach is based on a simple parallelization scheme of the core OLED functionality. In the proposed parallelization strategy, a clause is evaluated in parallel on sub-streams of the input stream and its independent scores are combined whenever a specialization

decision must be made. We present an evaluation of our approach on a benchmark activity recognition dataset and show that it can reduce training times, while it is also capable of super-linear speed-ups on some occasions.

The rest of this paper is structured as follows: In Sect. 2 we present some background on the EC. In Sect. 3 we present OLED and in Sect. 4 we present its parallel version. In Sect. 5 we present our experimental results, while in Sect. 6 we discuss related work. Finally, in Sect. 7 we discuss some directions for future work and conclude.

## 2 Background

The Event Calculus (EC) [19] is a temporal logic for reasoning about events and their effects. Its ontology consists of *time points* (integer numbers); *fluents*, i.e. properties that have different values in time; and events, i.e. occurrences in time that may alter fluents' values. The axioms of the EC incorporate the *common sense law of inertia*, according to which fluents persist over time, unless they are affected by an event. We use a simplified version of the EC that has been shown to suffice for event recognition [4]. The basic predicates and its domain-independent axioms are presented in Table 1. Axiom (1) states that a fluent  $F$  holds at time  $T$  if it has been initiated at the previous time point, while Axiom (2) states that  $F$  continues to hold unless it is terminated. Definitions for  $\text{initiatedAt}/2$  and  $\text{terminatedAt}/2$  predicates are given in an application-specific manner by a set of *domain-specific* axioms.

We illustrate our approach using the task of activity recognition, as defined in the CAVIAR project<sup>1</sup>. The CAVIAR dataset consists of videos where actors perform some activities. Manual annotation (performed by the CAVIAR team) provides ground truth for two activity types. The first type corresponds to simple events and consists of knowledge about the activities of a person at a certain video frame/time point, such as *walking*, or *standing still*. The second activity type corresponds to complex events and consists of activities that involve more

**Table 1.** The basic predicates and domain-independent axioms of EC.

Predicate	Meaning
$\text{happensAt}(E, T)$	Event $E$ occurs at time $T$
$\text{initiatedAt}(F, T)$	At time $T$ a period of time for which fluent $F$ holds is initiated
$\text{terminatedAt}(F, T)$	At time $T$ a period of time for which fluent $F$ holds is terminated
$\text{holdsAt}(F, T)$	Fluent $F$ holds at time $T$
Axioms	
$\text{holdsAt}(F, T + 1) \leftarrow \text{initiatedAt}(F, T)$	$\text{holdsAt}(F, T + 1) \leftarrow \text{holdsAt}(F, T), \text{not terminatedAt}(F, T)$

<sup>1</sup> <http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/>.

than one person, e.g. two people *meeting each other*, or *moving together*. The goal is to recognize complex events as combinations of simple events and additional contextual knowledge, such as a person’s direction and position.

Table 2(a) presents some example CAVIAR data, consisting of a narrative of simple events in terms of `happensAt/2`, expressing people’s short-term activities, and context properties in terms of `holdsAt/2`, denoting people’s coordinates and direction. Table 2(a) also shows the annotation of complex events (long-term activities) for each time-point in the narrative. Negated complex events’ annotation is obtained via the closed world assumption (although both positive and negated annotation atoms are presented in Table 2, to avoid confusion). Table 2(b) presents two domain-specific axioms in the  $\text{EC}$ .

Our goal is to learn definitions of complex events in terms of initiation and termination conditions, as in Table 2(b). In the learning setting that we assume the training data consist of Herbrand interpretations, i.e. sets of true ground atoms, as in Table 2(a). Positive examples are annotation atoms contained in such interpretations, while negative examples are false annotation atom instances generated via the closed world assumption. Given a set of training interpretations  $\mathcal{I}$ , some background theory  $B$ , which in our case consists of the domain-independent axioms of the  $\text{EC}$ , and some language bias  $M$ , the goal is to learn a theory  $H$  that fits the training data well, i.e. it accounts for as many positive examples and as few negative examples as possible. Formally, given a theory  $H$  and an interpretation  $I$ , let  $M_I^H$  denote a model of  $B \cup H \cup I$  and  $\text{annotation}(I)$  denote the annotation atoms of  $I$ . Although different semantics are possible, in this work by “model” we mean a stable model. Also, let  $\text{positives}(H, I)$

**Table 2. (a)** Example data from activity recognition. For example, at time point 1 person with  $id_1$  is *walking*, her  $(x, y)$  coordinates are  $(201, 454)$  and her direction is  $270^\circ$ . The annotation for the same time point states that persons with  $id_1$  and  $id_2$  are not moving together, in contrast to the annotation for time point 2. **(b)** An example of two domain-specific axioms in the  $\text{EC}$ . E.g. the first clause dictates that *moving together* between two persons  $X$  and  $Y$  is initiated at time  $T$  if both  $X$  and  $Y$  are walking at time  $T$ , their euclidean distance is less than 25 pixel positions and their difference in direction is less than  $45^\circ$ . The second clause dictates that *moving together* between  $X$  and  $Y$  is terminated at time  $T$  if one of them is standing still at time  $T$  (exhibits an inactive behavior) and their euclidean distance at  $T$  is greater than 30.

(a)	(b)
<b><u>Narrative for time 1:</u></b>	<b><u>Two Domain-specific axioms:</u></b>
<code>happensAt(walk(id<sub>1</sub>), 1).</code>	<code>initiatedAt(move(X, Y), T) ←</code>
<code>happensAt(walk(id<sub>2</sub>), 1).</code>	<code>  happensAt(walk(X), T),</code>
<code>holdsAt(coords(id<sub>1</sub>, 201, 454), 1).</code>	<code>  happensAt(walk(Y), T),</code>
<code>holdsAt(coords(id<sub>2</sub>, 230, 440), 1).</code>	<code>  distLessThan(X, Y, 25, T),</code>
<code>holdsAt(direction(id<sub>1</sub>, 270), 1).</code>	<code>  dirLessThan(X, Y, 45, T).</code>
<code>holdsAt(direction(id<sub>2</sub>, 270), 1).</code>	
<b><u>Annotation for time 1:</u></b>	<b><u>Annotation for time 2:</u></b>
<code>not holdsAt(move(id<sub>1</sub>, id<sub>2</sub>), 1)</code>	<code>holdsAt(move(id<sub>1</sub>, id<sub>2</sub>), 2)</code>
	<code>terminatedAt(move(X, Y), T) ←</code>
	<code>  happensAt(inactive(X), T),</code>
	<code>  distMoreThan(X, Y, 30, T).</code>

(resp.  $negatives(H, I)$ ) be the set of complex event instances  $a$  with the property  $\alpha \in M_I^H \cap annotation(I)$  (resp.  $\alpha \in M_I^H \setminus annotation(I)$ ). The goal then is to learn a theory  $H$  with the property

$$\operatorname{argmax}_{H \in \mathcal{L}(M)} \left( \sum_{I \in \mathcal{I}} |positives(H, I)| - |negatives(H, I)| \right)$$

where  $\mathcal{L}(M)$  denotes the hypothesis language defined by the language bias  $M$ . The language bias that we assume is mode declarations [20].

### 3 The OLED System

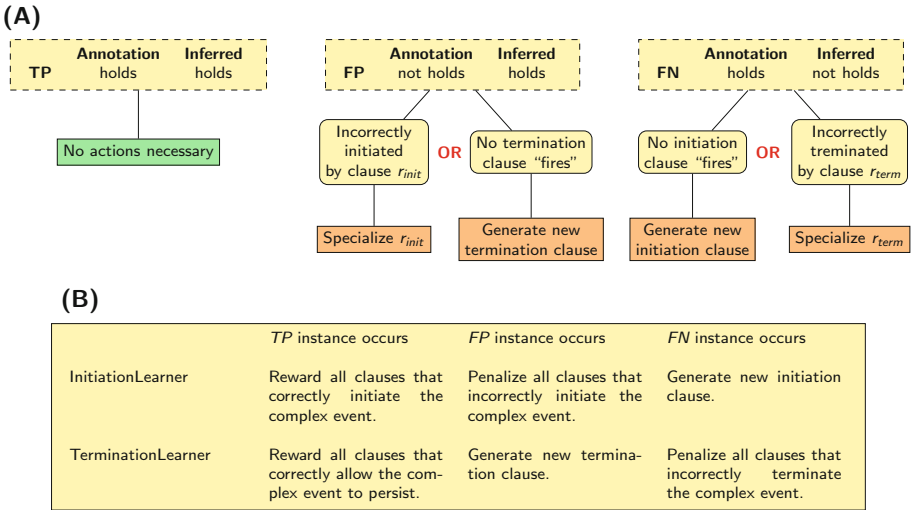
OLED [18] learns a theory by joining together independently-constructed clauses, each of which is learnt in an online fashion. It relies on the Hoeffding bound [15] to approximate the quality of a clause on the entire input using only a subset of the data. Given a random variable  $X$  with range in  $[0, 1]$  and an observed mean  $\bar{X}$  of its values after  $n$  independent observations, the Hoeffding Bound states that, with probability  $1 - \delta$ , the true mean  $\hat{X}$  of the variable lies in an interval  $(\bar{X} - \epsilon, \bar{X} + \epsilon)$ , where  $\epsilon = \sqrt{\frac{\ln(1/\delta)}{2n}}$ . In other words, the true average can be approximated by the observed one with probability  $1 - \delta$ , given an error margin  $\epsilon$  that becomes smaller as the number of observations  $n$  increases.

OLED learns a clause in a top-down fashion, by specializing it using literals from a bottom clause [8]. The Hoeffding bound is utilized in the specialization process as follows. Given a clause evaluation function  $G$  and some clause  $r$ , OLED evaluates  $r$  and all of its candidate specializations on training interpretations that stream-in, counting positive and negative examples in these interpretations that are covered by each of these clauses. Assume that after  $n$  examples,  $r_1$  is  $r$ 's specialization with the highest observed mean  $G$ -score  $\bar{G}$  and  $r_2$  is the second-best one, i.e.  $\Delta\bar{G} = \bar{G}(r_1) - \bar{G}(r_2) > 0$ . Then by the Hoeffding bound we have that for the true mean of the scores' difference  $\Delta\hat{G}$  it holds that  $\Delta\hat{G} > \Delta\bar{G} - \epsilon$ , with probability  $1 - \delta$ , where  $\epsilon = \sqrt{\frac{\ln(1/\delta)}{2n}}$ . Hence, if  $\Delta\bar{G} > \epsilon$  then  $\Delta\hat{G} > 0$ , implying that  $r_1$  is indeed the best specialization, with probability  $1 - \delta$ . In order to decide which specialization to select, it thus suffices to accumulate example counts from the incoming interpretations until  $\Delta\bar{G} > \epsilon$ . These interpretations need not be stored or reprocessed. Each interpretation is processed once to extract the necessary statistics for calculating  $G$ -scores and is subsequently discarded, thus giving rise to an online (single-pass) clause construction strategy. To ensure that no clause  $r$  is replaced by a specialization of lower quality,  $r$  itself is also considered as a potential candidate along with its specializations, ensuring that specializing  $r$  is a better decision, with probability  $1 - \delta$ , than not specializing it at all.

The default specialization process follows a FOIL-like, hill-climbing strategy, where a single literal is added to a clause at each specialization step. However, OLED supports different specialization strategies as well, e.g. by allowing to simultaneously try all specializations up to a given clause length, or by supporting user-defined, TILDE-like look-ahead specifications [7].

To calculate  $G$ -scores, each clause  $r$  is equipped with a true positive ( $TP$ ), a false positive ( $FP$ ) and a false negative ( $FN$ ) counter, whose values are updated accordingly as  $r$  gets evaluated on training interpretations that stream-in. True negative counts are not taken into account, since the annotation for complex events is acquired via the closed world assumption. Although different scoring functions may be plugged into OLED, in this work we use precision to score initiation clauses and recall to score termination clauses, as in [18]. Moreover, OLED supports a clause pruning mechanism, that allows to remove low-quality clauses (e.g. clauses that have been generated from noisy interpretations) and a tie-breaking mechanism, that allows to randomly select between equally good specializations. We refer to [18] for more details on these features.

In the general case, a theory learnt by OLED is a collection of clauses constructed with the online mechanism described above. A clause is generated from a positive example in an incoming interpretation, by constructing a bottom clause



**Fig. 1.** (A) Different behaviors of initiation and termination clauses w.r.t. to occurrences of  $TP$ ,  $FP$  and  $FN$  complex event instances. Dash-lined boxes explain what it means to encounter a  $TP$ ,  $FP$ ,  $FN$  complex event instance, in terms of (dis)agreement between the actual label of the instance and the one inferred by the theory. Round-cornered boxes describe the causes of  $FP$ ,  $FN$  occurrences w.r.t. the different types of clause (initiation or termination). Regular boxes at the “leaves” of the tree-like structures indicate proper courses of action in order to eliminate  $FP/FN$  instances. (B) Actions taken by the two different processes that learn initiation and termination clauses in parallel, w.r.t.  $TP$ ,  $FP$ ,  $FN$  complex event occurrences. These actions are in accordance with the indicated actions in (A) (leaves of the trees). “Rewarding” a clause refers to increasing the  $TP$  count of the clause, while “penalizing” a clause refers to increasing its  $FP$  or  $FN$  counts. Penalizing clauses reduces their score, it therefore contributes to their specialization after a sufficient number of examples has been seen.

$\perp$  from that instance and adding the empty-bodied clause “ $r = head(\perp) \leftarrow$ ” to theory  $H$ . From that point on,  $r$  is gradually specialized by the addition of literals from  $\perp$  to its body. New clauses are added to  $H$  whenever existing clauses in  $H$  become too specific to account for positive examples in new incoming interpretations. Bottom clause construction is preceded by an abductive process that handles the fact that target predicates (`initiatedAt/2` and `terminatedAt/2`) differ from observed annotation predicates (`holdsAt/2`). We refer to [18] for more details.

When learning domain-specific axioms in the Event Calculus, the aforementioned generic theory construction strategy must be modified to account for the fact that initiation and termination clauses behave differently w.r.t. encountered  $TP$ ,  $FP$  and  $FN$  complex event instances. A description of this behavior is illustrated in Fig. 1(A). To handle this behavior, initiation and termination clauses are learnt separately, by two parallel processes, each of which runs the core OLED Algorithm. The input stream is forwarded to each of these processes. Figure 1(B) presents the different actions that each learner takes whenever it encounters  $TP$ ,  $FP$  and  $FN$  instances.

## 4 A Parallel Version of OLED

We now proceed to the description of a data parallel version of OLED, which we henceforth denote by  $\rho$ -OLED. The parallelization strategy is based on evaluating a clause and its candidate specializations on incoming interpretations by distributing the workload across multiple processing nodes that operate on sub-streams of the input stream. When a node is about to specialize or remove a clause  $r$ , it consults its peer nodes and combines their evaluation results for  $r$  with its own, so that a more informed decision is made. We next describe this strategy in more detail.

### 4.1 Main Operations of the Parallel OLED Strategy

We assume that learning is performed by a set  $\mathcal{N}$  of independent processing nodes. Each node handles a sub-stream  $S_i$  of training interpretations, generated from an input stream  $\mathcal{S}$ , according to some data distribution scheme. For instance, the data from  $\mathcal{S}$  may be distributed to the processing nodes in  $\mathcal{N}$  in a “round-robin” manner, or by using specific data attributes as “key” in the distribution process. Processing nodes communicate by exchanging messages and they collaborate in order to learn a theory  $H$  in parallel. In particular,  $\rho$ -OLED differs from the sequential algorithm in the following respects:

**New clause generation:** When a node  $N_i$  generates a new clause  $r$ , it broadcasts  $r$  to all other nodes in  $\mathcal{N}$ , via an `AddNewClause( $r$ )` message (see Table 3 for the main types of message of parallel OLED). Each node that receives such a message adds clause  $r$  to its own theory and starts scoring  $r$ , and its candidate

**Table 3.** The main messages exchanged between data processing nodes in parallel OLED.

Message	Conditions for message broadcast	Actions upon message receipt
<b>AddNewClause</b> ( $r$ )	Generation of clause $r$	Add $r$ to local theory
<b>SpecializeRequest</b> ( $r_{id}$ )	Clause with id $r_{id}$ is about to be specialized (the Hoeffding test for this clause has succeeded)	Reply to the sender by the local $TP, FP, FN, E$ counts for clause with id $r_{id}$ and for each of its candidate specializations
<b>SpecializeReply</b> ( $args$ ), where $args = \langle r_{id}, TP, FP, FN, E \rangle$	Reply to a specialization request message for the clause with id $r_{id}$	Add the received counts for the corresponding clause to the local ones and repeat the Hoeffding test
<b>Replace</b> ( $r_{id}, r'$ )	Clause with id $r_{id}$ has been specialized to clause $r'$	Replace clause with id $r_{id}$ by $r'$ in local theory
<b>PruneRequest</b> ( $r_{id}$ )	Clause with id $r_{id}$ is about to be pruned	Reply to the sender by the local $TP, FP, FN$ counts for clause with id $r_{id}$ , as well as the period for which $r$ remains (locally) unchanged
<b>PruneReply</b> ( $args$ ), where $args = \langle r_{id}, TP, FP, FN, T \rangle$ , $T$ being the period for which the clause with id $r_{id}$ remained unchanged at the sender node	Reply to a prune request message	Add the received counts for the corresponding clause to the local ones and repeat the clause removal test
<b>Remove</b> ( $r_{id}$ )	Clause with id $r_{id}$ has been pruned	Remove clause with id $r_{id}$ from local theory

specializations on its local data stream. As in the sequential version of OLED, a new clause  $r$  consists of an initially empty-bodied clause “ $head(\perp) \leftarrow$ ”, where  $\perp$  is a bottom clause generated at  $N_i$ , which is subsequently used to gradually specialize  $r$ .

**Clause specialization:** When a node  $N_i$  is about to specialize a clause  $r$ , i.e. when OLED’s Hoeffding test for clause  $r$  succeeds, locally at  $N_i$ , node  $N_i$  sends a **SpecializeRequest**( $r_{id}$ ) message to all other nodes, where  $r_{id}$  is a unique identifier of clause  $r$ , common to all copies of  $r$  shared among processing nodes. Upon receiving such a message, each node uses  $r_{id}$  to retrieve its own evaluation statistics for clause  $r$  and its candidate specializations, which are sent over to the requesting node  $N_i$ . These statistics consist of  $TP, FP, FN$  and  $E$  counts for clause  $r$  and its candidate specializations, where by  $E$  we denote the number of examples on which a clause has been evaluated so far (number of groundings of target predicates). The received counts for clause  $r$  and its specializations are combined with node  $N_i$ ’s local counts as follows (we describe the process for clause  $r$  only, but it is similar for each of its specializations). Denoting by  $TP_r^j, FP_r^j, FN_r^j$  and  $E_r^j$  the respective counts for clause  $r$ , received from node  $N_j \in \mathcal{N}, j \neq i$ , the current node  $N_i$  updates  $r$ ’s counts accordingly, by increasing

$r$ 's local counts with those received from other nodes. For instance, the new  $TP$  count for clause  $r$  in node  $N_i$  becomes  $TP_r^i = TP_r^i + \sum_{N_j \in \mathcal{N}} TP_r^j$ .

Each processing node  $N_i \in \mathcal{N}$  maintains a record, for each clause  $r$  in its theory and each one of  $r$ 's specializations, that contains the exact counts previously received for them, from each node  $N_j \in \mathcal{N}, j \neq i$ . When node  $N_i$  receives a set of new  $TP_r^j, FP_r^j, FN_r^j$  and  $E_r^j$  counts for clause  $r$  from node, the respective previous counts are subtracted from the new ones, to avoid over-scoring  $r$  with counts that have already been taken into account in previous updates. The same holds for  $r$ 's specializations.

Once individual clause evaluation statistics are combined as described above, node  $N_i$  repeats the Hoeffding test for clause  $r$  to assess if the test still succeeds after the accumulated counts have been taken into account. If it does, clause  $r$  is replaced in  $H$ , the current theory at node  $N_i$ , by its best-scoring specialization  $r'$  that results from the Hoeffding test. Then, node  $N_i$  sends out a **Replace**( $r_{id}, r'$ ) message to all other nodes, instructing them to also replace their local copy that corresponds to  $r_{id}$  in their own theories with  $r'$ . If, on the other hand, the Hoeffding test fails at node  $N_i$  after the updated counts are taken into account, clause  $r$  is not specialized.

**Clause pruning:** For a clause  $r$  to be removed, two conditions must hold: First, clause  $r$  must be unchanged (not specialized) for a sufficiently long period, which, in the single-core version of OLED, is set to the average number of examples for which the Hoeffding test succeeds, i.e. the average value of  $n = \mathcal{O}(\frac{1}{\epsilon^2} \ln \frac{1}{\delta})$  that has resulted in clause specializations so far. Second, from that point on where clause  $r$  remains unchanged, a sufficiently large number of data must be seen, in order to use a Hoeffding test to infer that, with probability  $1 - \delta$ , the quality of clause  $r$  is below the pruning threshold, i.e. a user-defined lower bound on the quality of acceptable clauses.

In  $\rho$ -OLED, each node uses the above heuristics to decide locally whether a clause  $r$  should be pruned. Once it has seen enough data from its own stream to make that decision for clause  $r$ , it sends a **PruneRequest**( $r_{id}$ ) message to all other nodes. Each node that receives such a message sends back to the requesting node the necessary information (period for which clause  $r$  remains unchanged and  $TP, FP, FN$  and  $E_r$  counts for clause  $r$ ), which node  $N_i$  uses to re-assess whether clause  $r$  should be pruned, based on the global view of clause  $r$ , obtained by combining  $r$ 's separate evaluations from all processing nodes. If node  $N_i$  eventually decides to prune clause  $r$ , it sends a **Remove**( $r_{id}$ ) to all other nodes, which instructs them to also remove clause  $r$  from their theories.

## 4.2 Decentralized Coordination

Each processing node in  $\rho$ -OLED operates autonomously and there is no centralized coordination. This may result in undesired behavior, therefore some extra actions are in order, at an implementation level, to avoid such behavior. We next discuss such issues and outline the way that  $\rho$ -OLED handles them.



Clause specialization requires some coordination between processing nodes. For instance, assume that node  $N_j$  handles a `SpecializeRequest` for some clause  $r$ , sent from a node  $N_i$ .  $N_j$  sends  $r$ 's evaluation statistics to the requesting node  $N_i$  and it subsequently continues to process data from its local training stream. This implies that during the time taken for node  $N_i$  to decide on  $r$ 's specialization (receive the statistics for  $r$  and its candidate specializations from all nodes and repeat the Hoeffding test), node  $N_j$  continues to evaluate clause  $r$  on its own data. It is possible that during this time the Hoeffding test for clause  $r$  succeeds at node  $N_j$ , in which case it will attempt to specialize  $r$ . This is unnecessary, since  $r$ 's specialization is already under assessment at node  $N_i$ . To avoid this behaviour and ensure that a potential specialization of a clause  $r$  is handled by a single node at a time, each recipient node of a `SpecializeRequest` message “marks” the clause in question as a specialization candidate. For a marked clause  $r$ , all potential specialization attempts are temporarily suspended, until a “verdict” for this clause is received from the node that is currently attempting to specialize clause  $r$ .

The above strategy is insufficient in cases where the Hoeffding test for clause  $r$  succeeds at more than one processing nodes simultaneously, or at a very close temporal proximity. In such cases, a node  $N_i$  may need to handle a `SpecializeRequest` for some clause  $r$ , while currently attempting itself to specialize  $r$ , implying that two nodes are attempting to specialize the same clause simultaneously. Consider for instance a situation where node  $N_i$  has just finished processing an interpretation where the Hoeffding test for clause  $r$  succeeded, while in the meantime, a `SpecializeRequest` message for the same clause  $r$ , sent from some other node  $N_j$ , has been enqueued in  $N_i$ 's message queue. To resolve conflicts in cases like these, a priority order is imposed beforehand on all processing nodes, using each node's index  $k$ ,  $1 \leq k \leq |\mathcal{N}|$ . Nodes of higher index are prioritized to specialize a clause  $r$  over nodes of lower index. That is, a node of index  $k$  abandons its effort to specialize a clause  $r$  whenever it encounters a `SpecializeRequest` message for the same clause, received from a node with index  $k' > k$ . Similarly, nodes of higher index do not serve specialization requests for a clause  $r$ , received from nodes of lower index, in case they themselves are already attempting to specialize  $r$ .

A similar coordination mechanism is used to ensure that a potential removal of a low-quality clause during pruning is handled by a single node at a time.

Another cause for unwanted behaviour is related to delays in message passing. For instance, a node  $N_i$  may be currently carrying out an intensive, time-consuming task (e.g. processing a large and complex interpretation), while some other node  $N_j$  is expecting  $N_i$ 's reply on a `SpecializeRequest` message, in order to specialize some clause  $r$ . This results in  $N_j$  “wasting data”, since it keeps processing new interpretations during this time. These data could have been used to evaluate new specializations for clause  $r$ , had node  $N_i$  responded in a timely fashion. To avoid situations like these, in practice each node uses a time-out parameter  $t$  to handle the replies of its peers nodes, considering only the replies received within the time-out. The time-out parameter is adapted during the learning process according to the mean processing time per training interpretation.

## 5 Empirical Evaluation

We present an experimental evaluation of our approach on CAVIAR (described in Sect. 2), a benchmark dataset for activity recognition. CAVIAR contains 282,067 training interpretations with a mean size of 25 atoms each.  $\rho$ -OLED is implemented in the Scala programming language. It uses Clingo<sup>2</sup> as its main reasoning component and Scala’s akka Actors library<sup>3</sup> to model the behavior of a processing node and implement message passing. The code and data of the empirical analysis are available online<sup>4</sup>. All experiments were conducted on a Debian Linux machine with a 3.6 GHz processor (4 cores and 8 threads) and 16 GB of RAM.

The purpose of our first experiment was to compare  $\rho$ -OLED with its monolithic counterpart. We performed learning with 1, 2, 4 and 8 processing threads (each representing a processing node) for constructing the definitions of two complex events, related to two persons *meeting each other* or *moving together*. We used tenfold cross-validation with an 80%–20% training-testing ratio. CAVIAR contains 6,272 positive interpretations for *moving* (i.e. interpretations where *moving* occurs) and 3,722 positive interpretations for *meeting*, forming respectively 12 positive sequences for *moving* and 11 positive sequences for *meeting* (a positive (resp. negative) sequence encompasses a time interval where a complex event holds (resp. does not hold) continuously). The testing set for each fold of the cross-validation process consisted of 2 positive sequences per complex event, plus negative sequences amounting to the 20% of the total negatives in the dataset, while the remaining positive and negative sequences were used for training. For this experiment positive and negative sequences in the training set were evenly distributed across the different processing nodes, so that all nodes were fed with approximately the same number of positive and negative examples. In each fold of the cross-validation process, the training interpretations were presented to each processing node in a random order. The parameters for both the sequential and the parallel version of OLED was  $\delta = 10^{-5}$  and clause pruning threshold set to 0.65 for *moving* and 0.8 for *meeting*. These values were chosen empirically based on previous experiments with OLED on the CAVIAR dataset [18]. The pruning threshold values refer to precision for initiation clauses and recall for termination clauses, which were used as scoring functions in this experiment.

We also created a larger version of CAVIAR, in order to evaluate our algorithms on a more demanding learning task. This dataset consists of 10 copies of the original CAVIAR dataset, where each copy differs from the others only in the constants referring to the tracked entities (persons, objects) that appear in simple and complex events. This dataset contains 100 different tracked entities, as compared to only 10 entities of the original CAVIAR dataset. In each copy of the dataset, the coordinates of each entity  $p$  differ by a fixed offset from the coordinates of the entity of the original dataset that  $p$  mirrors. The setting for the x10-CAVIAR experiment was as described above.

<sup>2</sup> <http://potassco.sourceforge.net/>.

<sup>3</sup> <http://akka.io/>.

<sup>4</sup> <https://github.com/nkatzz/OLED>.

The results from our experiment with CAVIAR and x10-CAVIAR are presented in Table 4(A) and (B) respectively, in the form of averages (over the ten runs of the cross-validation process) for training time,  $F_1$ -score and theory size (total number of literals), as well as average number of exchanged messages.  $F_1$ -scores were obtained by micro-averaging results from each fold. We also present  $F_1$ -scores and theory sizes for hand-crafted theories for the two target complex events. The hand-crafted theories may be considered as the standard in this domain and they are presented in [5]. They are available online<sup>5</sup>, in addition to learnt theories for *meeting* and *moving* with OLED. Table 4 also presents the achieved speed-ups for  $\rho$ -OLED, defined as  $T_1/T_n$ , where  $T_1$  and  $T_n$  are respectively the training times of a monolithic and a parallel learner that uses  $n$  cores. The speed-up is linear if it's approximately equal to  $n$ , for each  $n$ , while it is sub-linear (resp. super-linear) if it is smaller (resp. greater) than  $n$ .

Starting with the results from the CAVIAR dataset (Table 4(A)), we see that  $\rho$ -OLED constructed theories of slightly higher  $F_1$ -score for *meeting*, as compared to its single-core counterpart. In the monolithic setting, OLED postpones

**Table 4. (A)** Experimental results from the CAVIAR dataset; **(B)** Experimental results from the x10-CAVIAR dataset.

		#cores	Time (sec)	Speed-up	$F_1$ -score	Theory size	#Msgs	
<b>(A)</b>	<i>Meet</i>	1	46	–	0.798	28	–	
		2	18	2.5	<b>0.818</b>	31	<b>75</b>	
		4	15	<b>3</b>	0.805	34	168	
		8	<b>15</b>	<b>3</b>	0.802	35	358	
	<i>HandCrafted</i>	–	–	–	0.700	<b>24</b>	–	
	<i>Move</i>	1	68	–	<b>0.744</b>	<b>21</b>	–	
		2	31	2.1	0.740	<b>21</b>	<b>58</b>	
		4	27	2.5	0.739	<b>21</b>	112	
		8	<b>26</b>	<b>2.6</b>	0.743	23	228	
	<i>HandCrafted</i>	–	–	–	0.732	28	–	
	<b>(B)</b>	<i>Meet</i>	1	7588	–	<b>0.834</b>	36	–
			2	2144	3.5	<b>0.834</b>	36	<b>78</b>
4			1682	4.5	<b>0.834</b>	36	158	
8			<b>912</b>	<b>8.3</b>	0.832	36	342	
<i>HandCrafted</i>		–	–	–	0.700	<b>24</b>	–	
<i>Move</i>		1	7898	–	<b>0.758</b>	34	–	
		2	2312	3.4	0.753	34	<b>82</b>	
		4	1788	4.4	0.756	34	164	
		8	<b>966</b>	<b>8.1</b>	0.753	34	322	
<i>HandCrafted</i>		–	–	–	0.732	<b>28</b>	–	

<sup>5</sup> <http://users.iit.demokritos.gr/~nkatz/CAVIAR-theories/>.

the generation of new clauses, up to the point where existing clauses become too specific to account for new examples in the incoming interpretations. During this time, interpretations which may result in a good clause (recall that OLED learns by “encoding” interpretations into bottom clauses), are “skipped”, i.e. they are not used for learning new clauses, since they are covered by existing ones. In contrast, the data distribution in  $\rho$ -OLED resulted in cases where interesting interpretations that would have been missed in the monolithic setting, are actually used for learning. A similar effect was not observed for *moving*, which has a simpler definition than *meeting*.

Regarding training times, OLED achieves a significant speed-up for both complex events, by moving from sequential learning to learning with 2 cores, but from that point on, training times do not improve proportionally to the number of cores, resulting in sub-linear speed-ups. This is not the case however in the x10-CAVIAR experiment (Table 4(B)), where  $\rho$ -OLED achieves super-linear speed-ups. The x10-CAVIAR dataset consists of larger training interpretations, each containing an increased number of domain constants and ground literals. Such interpretations are significantly harder to be reasoned upon, as indicated by the exponential growth in training times in the x10-CAVIAR experiment. Therefore, the contrast between the speed-up patterns of the regular CAVIAR and the significantly larger x10-CAVIAR version seems to be in line with the fact that gains by parallelizing an ILP algorithm are often observed only when significant data volumes are involved [12, 28, 30]. Additionally, the reported behavior seems to imply that the gain in efficiency of our proposed parallel learning strategy increases with the difficulty of the learning task at hand, in terms of the “unit cost” of processing individual interpretations.

Due to the increase in training data size in the x10-CAVIAR experiment,  $F_1$ -scores for all runs (number of cores) are improved as compared to the regular CAVIAR experiment and they seem to converge. For example, in the regular CAVIAR experiment, good rules were often constructed “too-late”, from interpretations that were encountered shortly before the data were exhausted. Such rules may be discarded, since OLED (and its parallel version) use a “warm-up” period parameter that controls a minimum number of interpretations a rule must be evaluated on, in order to be included in an output hypothesis. In contrast, in the x10-CAVIAR experiment such problems were avoided, thanks to the increase in training data size.

We performed an additional experiment where the goal was to assess the effect of uneven data distribution on the amount of communication, total training time and  $F_1$ -score. The experimental setting was similar to the one described previously, i.e. a tenfold cross-validation process with an 80% – 20% training-testing ratio. One of the nodes, however, handled a larger data load than its peers. To introduce the imbalance we used an external data distribution processes that takes as input an imbalance parameter  $k$ . This process reads the data from disk, in the actual order in which they appear in the CAVIAR videos, and forwards training interpretations to processing nodes as follows: The first  $k$  interpretations are forwarded to the first node. Subsequently, each one of the

**Table 5.** Effects on the imbalance in data load on CAVIAR, using 8 processing nodes.

	Imbalance	Time (sec)	#Msgs	$F_1$ -score	Theory size
<i>Meet</i>	10	<b>16</b>	348	0.802	<b>34</b>
	50	24	327	<b>0.808</b>	<b>34</b>
	100	41	<b>298</b>	0.799	<b>34</b>
<i>Move</i>	10	<b>24</b>	231	0.739	22
	50	38	212	0.741	<b>23</b>
	100	52	<b>191</b>	<b>0.742</b>	<b>23</b>

following  $N - 1$  interpretations ( $N$  being the number of used nodes) is forwarded to one of the remaining  $N - 1$  nodes. The next  $k$  interpretations are forwarded again to the first node and so on, so that the first node eventually handles  $k$ -times more data than its peer nodes. In the process of data distribution, data sequences that are intended to be used for testing are “skipped” (they are not forwarded to any node).

We performed experiments in this setting with 8 processing nodes and three different values for the imbalance parameter  $k = 10, 50, 100$ . The results are presented in Table 5 in the form of averages from the tenfold cross-validation process for total training time, number of messages,  $F_1$ -score (micro-averaged over all folds) and theory size. As the imbalance parameter  $k$  grows, training time increases slightly, while the amount of communication drops. The increase in training time may be explained by the “bottleneck” of a single node handling larger data loads sequentially, as the imbalance increases, while the drop in the total number of exchanged messages is due to the fact the majority of the processing nodes, which handle fewer training data, also broadcast fewer messages, as compared to the scenario where data are evenly distributed between nodes. Regarding the  $F_1$ -score and the theory size, only small changes are reported with respect to the results of Table 4(A). These differences are attributed to the different order in which training interpretations are presented to p-OLED in the two experiments.

## 6 Related Work

An overview of existing approaches to learning theories in the Event Calculus with ILP may be found in [16, 17] and a discussion on how OLED compares to such approaches may be found in [16, 18]. In this section we mainly discuss parallel ILP algorithms, for which a substantial amount of work exists in the literature. A thorough review may be found in [12, 30]. Parallel ILP algorithms exploit parallelism across three main axes [12]: Searching through the hypothesis space in parallel (search parallelism); splitting the training data and learning from data subsets (data parallelism); and evaluating candidate clauses in parallel (evaluation/coverage parallelism).

In [28] the authors present a data-parallel version of a standard set-cover loop: Each processing node learns a fragment of the concept definition from a partition of the data, and then these fragments are exchanged between all nodes. Good-enough clauses are kept by all nodes. A cover removal step is subsequently implemented by each core and the set-cover loop continues. Overall, the approach in [28] learns much faster than a sequential algorithm, achieving super-linear speed-ups. A similar approach is proposed in [11], where the training interpretations are split across multiple nodes and searched in parallel, while the best rules from each node are “pipe-lined” to all other nodes.

In [30] the authors use a MapReduce-based framework to parallelize the operation of a classical set-cover ILP algorithm towards both evaluation-parallelism and search-parallelism. In the former case, coverage tests of candidate clauses are performed in parallel, on disjoint partitions of the data. In the latter case, bottom clauses (which are generalized to acquire a hypothesis clause) are generated and searched in a concurrent fashion from more than one “seed” examples. The reducer then selects the best hypothesis clause that results from this process. A similar approach for parallel exploration of independent hypotheses has been proposed in [22], while similar approaches towards parallel coverage tests have been proposed in [10,14]. In [21], the approach of [30] was extended to a framework that is capable of self-regulating the workload of distributing learning costs across multiple nodes. In [25,26] the authors propose a strategy for collaborative learning of action models. The problem is modelled in a multi-agent systems context, where autonomous agents communicate with each other and revise their local models in an effort to establish global consistency of the latter. An important difference of this work is that the communication is based on the exchange of examples, as opposed to clauses, which is the case with  $\rho$ -OLED and most of the works described above. Finally, some work exists in the literature on parallelizing (unsupervised) relational data-mining tasks, such as frequent pattern mining [2,3].

A main difference of the work presented here from the aforementioned approaches to parallel ILP is that they mostly rely on iterative ILP algorithms, which require several passes over the data to compute a hypothesis. In contrast, OLED is an online, single-pass algorithm. In relation to the latter, some work on streaming ILP exists. However, existing approaches are either oriented towards unsupervised tasks like frequent pattern discovery [27], or they rely on propositionalization techniques and off-the-self, online propositional learners [29].

## 7 Conclusions and Future Work

We presented a parallel version of a recently proposed algorithm for online learning of event definitions in the form of Event Calculus theories. We also presented an experimental evaluation of our approach on a benchmark dataset for activity recognition, which demonstrates that it can reduce training times, while also achieving super-linear speed-ups on some occasions. As future work, we aim to evaluate our approach in a distributed setting and on larger datasets, in terms of

in-situ, geographically distributed learning, as required in maritime monitoring [24]. We also plan to formally analyze the behavior of the proposed approach in terms of communication cost, comparison to its monolithic counterpart in terms of convergence and convergence speed, as well as comparison to similar learning strategies that adopt different communication protocols.

**Acknowledgments.** This work is funded by the H2020 project datAcron (687591).

## References

1. Alevizos, E., Skarlatidis, A., Artikis, A., Paliouras, G.: Probabilistic complex event recognition: a survey. *ACM Comput. Surv.* (2018, to appear)
2. Appice, A., Ceci, M., Turi, A., Malerba, D.: Sampling very large databases for parallel and distributed relational frequent pattern discovery. In: *First International Workshop on Ubiquitous Knowledge Discovery Workshop (2008)*
3. Appice, A., Ceci, M., Turi, A., Malerba, D.: A parallel, distributed algorithm for relational frequent pattern discovery from very large data sets. *Intell. Data Anal.* **15**(1), 69–88 (2011)
4. Artikis, A., Sergot, M., Paliouras, G.: An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.* **27**(4), 895–908 (2015)
5. Artikis, A., Skarlatidis, A., Paliouras, G.: Behaviour recognition from video content: a logic programming approach. *Int. J. Artif. Intell. Tools* **19**(2), 193–209 (2010)
6. Artikis, A., Skarlatidis, A., Portet, F., Paliouras, G.: Logic-based event recognition. *Knowl. Eng. Rev.* **27**(4), 469–506 (2012)
7. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. *Artif. Intell.* **101**(1), 285–297 (1998)
8. De Raedt, L.: *Logical and Relational Learning*. Springer, Heidelberg (2008). <https://doi.org/10.1007/978-3-540-68856-3>
9. Etzion, O., Niblett, P.: *Event Processing in Action*. Manning Publications Co., Greenwich (2010)
10. Fidjeland, A.K., Luk, W., Muggleton, S.H.: Customisable multi-processor acceleration of inductive logic programming. In: *Latest Advances in Inductive Logic Programming*, pp. 123–141 (2014)
11. Fonseca, N.A., Silva, F.M.A., Costa, V.S., Camacho, R.: A pipelined data-parallel algorithm for ILP. In: *2005 IEEE International Conference on Cluster Computing (CLUSTER 2005)*, Boston, Massachusetts, USA, 26–30 September 2005, pp. 1–10 (2005)
12. Fonseca, N.A., Srinivasan, A., Silva, F., Camacho, R.: Parallel ILP for distributed-memory architectures. *Mach. Learn.* **74**(3), 257–279 (2009)
13. Gama, J.: *Knowledge Discovery from Data Streams*. CRC Press, Florida (2010)
14. Graham, J.H., David Page Jr., C., Kamal, A.H.: Accelerating the drug design process through parallel inductive logic programming data mining. In: *2nd IEEE Computer Society Bioinformatics Conference, CSB 2003*, Stanford, CA, USA, 11–14 August 2003, pp. 400–402 (2003)
15. Hoeffding, W.: Probability inequalities for sums of bounded random variables. *J. Am. Stat. Assoc.* **58**(301), 13–30 (1963)
16. Katzouris, N.: *Scalable relational learning for event recognition*. Ph.D. thesis, University of Athens (2017). <http://users.iit.demokritos.gr/~nkatz/papers/nkatz-phd.pdf>

17. Katzouris, N., Artikis, A., Paliouras, G.: Incremental learning of event definitions with inductive logic programming. *Mach. Learn.* **100**(2–3), 555–585 (2015)
18. Katzouris, N., Artikis, A., Paliouras, G.: Online learning of event definitions. *TPLP* **16**(5–6), 817–833 (2016)
19. Kowalski, R., Sergot, M.: A logic-based calculus of events. *New Gener. Comput.* **4**(1), 67–95 (1986)
20. Muggleton, S.: Inverse entailment and prolog. *New Gener. Comput.* **13**(3&4), 245–286 (1995)
21. Nishiyama, H., Ohwada, H.: Yet another parallel hypothesis search for inverse entailment. In: *ILP* (2015)
22. Ohwada, H., Mizoguchi, F.: Parallel execution for speeding up inductive logic programming systems. In: Arikawa, S., Furukawa, K. (eds.) *DS 1999. LNCS (LNAI)*, vol. 1721, pp. 277–286. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-46846-3\\_25](https://doi.org/10.1007/3-540-46846-3_25)
23. Paschke, A., Bichler, M.: Knowledge representation concepts for automated SLA management. *Decis. Support Syst.* **46**(1), 187–205 (2008)
24. Patroumpas, K., Alevizos, E., Artikis, A., Vodas, M., Pelekis, N., Theodoridis, Y.: Online event recognition from moving vessel trajectories. *GeoInformatica* **21**(2), 389–427 (2017)
25. Rodrigues, C., Soldano, H., Bourgne, G., Rouveirol, C.: A consistency based approach of action model learning in a community of agents. In: *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2014, Paris, France, 5–9 May 2014*, pp. 1557–1558 (2014)
26. Rodrigues, C., Soldano, H., Bourgne, G., Rouveirol, C.: Multi agent learning of relational action models. In: *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18–22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pp. 1087–1088 (2014)
27. Silva, A., Antunes, C.: Multi-relational pattern mining over data streams. *Data Min. Knowl. Disc.* **29**(6), 1783–1814 (2015)
28. Skillicorn, D.B., Wang, Y.: Parallel and sequential algorithms for data mining using inductive logic. *Knowl. Inf. Syst.* **3**(4), 405–421 (2001)
29. Srinivasan, A., Bain, M.: Relational models with streaming ILP. In: *ILP* (2013)
30. Srinivasan, A., Faruque, T.A., Joshi, S.: Data and task parallelism in ILP using mapreduce. *Mach. Learn.* **86**(1), 141–168 (2012)





# Relational Restricted Boltzmann Machines: A Probabilistic Logic Learning Approach

Navdeep Kaur<sup>1</sup>, Gautam Kunapuli<sup>2</sup>(✉) , Tushar Khot<sup>3</sup>, Kristian Kersting<sup>4</sup> ,  
William Cohen<sup>5</sup>, and Sriraam Natarajan<sup>2</sup> 

<sup>1</sup> Indiana University, Bloomington, USA  
navdkaur@indiana.edu

<sup>2</sup> The University of Texas at Dallas, Richardson, USA  
{gautam.kunapuli, sriraam.natarajan}@utdallas.edu

<sup>3</sup> Allen Institute of Artificial Intelligence, Seattle, USA  
tushar.v.khot@gmail.com

<sup>4</sup> Technische Universität Darmstadt, Darmstadt, Germany  
kersting@cs.tu-darmstadt.de

<sup>5</sup> Carnegie Mellon University, Pittsburgh, USA  
wcohen@gmail.com

**Abstract.** We consider the problem of learning Boltzmann machine classifiers from relational data. Our goal is to extend the deep belief framework of RBMs to statistical relational models. This allows one to exploit the feature hierarchies and the non-linearity inherent in RBMs over the rich representations used in statistical relational learning (SRL). Specifically, we use lifted random walks to generate features for predicates that are then used to construct the observed features in the RBM in a manner similar to Markov Logic Networks. We show empirically that this method of constructing an RBM is comparable or better than the state-of-the-art probabilistic relational learning algorithms on six relational domains.

## 1 Introduction

Restricted Boltzmann machines (RBMs, [30]) are popular models for learning probability distributions due to their expressive power. Consequently, they have been applied to various tasks such as collaborative filtering [39], motion capture [41] and others. Similarly, there has been significant research on the theory of RBMs: approximating log-likelihood gradient by contrastive divergence (CD, [17]), persistent CD [42], parallel tempering [11], extending them to handle real-valued variables and discriminative settings. While these models are powerful, they make the standard assumption of using *flat feature vectors* to represent the problem.

In contrast to flat-feature representations, Statistical Relational Learning (SRL, [9, 15]) methods use richer symbolic features during learning; however, they have not been fully exploited in deep-learning methods. Learning SRL

models is computationally intensive [33] however, particularly model structure (qualitative relationships). This is due to the fact that structure learning requires searching over objects, their attributes, and attributes of related objects. Hence, the state-of-the-art learning method for SRL models learns a series of weak relational rules that are combined during prediction. While empirically successful, this method leads to rules that are dependent on each other making them uninterpretable, since weak rules cannot always model rich relationships that exist in the domain. For instance, a weak rule could say something like: “a professor is popular if he teaches a course”. When learning discriminatively, this rule could have been true if some professors teach at least one course, while at least one not so popular professor did not teach a course in the current data set. We propose to use a set of interpretable rules based on the successful Path Ranking Algorithm (PRA, [28]). Recently, Hu et al. [20] employed logical rules to enhance the representation of neural networks. There has also been work on lifting neural networks to relational settings [4]. While specific methodologies differ, all these methods employ relational and logic rules as features of neural networks and train them on relational data. In this spirit, we propose a methodology for lifting RBMs to relational data. While previous methods on lifting relational networks employed logical constraints or templates, we use relational random walks to construct relational rules, which are then used as features in a RBM. Specifically, we consider random walks constructed by the PRA approach of Lao and Cohen [28] to develop features that can be trained using RBMs. We consider the formalism of discriminative RBMs as our base classifier and use these relational walks with the base classifier.

We propose two approaches to instantiating RBM features: (1) similar to the approach of Markov Logic Networks (MLNs, [12]) and Relational Logistic Regression (RLR, [21]), we instantiate features with *counts* of the number of times a random walk is satisfied for every training example; and (2) similar to Relational Dependency Networks (RDNs, [32]), we instantiate features with *existentials* (1 if  $\exists$  at least one instantiation of the path in the data, otherwise 0). Given these features, we train a discriminative RBM with the following assumptions: the input layer is multinomial (to capture counts and existentials), the hidden layer is sigmoidal, and the output layer is Bernoulli.

We make the following contributions: (1) we combine the powerful formalism of RBMs with the representation ability of relational logic; (2) we develop a *relational RBM* that does not fully propositionalize the data; (3) we show the connection between our proposed method and previous approaches such as RDNs, MLNs and RLR, and (4) we demonstrate the effectiveness of this novel approach by empirically comparing against state-of-the-art methods that also learn from relational data.

The rest of the paper is organized as follows: Sect. 2 presents the background on relational random walks and RBMs, Sect. 3 present our RRBm approach and algorithms in detail, and explore its connections to some well-known probabilistic relational models. Section 4 presents the experimental results on standard relational data sets. Finally, the last section concludes the paper by outlining future research directions.

## 2 Prior Work and Background

In this article, we represent relational data using standard first-order logic notation. We refer to objects of a particular type as *constants* of that type. Relations in the domain are called *predicates* and the true relations in the data are called *ground atoms*.

### 2.1 Random Walks

Relational data is often represented using a ground (or lifted) graph. The constants (or types) form the nodes and the ground atoms (or predicates) form the edges.  $N$ -ary predicates can be represented with hyperedges or multiple binary relations, where a node is introduced for every ground atom (or predicate) and edges are introduced from this node to each argument. This graph representation allows the use of many path-based approaches for discovering the structure of the data. A path in a ground relational graph (where nodes are constants) corresponds to a conjunction of ground atoms. For example, the path  $s_1 - \text{takes} - c_1 - \text{taughtBy} - p_1$  describes an example where the student  $s_1$  takes class  $c_1$  taught by professor  $p_1$ . In a ground relational graph, this path can be converted to:  $\text{takes}(s_1, c_1) \wedge \text{taughtBy}(c_1, p_1)$ . In contrast, in a lifted relational graph (where nodes are types), paths are conjunctions of predicates with shared variables:  $\text{takes}(S, C) \wedge \text{taughtBy}(C, P)$ .

### 2.2 Relational Probabilistic Models

Markov Logic Networks (MLNs, [12]) are relational undirected models, where first-order logic formulas correspond to cliques of a Markov network, and formula weights correspond to the clique potentials. An MLN can be instantiated as a Markov network with a node for each ground predicate (atom) and a clique for each ground formula. All groundings of the same formula are assigned the same weight leading to the following joint probability distribution over all atoms:  $P(X=x) = \frac{1}{Z} \exp(\sum_i w_i n_i(x))$ , where  $n_i(x)$  is the number of times the  $i$ -th formula is satisfied by possible world  $x$ , and  $Z$  is a normalization constant. Intuitively, a possible world where formula  $f_i$  is true one more time than a different possible world is  $e^{w_i}$  times as probable, all other things being equal. We focus on discriminative learning, where we learn a conditional distribution of one predicate given all other predicates.

Another such discriminative model is relational logistic regression (RLR, [21]), which extends logistic regression to relational settings, and where training examples can have differing feature sizes. An interesting observation is that RLR can be considered as an aggregator when there are multiple values for the same set of features.

### 2.3 Structure Learning Approaches

Many structure learning approaches for Probabilistic Logical Models (PLMs), including MLNs, use graph representations. For example, Learning via Hypergraph Lifting (LHL, [23]) builds a hypergraph over ground atoms; LHL then

clusters the atoms to create a “lifted” hypergraph, and traverses this graph to obtain rules. Specifically, they use depth-first traversal to create the paths in this “lifted” hypergraph to create potential clauses by using the conjunction of predicates from the path as the body of the clause.

Learning with Structural Motifs (LSM, [24]) performs random walks over the graph to cluster nodes and performs depth-first traversal to generate potential clauses. We use *random walks over a lifted graph* to generate all possible clauses, and then use a non-linear combination (through the hidden layer) of ground clauses, as opposed to linear combination in MLNs. Our hypothesis space includes the clauses generated by both these approaches without the additional complexity of clustering the nodes.

## 2.4 Propositionalization Approaches

To learn powerful deep models on relational data, propositionalization is used to convert ground atoms into a fixed-length feature vector. For instance, kFoil [27] uses a *dynamic* approach to learn clauses to propositionalize relational examples for SVMs. Each clause is converted into a Boolean feature that is 1, if an example satisfies the clause body and each clause is scored based on the improvement of the SVM learned using the clause features. Alternately, the Path Ranking Algorithm (PRA) [28], which has been used to perform knowledge base completion, creates features for a pair of entities by generating random walks from a graph. We use a similar approach to perform random walks on the lifted relational graph to learn the structure of our relational model.

## 2.5 Restricted Boltzmann Machines

Boltzmann machines (BMs, [30]) model probability distributions and are interpretable as artificial neural networks [1]. A BM consists of visible units  $V$  (representing observations) and hidden units  $H$  (representing dependencies between features). A general BM is a fully-connected Markov random field [26], which makes learning computationally intensive. A more tractable model, the Restricted Boltzmann Machine (RBM), constrains the BM to a bipartite graph of visible and hidden units. A singular benefit of this representation is that hidden-layer outputs of one RBM can be used as input to another higher-level RBM, a procedure known as *stacking*. Stacking uses RBMs as building blocks to construct deep belief networks (DBNs) with multiple layers of non-linear transformations of input data; this results in powerful deep belief networks [18].

RBMs have been used as feature extractors for supervised learning [14] and to initialize deep neural networks [19]. Larochelle and Bengio [29] proposed a standalone formulation for supervised classification called *discriminative RBMs*. We adopt this formalism to demonstrate our approach of combining rich first-order logic representations of relational data with nonlinear classifiers learned by discriminative RBMs.

### 3 Relational Restricted Boltzmann Machines

Reconsider MLNs, arguably one of the leading relational approaches unifying logic and probability. The use of relational formulas as features within a log-linear model allows the exploitation of “deep” knowledge. Nevertheless, this is still a shallow architecture as there are no “hierarchical” formulas defined from lower levels. The hierarchical stacking of layers, however, is the essence of deep learning and, as we demonstrate in this work, critical for relational data, even more than for propositional data. This is due to one of the key features of relational modeling: predictions of the model may depend on the number of individuals, that is, the population size. Sometimes this dependence is desirable, and in other cases, model weights may need to change. In either case, it is important to understand how predictions change with population size when modeling or even learning the relational model [21].

We now introduce Relational RBMs, a deep, relational classifier that can learn hierarchical relational features through its hidden layer and model non-linear decision boundaries. The idea is to use *lifted random walks* to generate relational features for predicates that are then counted (or used as existentials) to become RBM features. Of course, more than one RBM could be trained, stacking them on top of each other. For the sake of simplicity, we focus on a single layer; however, our approach is easily extended to multiple layers. Our learning task can be defined as follows:

**Given:** Relational data,  $D$ ; Target Predicate,  $T$ .

**Learn:** Relational Restricted Boltzmann Machine (RRBM) in a discriminative fashion.

We are given data,  $D = \{(\mathbf{x}_i, \hat{y}_i)_{i=1}^{\ell}\}$ , where each training example is a vector,  $\mathbf{x}_i \in \mathbb{R}^m$  with a multi-class label,  $\hat{y}_i \in \{1, \dots, C\}$ . The training labels are represented by a one-hot vectorization:  $\mathbf{y}_i \in \{0, 1\}^C$  with  $y_i^k = 1$  if  $\hat{y}_i = k$  and zero otherwise. For instance, in a three-class problem, if  $\hat{y}_i = 2$ , then  $\mathbf{y}_i = [0, 1, 0]$ . The goal is to train a classifier by maximizing the log-likelihood,  $\mathcal{L} = \sum_{i=1}^{\ell} \log p(\mathbf{y}_i, \mathbf{x}_i)$ . In this work, we employ discriminative RBMs, for which we make some key modeling assumptions:

1. input layers (relational features) are modeled using a multinomial distribution, for counts or existentials;
2. the output layer (target predicate) is modeled using a Bernoulli distribution
3. hidden layers are continuous, with a range in  $[0, 1]$ .

#### 3.1 Step 1: Relational Data Representation

We use a lifted-graph representation to model relational data,  $D$ . Each type corresponds to a node in the graph and the predicate  $\mathbf{r}(\mathbf{t}_1, \mathbf{t}_2)$  is represented by a directed edge from the node  $\mathbf{t}_1$  to  $\mathbf{t}_2$  in the graph. For  $N$ -ary predicates, say  $\mathbf{r}(\mathbf{t}_1, \dots, \mathbf{t}_n)$ , we introduce a special compound value type (CVT)<sup>1</sup>,  $\mathbf{r}_{\text{CVT}}$ , for each

<sup>1</sup> [wiki.freebase.com/wiki/Compound.Value.Type](http://wiki.freebase.com/wiki/Compound.Value.Type).

n-ary predicate. For each argument  $\mathbf{t}_k$ , an edge  $\mathbf{e}_{\mathbf{r}k}$  is added between the nodes  $\mathbf{r}_{\text{CVT}}$  and  $\mathbf{t}_k$ . Similarly for unary predicates,  $\mathbf{r}(\mathbf{t})$  we create a binary predicate  $\text{isa}(\mathbf{t}, \mathbf{r})$ .

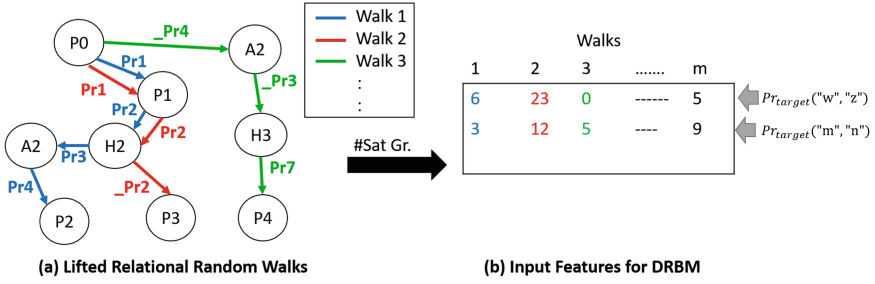
### 3.2 Step 2: Relational Transformation Layer

Now, we generate the input feature vector  $\mathbf{x}_i$  from a relational example,  $T(\mathbf{a}_{1j}, \mathbf{a}_{2j})$ . Inspired by the Path Ranking Algorithm [28], we use random walks on our lifted relational graph to encode the local relational structure for each example. We generate  $m$  unique random walks connecting the argument types for the target predicate to define the  $m$  dimensions of  $\mathbf{x}$ . Specifically, starting from the node for the first argument’s type, we repeatedly perform random walks till we reach the node for the second argument. Since random walks also correspond to the set of candidate clauses considered by structure-learning approaches for MLNs [23,24], this transformation function can be viewed as the *structure of our relational model*.

A key feature of an RBM trained on standard i.i.d. data is that the feature set  $\mathbf{x}$  is defined in advance and is finite. With relational data, this set can potentially be infinite, and feature size can vary with *each training instance*. For instance, if the random walk is a paper written by a **professor – student** combination, not all **professor – student** combinations will have the same number of feature values. This is commonly referred as *multiple-parent* problem [34]. To alleviate this problem, SRL methods consider one of two approaches – aggregators or combining rules. Aggregators combine multiple values to a single value, while combining rules combine multiple probability distributions into one. While these solutions are reasonable for traditional probabilistic models that estimate distributions, they are not computationally feasible for the current task.

Our approach to the multiple-parent problem is to consider *existential semantics*: if there exists *at least one instance of the random walk that is satisfied for an example*, the feature value corresponding to that random walk is set to 1 (otherwise, to 0). This approach was also recently (and independently of our work) used by Wang and Cohen [43] for ranking via matrix factorization. This leads to our first model: RRBm-Existentials, or RRBm-E, where E denotes the existential semantics used to construct the RRBm. One limitation of RRBm-E is that it does not differentiate between a **professor – student** combination that has only one paper and another that has 10 papers, that is, it does not take into account how often a relationship is true in the data. Inspired by MLNs, we also consider *counts of the random walks* as feature values, a model we denote RRBm-Counts or RRBm-C (Fig. 1). For example, if a **professor – student** combination has written 10 papers, the feature value corresponding to this random walk for that combination is 10. To summarize, we define two transformation functions,  $\mathbf{x}_j = g(\mathbf{a}_{1j}, \mathbf{a}_{2j})$

- $g_e(\mathbf{a}_{1j}, \mathbf{a}_{2j}, \mathbf{p}) = 1$ , if  $\exists$  a grounding of the  $p^{\text{th}}$  random walk connecting object  $\mathbf{a}_{1j}$  to object  $\mathbf{a}_{2j}$ , otherwise 0 (RRBm-E);
- $g_c(\mathbf{a}_{1j}, \mathbf{a}_{2j}, \mathbf{p}) = \#\text{groundings of } p^{\text{th}} \text{ random walk connecting object } \mathbf{a}_{1j} \text{ to object } \mathbf{a}_{2j}$  (RRBm-C).



**Fig. 1.** Lifted random walks are converted into feature vectors by explicitly grounding every random walk for every training example. Nodes and edges of the graph in (a) represent types and predicates, and underscore ( $\_Pr$ ) represents the inverted predicates. The random walks counts (b) are then used as feature values for learning a discriminative RBM (DRBM). An example of random walk represented as clause is (c).

For example, consider that the walk  $takes(S, C) \wedge taughtBy(C, P)$  is used to generate a feature for  $advisedBy(s_1, p_1)$ . The feature from  $g_c$  would be set to the count:  $|\{C \mid takes(s_1, C) \wedge taughtBy(C, p_1)\}|$ . With the function,  $g_e$ , this feature would be set to 1, if  $\exists C, takes(s_1, C) \wedge taughtBy(C, p_1)$ .

These transformation functions also allow us to relate our approach to other well-known relational models. For instance,  $g_c$  uses counts similar to MLNs, while  $g_e$  uses existential semantics similar to RDNs [32]. Using features from  $g_e$  to learn weights for a logistic regression model would lead to an RLR model, while using features from  $g_c$  would correspond to learning an MLN (as we show later). One could also imagine using RLR as an aggregator from these random walks, but that is a direction for future work. While counts are more informative and connect to existing SRL formalisms such as MLNs, exact counting is computationally expensive in relational domains. This can be mitigated by using approximate counting approaches, such as the one due to [7] that leverages the power of graph databases. Our empirical evaluation did not require count approximations; we defer integration of approximate counting to future research.

### 3.3 Step 3: Learning Relational RBMs

The output of the relational transformation layer is fed into multilayered discriminative RBM (DRBM) to learn a regularized, non-linear, weighted combination of features. The relational transformation layer stacked on top of the DRBM forms the Relational RBM model. Due to non-linearity, we are able to learn a much more expressive model than traditional MLNs and RLRs. Recall that the DRBM as defined by [29] consists of  $n$  hidden units,  $\mathbf{h}$ , and the joint probability

is modeled as  $p(\mathbf{y}, \mathbf{x}, \mathbf{h}) \propto e^{-E(\mathbf{y}, \mathbf{x}, \mathbf{h})}$ , where the energy function is parameterized  $\Theta \equiv (W, \mathbf{b}, \mathbf{c}, \mathbf{d}, U)$ :

$$E(\mathbf{y}, \mathbf{x}, \mathbf{h}) = -\mathbf{h}^T W \mathbf{x} - \mathbf{b}^T \mathbf{x} - \mathbf{c}^T \mathbf{h} - \mathbf{d}^T \mathbf{y} - \mathbf{h}^T U \mathbf{y}. \quad (1)$$

As with most generative models, computing the joint probability  $p(\mathbf{y}, \mathbf{x})$  is intractable, but the conditional distribution  $P(\hat{y}|\mathbf{x})$  can be computed exactly [39] as

$$p(\hat{y}|\mathbf{x}) = \frac{e^{d_{\hat{y}} + \sum_{j=1}^n \sigma(c_j + U_{j\hat{y}} + \sum_{f=1}^m W_{jf} x_{jf})}}{\sum_{k=1}^C e^{d_k + \sum_{j=1}^n \sigma(c_j + U_{jk} + \sum_{f=1}^m W_{jf} x_{jf})}}. \quad (2)$$

In (2),  $\sigma(z) = e^z / \sum_i e_i^z$ , the logistic softmax function and the index  $f$  sums over all the features  $x_f$  of a training example  $\mathbf{x}$ . During learning, the log-likelihood function is maximized to compute the DRBM parameters  $\Theta$ . The gradient of the conditional probability (Eq. 2) can be computed as:

$$\frac{\partial}{\partial \theta} \log p(\hat{y}_i | \mathbf{x}_i) = \sum_{j=1}^n \sigma(o_{\hat{y}_j}(\mathbf{x}_i)) \frac{\partial o_{\hat{y}_j}(\mathbf{x}_i)}{\partial \theta} + \sum_{k=1}^C \sum_{j=1}^n \sigma(o_{kj}(\mathbf{x}_i)) p(k|\mathbf{x}_i) \frac{\partial o_{kj}(\mathbf{x}_i)}{\partial \theta}. \quad (3)$$

In (3),  $o_{\hat{y}_j}(\mathbf{x}_i) = c_j + U_{j\hat{y}} + \sum_{f=1}^m W_{jf} x_{i,f}$ , where  $\mathbf{x}$  refers to random-walk features for every training example. As mentioned earlier, we assume that input features are modeled using a multinomial distribution. To consider counts as multinomials, we use an upper bound on counts:  $2 \max(\text{count}(x_i^j))$  for every feature; bounds are the same for both train and test sets to avoid overfitting. In other words, the bound is simply twice the max feature count over all the examples of the training set. We can choose the scaling factor through cross-validation, but value 2 seems to be a reasonable scale in our experiments. For the test examples, we can use the random walks to generate the features and the RBM layers to generate predictions from these features.

**RRBM Algorithm:** The complete approach to learn Relational RRBMs is shown in Algorithm 1. In Step 1, we generate type-restricted random walks using PRA. These random walks ( $\mathbf{rw}$ ) are used to construct the feature matrix. For each example, we obtain exact counts for each random walk, which becomes the corresponding feature value for that example. A DRBM can be trained on the features as explained in Step 3.

### 3.4 Relation to Probabilistic Relational Models

The random walks can be interpreted as logical clauses (that are used to generate features) and the DRBM input feature weights  $\mathbf{b}$  in (1) can be interpreted as clause weights ( $w_p$ ). This interpretation highlights connections between our approach and Markov logic networks. Intuitively, the relational transformation layer captures the structure of MLNs and the RBM layer captures the weights of the MLNs. More concretely,  $\exp(\mathbf{b}^T \mathbf{x})$  in (1) can be viewed as  $\exp(\sum_p w_p n_p(\mathbf{x}))$



---

**Algorithm 1.** LearnRRBM( $T, G, P$ ): Relational Restricted Boltzmann Machines

**Input**  $T(\mathbf{t}_1, \mathbf{t}_2)$ : target predicate,  $G$ : lifted graph over types,  $m$ : number of features

---

```

1:  $\triangleright$  Generate  $m$  random walks between  $\mathbf{t}_1$  and  $\mathbf{t}_2$ 
2:  $\mathbf{rw} :=$  PerformRandomWalks( $G, \mathbf{t}_1, \mathbf{t}_2, m$ )
3: for  $0 \leq j < l$  do  $\triangleright$  Iterate over all training examples
4:    $\triangleright$  Generate features for  $T(\mathbf{a}_{1j}, \mathbf{a}_{2j})$ 
5:   for  $0 \leq p < m$  do  $\triangleright$  Iterate over all the paths
6:      $\triangleright p^{th}$  feature computed from the arguments of  $x_j$ 
7:      $x_j[p] := g_c(\mathbf{a}_{1j}, \mathbf{a}_{2j}, \mathbf{rw}[p])$ 
8:   end for
9: end for
10:  $\mathbf{x} := \{\mathbf{x}_j\}$   $\triangleright$  Input matrix
11:  $\triangleright$  Learn DRBM from the features and examples
12:  $\Theta :=$  LearnDRBM( $\mathbf{x}, \mathbf{y}$ )
13: return RRBm( $\Theta, \mathbf{rw}$ )

```

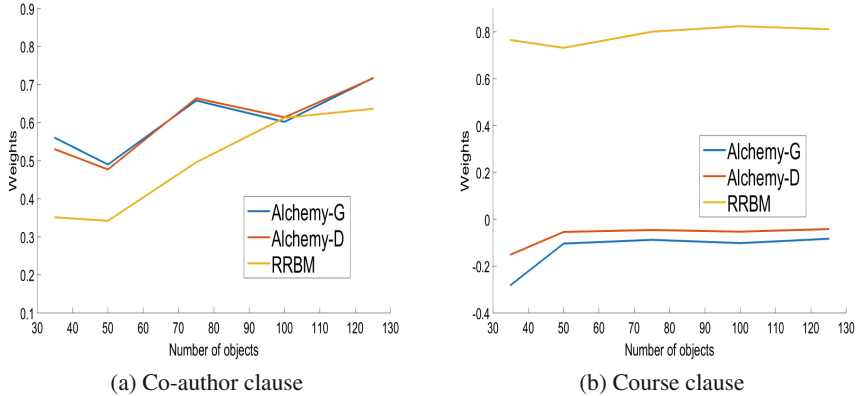
---

in the probability distribution for MLNs. To verify this intuition, we compare the weights learned for clauses in MLNs to weights learned by RRBm-C. We generated a synthetic data set for a university domain with varying number of objects (professors and students). We picked a subset of **professor** – **student** pairs to have an **advisedBy** relationship and add common papers or common courses based on the following two clauses:

1.  $\text{author}(A, P) \wedge \text{author}(B, P) \rightarrow \text{advisedBy}(A, B)$
2.  $\text{teach}(A, C) \wedge \text{registered}(B, C) \rightarrow \text{advisedBy}(A, B)$

The first clause states that if a professor  $A$  co-authors a paper  $P$  with the student  $B$ , then  $A$  advises  $B$ . The second states that if a student  $B$  registers for a course  $C$  taught by professor  $A$  then  $A$  advises  $B$ . Figure 2 shows the weights learned by discriminative and generative weight learning in Alchemy and RRBm for these two clauses as a function of the number of objects in the domain. Recall that in MLNs, the weight of a rule captures the confidence in that rule — the higher the number of instances satisfying a rule, the higher is the weight of the rule. As a result, the weight of the rule learned by Alchemy also increases in Fig. 2. We observe a similar behavior with the weight learned for this feature in our RRBm formulation as well. While the exact values differ due to difference in the model formulation, this illustrates clearly that the intuitions of the model parameters from standard PLMs are still applicable.

In contrast to standard PLMs, RRBms are not a shallow architecture. This can be better understood by looking at the rows of the weights  $W$  in the energy function (1): they act as additional filter features, combining different clause counts. That is,  $E(\mathbf{y}, \mathbf{x}, \mathbf{h})$  looks at how well the usage profile of a clause aligns with different filters associated with rows  $W_j$ . These filters are shared across different clauses, but different clauses will make comparisons with different filters by controlling clause-dependent biases  $U_{jy}$  in the  $\sigma$  terms. Notice also, that two



**Fig. 2.** Weights learned by Alchemy and RRBM for a clause vs. size of the domain.

similar clauses could share some filters in  $W$ , that is, both could simultaneously have large positive values of  $U_{jy}$  for some rows  $W_j$ . This can be viewed as a form of statistical predicate invention as it discovers new concepts and is akin to (discriminative) second-order MLNs. In contrast to second-order MLNs, however, no second-order rules are required as input to discover new concepts. While MLNs can learn arbitrary  $N$ -ary target predicates, due to the definition of random walks in the original work, we are restricted to learning binary relations.

## 4 Experiments

To compare RRBM approaches to state-of-the-art algorithms, we consider RRBM-E, RRBM-C and RRBM-CE. The last approach, RRBM-CE combines features from both existential and count RRBM (i.e., union of count and existential features). Our experiments seek to answer the following questions:

- Q1:** How do RRBM-E and RRBM-C compare to baseline MLNs and Decision Trees?
- Q2:** How do RRBM-E and RRBM-C compare to the state-of-the-art SRL approaches?
- Q3:** How do RRBM-E, RRBM-C, and RRBM-CE generalize across all domains?
- Q4:** How do random-walk generated features compare to propositionalization?

To answer **Q1**, we compare RRBM to Learning with Structural Motifs (LSM, [24]). Specifically, we perform structure learning with LSM followed by weight learning with Alchemy [25] and denote this as MLN. We would also like to answer the question: how crucial is it to use a RBM, and not some other ML algorithm? We use decision trees [36] as a proof-of-concept for demonstrating that a good probabilistic model when combined with our random walk features can potentially yield better results than naive combination of ML algorithm with features. We denote the decision tree model **Tree-C**. For LSM, we used the parameters recommended by [24]. However, we set the maximum path length of

random walks of LSM structure learning to 6 to be consistent with the maximum path length used in RRBM. We used both discriminative and generative weight-learning forAlchemy and present the best-performing result.

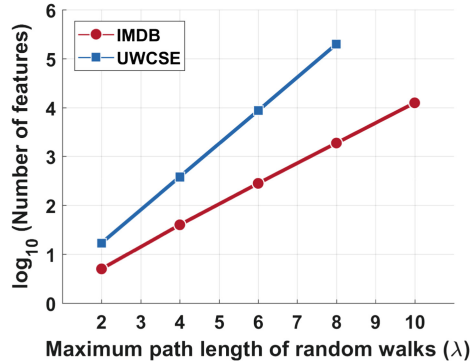
To answer **Q2**, we compare RRBM-C to MLN-Boost [22], and RRBM-E to RDN-Boost [32] both of which are SRL models that learn the structure and parameters simultaneously. For MLN-Boost and RDN-Boost, we used default settings and 20 gradient steps. For RRBM, since path-constrained random walks [28] are performed on binary predicates, we convert unary and ternary predicates into binary predicates. For example, predicates such as `teach(a1, a2, a3)` are converted to three binary predicates: `teachArg1(id, a1)`, `teachArg2(id, a2)`, `teachArg3(id, a3)` where `id` is the unique identifier for a predicate. As another example, unary predicates such as `student(s1)` are converted to binary predicates of the form `isa(s1, student)`. To ensure fairness, we used binary predicates as inputs to all the methods considered here. We also allow inverse relations in random walks, that is, we consider a relation and its inverse to be distinct relations. For one-to-one and one-to-many relations, this sometimes leads to uninteresting random walks of the form `relation`  $\rightarrow$  `relation`<sup>-1</sup>  $\rightarrow$  `relation`. In order to avoid this situation, we add additional sanity constraints on walks that prevent relations and their inverses from immediately following one another and avoid loops.

To answer **Q4**, we compare our method with Bottom Clause Propositionalization [13] (BCP-RBM), which generates one bottom clause for each example and considers each atom in the body of the bottom clause to be a unique feature. We utilize Progol [38] to generate bottom clauses by using its default configuration but setting variable depth = 1 to handle large data sets. Contrary to the original work [13] that uses a neural network, we use RBM as the learning model, as our goal is to demonstrate the usefulness of random walks to generate features.

In our experiments, we subsample training examples at a 2 : 1 ratio of negatives to positives during training. The number of RBM hidden nodes are set to 60% of visible nodes, the learning rate,  $\eta = 0.05$  and the number of epochs to 5. These hyperparameters have been optimized by line search.

**A Note On Hyperparameter Selection:** An important hyperparameter for RRBM is the maximum path length of random walks, which influences the number of RRBM features. Figure 3 shows that the number of features generated grows exponentially with maximum path length. We restricted the maximum path length of random walks to  $\lambda = 6$  in order to strike a balance between tractability and performance;  $\lambda = 6$  demonstrated consistently good performance across a variety of data sets, while keeping the feature size tractable. As mentioned above, other benchmark methods such as LSM were also restricted to a maximum random walk length of 6 for consistency and fairness.

Hyperparameter selection is an open issue in both relational learning as well as deep learning; in the latter, careful tuning of hyperparameters and architectures such as regularization constants and number of layers is critical. Recent work on automated hyperparameter selection can also be used with RRBM, if a more systematic approach to hyperparameter selection for RRBM is desired,



**Fig. 3.** The number of RRBM features grows exponentially with maximum path length of random walks. We set  $\lambda = 6$  to balance tractability with performance.

especially in practical settings. Bergstra and Bengio [2] demonstrated that random search is more efficient for hyperparameter optimization than grid search or manual tuning. This approach can be used to select optimal  $\eta$  and  $\lambda$  jointly. Snoek et al. [40] recently used Bayesian optimization for automated hyperparameter tuning. While this approach was shown to be highly effective across diverse machine learning formalisms including for support vector machines [6], latent Dirichlet allocation [3] and convolutional neural networks [16], it requires powerful computational capabilities and parallel processing to be feasible in practical settings.

#### 4.1 Data Sets

We used several benchmark data sets to evaluate the performance of our algorithms. We compare several approaches using conditional log-likelihood (CLL), area under ROC curve (AUC-ROC), and area under precision-recall curve (AUC-PR). Measuring PR performance on skewed relational data sets yields a more conservative view of learning performance [8]. As a result, we use this metric to report statistical significant improvements at  $p = 0.05$ . We employ 5-fold cross validation across all data sets.

**UW-CSE:** The UW-CSE data set [37] is a standard benchmark that consists of predicates and relations such as `professor`, `student`, `publication`, `hasPosition` and `taughtBy` etc. The data set contains information from five different areas of computer science about professors, students and courses, and the task is to predict the `advisedBy` relationship between a professor and a student. For MLNs, we present results from generative weight learning as it performed better than discriminative weight learning.

**Mutagenesis:** The MUTAGENESIS data set<sup>2</sup> has two entities: `atom` and `molecule`, and consists of predicates that describe attributes of atoms and

<sup>2</sup> [cs.sfu.ca/~oschulte/BayesBase/input](http://cs.sfu.ca/~oschulte/BayesBase/input).

molecules, as well as the types of relationships that exist between atom and molecule. The target predicate is `moleatm(aid,mid)`, to predict whether a molecule contains a particular atom. For MLN, we present generative weight learning as it had better results.

**Cora Entity Resolution** is a citation matching data set [35]; in the citation-matching problem, a “group” is a set of citations that refer to the same publication. Here, a large fraction of publications belong to non-trivial groups, that is, groups that have more than one citation; the largest group contains as many as 54 citations, which makes this a challenging problem. It contains the predicates such as `Author`, `Title`, `Venue`, `HasWordTitle`, `SameAuthor` and the target predicate is `SameVenue`. Alchemy did not complete running after 36 h and therefore we report results from [22].

**IMDB:** This data set was first created by Mihalkova and Mooney [31] and contains nine predicates: `gender`, `genre`, `movie`, `samegender`, `samegenre`, `samemovie`, `sameperson`, `workedunder`, `actor` and `director`; we predict the `workedUnder` relation. Since `actor` and `director` are unary predicates, we converted them to one binary predicate `isa(person,designation)` where designation can take two values - `actor` and `director`. For MLNs, we report the generative weight learning results here.

**Yeast:** Contains millions of facts [28] from papers published between 1950 and 2012 on the yeast organism *Saccharomyces cerevisiae*. It includes predicates like `gene`, `journal`, `author`, `title`, `chem`, etc. The target predicate is `cites`, that is, we predict the citation link between papers. As in the original paper, we need to prevent models from using information obtained later than the publication date. While calculating features for a citation link, we only considered facts that were earlier than a publication date. Since we cannot enforce this constraint in LSM, we do not report Alchemy results for Yeast.

**Sports:** NELL [5] is an online<sup>3</sup> never-ending learning system that extracts information from online text data, and converts this into a probabilistic knowledge base. We consider NELL data from the `sports` domain consisting of information about players and teams. The task is to predict whether a team plays a particular sport or not. Alchemy did not complete its run after 36 h, thus we do not report its result for this data set.

## 4.2 Results

**Q1:** Figure 4 compares our approaches to baseline MLNs and decision trees to answer **Q1**. RRBm-E and RRBm-C have significant improvement over `Tree-C` on UW and Yeast data sets, with comparable performance on the other four. Across all data sets (except Cora) and all metrics, RRBm-E and RRBm-C beat the baseline MLN approach. Thus, we can answer **Q1** affirmatively: RRBm models outperform baseline approaches in most cases.

<sup>3</sup> [rtw.ml.cmu.edu/rtw/](http://rtw.ml.cmu.edu/rtw/).

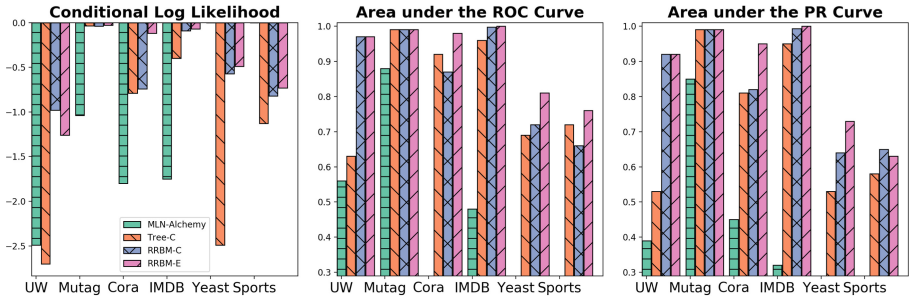


Fig. 4. (Q1): Results show that RRBM's generally outperform baseline MLN and decision-tree (Tree-C) models.

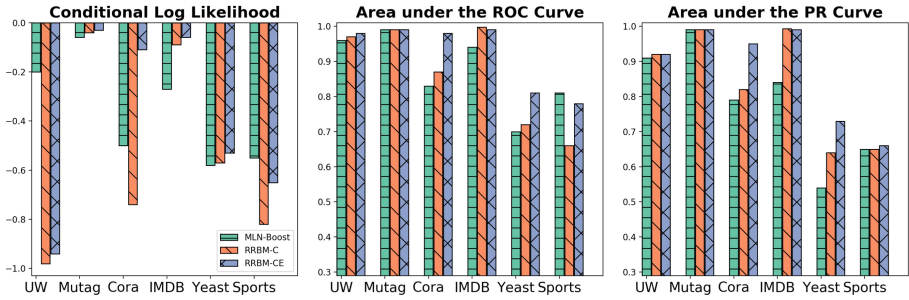


Fig. 5. (Q2) Results show better or comparable performance of RRBM-C and RRBM-CE to MLN-Boost, which all use counts.

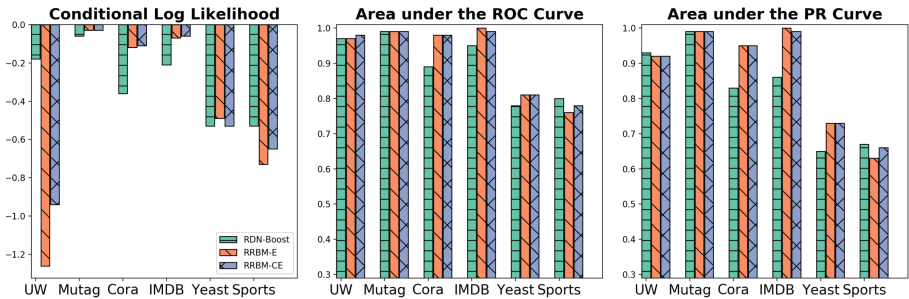
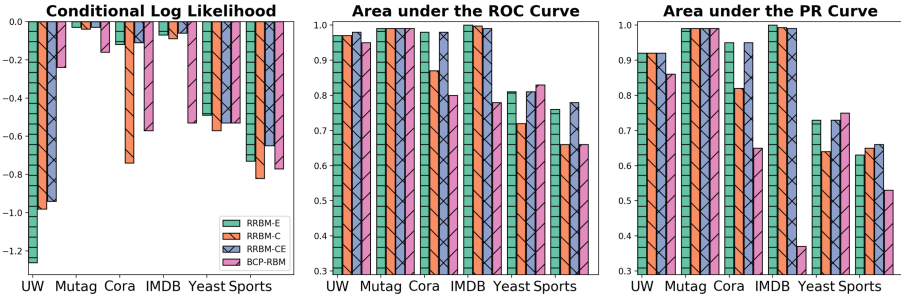


Fig. 6. (Q2) Results show better or comparable performance of RRBM-E and RRBM-CE to RDN-Boost, which all use existentials.



**Fig. 7. (Q4)** Results show better or comparable performance of our random-walk-based feature generation approach (RRBM) compared to propositionalization (BCP-RBM).

**Q2:** We compare RRBM-C to MLN-Boost (count-based models) and RRBM-E to RDN-Boost (existential-based models) in Figs. 5 and 6. Compared to MLN-Boost on CLL, RRBM-C has a statistically significant improvement or is comparable on all data sets. RRBM-E is comparable to RDN-Boost on all the data sets with statistical significant CLL improvement on Cora. We also see significant AUC-ROC improvement of RRBM-C on Cora and RRBM-E on IMDB. Thus, we confirm that RRBM-E and RRBM-C are better or comparable to the current best structure learning methods.

**Q3:** Broadly, the results show that RRBM approaches generalize well across different data sets. The results also indicate that RRBM-CE generally improves upon RRBM-C and has comparable performance to RRBM-E. This shows that existential features are sufficient or better at modeling. This is also seen in the boosting approaches, where RDN-Boost (existential semantics), generally outperforms MLN-Boost (count semantics).

**Q4:** Since BCP-RBM only generates existential features, we compare BCP-RBM with RRBM-E to answer Q4. Figure 7 shows that RRBM-E has statistically significantly better performance than BCP-RBM on three data sets on CLL. Further, RRBM-E demonstrates significantly better performance than BCP-RBM on four data sets: Cora, Mutagenesis, IMDB and Sports - both on AUC-ROC and AUC-PR. This allows us to state positively that random-walk features yield better or comparable performance than propositionalization. For IMDB, BCP-RBM generated identical bottom clauses for all positive examples, resulting in an extreme case of just a single positive example to be fed into RBM. This results in a huge skew (distinctly observable in AUC-PR of IMDB for BCP-RBM).

## 5 Conclusion

Relational data and knowledge bases are useful in many tasks, but feeding them to deep learners is a challenge. To address this problem, we have presented a combination of deep and statistical relational learning, which gives rise to a powerful deep architecture for relational classification tasks, called Relational RBMs.

In contrast to propositional approaches that use deep learning features as inputs to log-linear models (e.g. [10]), we proposed and explored a paradigm connecting PLM features as inputs to deep learning. While statistical relational models depend much more on the discriminative quality of the clauses that are fed as input, Relational RBMs can learn useful hierarchical relational features through its hidden layer and model non-linear decision boundaries. The benefits were illustrated on several SRL benchmark data sets, where RRBMs outperformed state-of-the-art structure learning approaches—showing the tight integration of deep learning and statistical relational learning.

Our work suggests several interesting avenues for future work. First, one should explore stacking several RRBMs. Since the relational feature extraction is separated from the deep learning method, different types of deep learning methods can easily be used and should be explored. Alternatively, one could explore to jointly learn the underlying relational model and the deep model using stochastic EM. This “deep predicate invention” holds promise to boost relational learning. Ultimately, one should close the loop and feed the deep learning features back into a (relational) log-linear model.

**Acknowledgements.** Kristian Kersting gratefully acknowledges the support by the DFG Collaborative Research Center SFB 876 projects A6 and B4. Sriraam Natarajan gratefully acknowledges the support of the DARPA DEFT Program under the Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0039. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, ARO, AFRL, or the US government.

## References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. *Cogn. Sci.* **9**(1), 147–169 (1985)
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. *JMLR* **13**, 281–305 (2012)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *JMLR* **3**, 993–1022 (2003)
4. Blockeel, H., Uwents, W.: Using neural networks for relational learning. In: *ICML 2004 Workshop on SRL*, pp. 23–28 (2004)
5. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: *AAAI*, pp. 1306–1313 (2010)
6. Cristianini, N., Shawe-Taylor, J.: *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, New York (2000)
7. Das, M., Wu, Y., Khot, T., Kersting, K., Natarajan, S.: Scaling lifted probabilistic inference and learning via graph databases. In: *SDM* (2016)
8. Davis, J., Goadrich, M.: The relationship between Precision-Recall and ROC curves. In: *ICML* (2006)



9. De Raedt, L., Kersting, K., Natarajan, S., Poole, D. (eds.): *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Morgan and Claypool Publishers, San Rafael (2016)
10. Deng, L.: Connecting deep learning features to log-linear models. In: *Log-Linear Models, Extensions and Applications*. MIT Press (2015)
11. Desjardins, G., Courville, A., Bengio, Y., Vincent, P., Dellaleau, O.: Parallel tempering for training of restricted Boltzmann machines. *AISTATS* **9**, 145–152 (2010)
12. Domingos, P., Lowd, D.: *Markov Logic: An Interface Layer for AI*. Morgan & Claypool Publishers, San Rafael (2009)
13. França, M.V.M., Zaverucha, G., d’Avila Garcez, A.S.: Fast relational learning using bottom clause propositionalization with artificial neural networks. *Mach. Learn.* **94**(1), 81–104 (2014)
14. Gehler, P.V., Holub, A.D., Welling, M.: The rate adapting Poisson model for information retrieval and object recognition. In: *ICML*, pp. 337–344 (2006)
15. Getoor, L., Taskar, B.: *Introduction to Statistical Relational Learning*. MIT Press, Cambridge (2007)
16. Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. MIT Press, Cambridge (2016)
17. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. *Neural Comput.* **14**, 1771–1800 (2002)
18. Hinton, G.E., Osindero, S.: A fast learning algorithm for deep belief nets. *Neural Comput.* **18**, 1527–1554 (2006)
19. Hinton, G.E.: To recognize shapes first learn to generate images. In: *Computational Neuroscience: Theoretical Insights into Brain Function*. Elsevier (2007)
20. Hu, Z., Ma, X., Liu, Z., Hovy, E.H., Xing, E.P.: Harnessing deep neural networks with logic rules. In: *ACL* (2016)
21. Kazemi, S., Buchman, D., Kersting, K., Natarajan, S., Poole, D.: Relational logistic regression. In: *KR* (2014)
22. Khot, T., Natarajan, S., Kersting, K., Shavlik, J.: Learning Markov logic networks via functional gradient boosting. In: *ICDM* (2011)
23. Kok, S., Domingos, P.: Learning Markov logic network structure via hypergraph lifting. In: *ICML* (2009)
24. Kok, S., Domingos, P.: Learning Markov logic networks using structural motifs. In: *ICML* (2010)
25. Kok, S., Sumner, M., Richardson, M., et al.: *The Alchemy system for statistical relational AI*. University of Washington, Technical report (2010)
26. Koller, D., Friedman, N.: *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, Cambridge (2009)
27. Landwehr, N., Passerini, A., De Raedt, L., Frasconi, P.: Fast learning of relational kernels. *Mach. Learn.* **78**, 305–342 (2010)
28. Lao, N., Cohen, W.: Relational retrieval using a combination of path-constrained random walks. *J. Mach. Learn.* **81**(1), 53–67 (2010)
29. Larochelle, H., Bengio, Y.: Classification using discriminative restricted Boltzmann machines. In: *ICML*, pp. 536–543 (2008)
30. Lecun, Y., Chopra, S., Hadsell, R., Marc’Aurelio, R., Huang, F.: A tutorial on energy-based learning. In: *Predicting Structured Data*. MIT Press (2006)
31. Mihalkova, L., Mooney, R.: Bottom-up learning of Markov logic network structure. In: *ICML* (2007)

32. Natarajan, S., Khot, T., Kersting, K., Guttman, B., Shavlik, J.: Gradient-based boosting for statistical relational learning: the relational dependency network case. *MLJ* **86**(1), 25–56 (2012)
33. Natarajan, S., Khot, T., Kersting, K., Shavlik, J. (eds.): *Boosted Statistical Relational Learners. From Benchmarks to Data-Driven Medicine.* SpringerBriefs in Computer Science. Springer, New York (2016)
34. Natarajan, S., Tadepalli, P., Dietterich, T., Fern, A.: Learning first-order probabilistic models with combining rules. *AMAI* **54**(1–3), 223–256 (2008)
35. Poon, H., Domingos, P.: Joint inference in information extraction. In: *AAAI* (2007)
36. Quinlan, J.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Mateo (1993)
37. Richardson, M., Domingos, P.: Markov logic networks. *Mach. Learn.* **62**, 107–136 (2006)
38. Muggleton, S.: Inverse entailment and Progol. *New Gener. Comput.* **13**(3–4), 245–286 (1995)
39. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted Boltzmann machines for collaborative filtering. In: *ICML*, pp. 791–798 (2007)
40. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: *NIPS*, pp. 2951–2959 (2012)
41. Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. In: *NIPS* (2007)
42. Tieleman, T.: Training restricted Boltzmann machines using approximations to the likelihood gradient. In: *ICML*, pp. 1064–1071 (2008)
43. Wang, W., Cohen, W.: Learning first-order logic embeddings via matrix factorization. In: *IJCAI* (2016)



# Parallel Inductive Logic Programming System for Superlinear Speedup

Hiroyuki Nishiyama<sup>(✉)</sup> and Hayato Ohwada<sup>(✉)</sup>

Faculty of Science and Technology, Tokyo University of Science, 2641 Yamazaki,  
Noda-shi, Chiba 278-8510, Japan  
hiroyuki@rs.noda.tus.ac.jp, ohwada@rs.tus.ac.jp

**Abstract.** In this study, we improve our parallel inductive logic programming (ILP) system to enable superlinear speedup. This improvement redesigns several features of our ILP learning system and parallel mechanism. The redesigned ILP learning system searches and gathers all rules that have the same evaluation. The redesigned parallel mechanism adds a communication protocol for sharing the evaluation of the identified rules, thereby realizing superlinear speedup.

## 1 Introduction

Inductive logic programming (ILP) is a superior supervised learning tool. However, learning for large problems usually requires a considerable amount of time. In addition, to generate good rules, continuous learning must be performed while changing the parameters of ILP learning. It is thus necessary to shorten the ILP learning time. To solve this problem, various studies have focused on speeding up ILP using parallel methods [1–3, 7, 8, 12]. Although the problems were partially solved, the process speed was not sufficiently increased based on the number of processors provided, and the quality of the generated rules was not optimal, resulting in difficulty in using it as a practical tool. Skillicorn and Wang [10] succeeded in achieving superlinear speedup, but their method used only four or six CPUs, which performed poorly when the dataset was large.

In this study, we improved our parallel ILP system [7, 8] to enable superlinear speedup. We redesigned several features of our ILP learning system [5] and parallel mechanism [7]. The redesigned ILP learning system searches and gathers all rules that have the same evaluation. The redesigned parallel mechanism adds a communication protocol for sharing the evaluation of the identified rules. To estimate the speedup, we used dairy cattle data (e.g. hormones, feed, and activity) to determine successful conditions for artificial insemination [4]. When 30 CPUs were used to solve a large problem (147,992 s for one CPU), we achieved a speedup of 46.85 times (3,159 s). In addition, we applied the parallel ILP system to a very large problem (162,768 s for 10 CPUs). However, the problem could not be solved using one CPU, because it was too large. When 30 and 174 CPUs were used to solve this problem, we achieved a speedup of 3.13 times (51,968 s) and 17.73 times (9,183 s) respectively, based on 10 CPUs. Using our parallel ILP

system, we thus succeeded in superlinear speedup and demonstrated it to be more useful for large problems. In addition, we obtained identical rules using one CPU, 10 CPUs, 30 CPUs, and 174 CPUs.

## 2 Improved ILP System

An ILP system finds a hypothesis from a bounded hypothesis space. The ideal hypothesis covers as many positive examples and as few negative examples as possible. Let  $p(h)$  and  $n(h)$  be the numbers of positive and negative examples covered by hypothesis  $h$ . The number of literals in  $h$  is denoted by  $c(h)$ . We express this as follows, where  $g(h)$  indicates the generality of  $h$  and  $f(h)$  indicates the compression.

$$\begin{aligned} g(h) &= p(h) - c(h), \\ f(h) &= g(h) - n(h) \end{aligned}$$

In the ILP system [5], the compression measure  $f(h)$  is used to evaluate hypothesis  $h$ . If an evaluation (a value of  $f(h)$ ) is the best in a bounded hypothesis space, hypothesis  $h$  is the best hypothesis in the space. To avoid noise, the ILP uses two thresholds:  $pLimit$  and  $nLimit$ .

$$\text{minimize } f(h) \text{ subject to } p(h) \geq pLimit$$

$$\text{maximize } f(h) \text{ subject to } n(h) \leq nLimit$$

In addition, the ILP system used a simple set-covering algorithm like that of the typical ILP algorithm [6]:

- Step 1:** Find a hypothesis using the method mentioned above.
- Step 2:** Remove the positive examples covered by the hypothesis from the entire set of positive examples.
- Step 3:** Add the hypothesis found to the set of hypotheses being built (also known as rules), which is initially empty.
- Step 4:** Repeat step 1 to step 3 until the entire set of positive examples are covered by the rules.
- Step 5:** Return the rules.

There is a possibility that some hypotheses have the same evaluation, that is, the best value in a bounded hypothesis space in step 1. For example, a hypothesis  $h1$  was found and  $f(h1)$  was 20; that is, the best value in a bounded hypothesis space ( $p(h1) \geq pLimit$  and  $n(h1) \leq nLimit$ ). In this situation, if another hypothesis  $h2$  is found and  $f(h2)$  is 20, then the same evaluation is the best value. In addition,  $h2$  is not identical to  $h1$ . In this case, the ILP system usually keeps the hypothesis ( $h1$ ) that is found first (hypotheses after the second ( $h2$ ) are dropped). However, the hypotheses after the second ( $h2$ ) are potentially good hypotheses, because the evaluation is identical to the first ( $h1$ ). If the order of finding hypotheses is changed, the identified rules may be changed in the same problem, because we used a simple set-covering algorithm. This algorithm removes positive examples covered by a hypothesis, identified from the entire set

of positive examples (in step 2). Therefore, there is a possibility that covered positive examples may be different between when hypothesis  $h1$  is identified first and when hypothesis  $h2$  is identified first in the same hypothesis space.

Our new ILP system searches and gathers all hypotheses that have the same (best) evaluation. During the search, if our system finds a hypothesis with the same evaluation that is the best value, the system does not drop the hypothesis, but stores it. The system then finds some hypotheses in a bounded hypothesis space. Using this method, the identified rules are never changed in the same problem, even if the order of finding hypotheses is changed. In addition, there is a possibility that the system can reduce the learning time, because the system may remove more positive examples covered by hypotheses from the entire set of positive examples (in step 2), when some hypotheses are found in a bounded hypothesis space (in step 1). For example, we succeeded in reducing the learning time by approximately 10% using dairy cattle data (using a CPU).

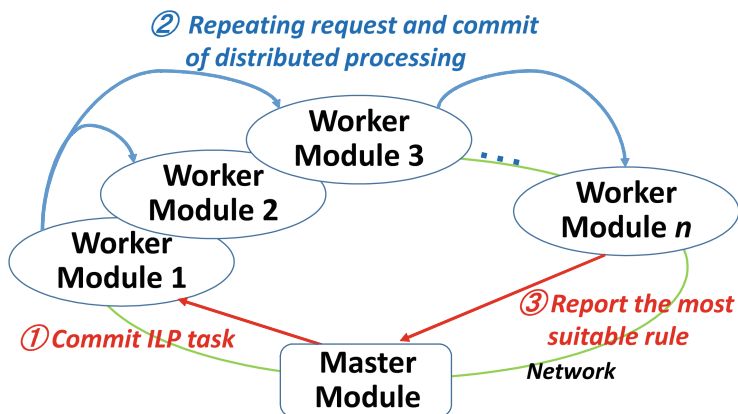
### 3 Improvement of Parallel ILP System

#### 3.1 Previous Parallel ILP System

Our previous research [7] designed and implemented a parallel-processing system for ILP. Figure 1 presents the system, which consists of a master module and worker modules (refer to [7] for details). In our system, the master module does not work for learning, but requests the first task to a worker module using the contract net negotiation protocol [11] and monitors all worker modules. The worker module itself has an autonomous function (unlike MapReduce). When a worker module has no task (e.g. immediately after starting up or completing a task), the worker module accepts a divided task from another worker module using the contract net negotiation protocol and starts the task. When the workload (the number of relationships in the information that the worker must search for) reaches a fixed quantity ( $divideNum$ ), the worker module requests other worker modules to process the divided task (using the contract net negotiation protocol). After the request for the first task issued by a master module is implemented for one worker module, autonomous process distribution begins among worker modules, and all existing worker modules are engaged (saturation of the task). Because the divided task continues to be repeated among worker modules, all worker modules finally complete all processing at approximately the same time. Depending on the size of the fixed quantity ( $divideNum$ ), it will take approximately several seconds. For example, when  $divideNum$  is defined as 200 in our implementation, the deviation was less than 1s, and the master module monitors and finds that all worker's tasks are finished, and receives the processing result (generated rules).

The flow of dividing a space among workers is as follows (step ② in Fig. 1).

- Workers use a branch-and-bound search and increase search *nodes*.
- If the  $nodes > divideNum$ , a worker requests a subtask (i.e., a part of the nodes: a part of the space) from others ( $divideNum$  is a threshold for dividing a task).



**Fig. 1.** System configuration of the ILP parallel-computation system and flow of distributed processing.

- A worker accepts the requested subtask if free (i.e., has no task).
- The requesting worker chooses and commits to the accepting worker that first sent the accepting message, and sends the subtask data. If the accepting worker does not have data or the background knowledge, then the requesting worker sends it with the subtask data.

This parallel-processing system has the following merits.

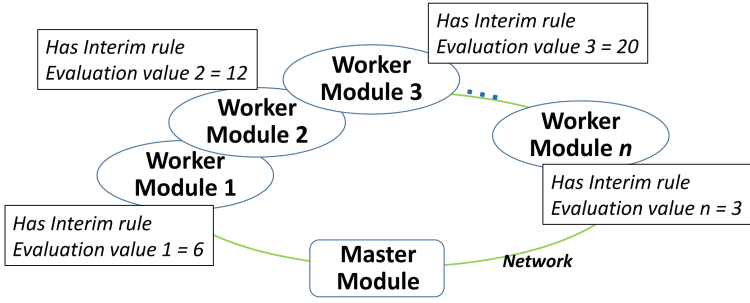
- 1. The master does not need to consider the division of the process in advance.
- 2. All workers work until the end (i.e. no free time) and finish at the same time.

The first merit indicates that the proposed system does not require the master to perform pre-division processing. The second merit means that the process speed can be increased by increasing the number of computers.

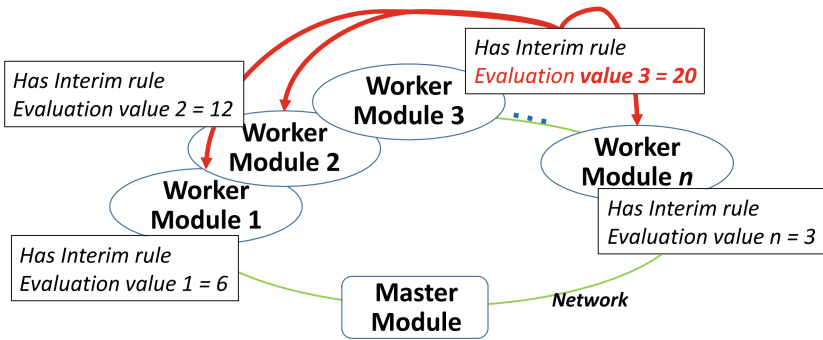
### 3.2 Improvement of the Parallel ILP System Communication Protocol

In the previous parallel ILP system, each worker module has the best evaluation of an identified hypothesis in its own module. Figure 2 illustrates one learning situation of the previous parallel ILP system. The value of the evaluation of the identified interim hypothesis (interim rule) of Worker Module 3 is 20. However, other worker modules have smaller values (i.e. the interim hypothesis is the best hypothesis that satisfies  $pLimit$  and  $nLimit$  in the searching process). In the branch-and-bound search, the other worker modules search for useless areas in the bounded hypothesis space. This system speed thus could not be sufficiently increased, even if all worker modules finished at the same time.

Our new parallel ILP system included a new communication protocol for sharing the best evaluation of the identified interim hypotheses (for only sharing

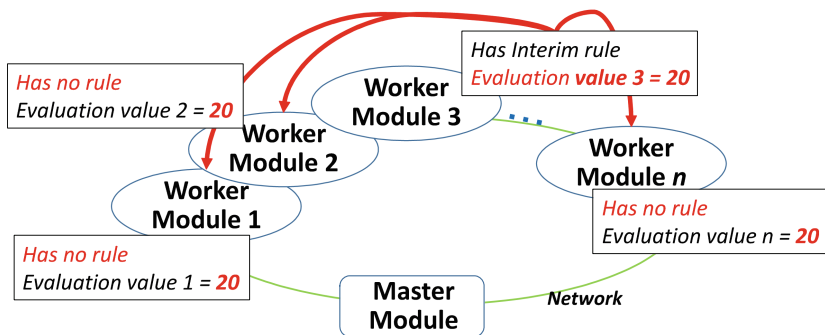


**Fig. 2.** One learning situation of the previous parallel ILP system. Each worker module has the best evaluation of an identified interim hypothesis.



**Fig. 3.** Worker Module 3 finds an interim hypothesis, and the evaluation is 20, the best evaluation of the module and others. Worker Module 3 then sends the value to all modules to share the best value.

the best evaluation value). In this protocol, when a worker module found an interim hypothesis with a better evaluation, then the worker sent the value to all other workers. Next, when another worker module received a value better than its own, then the worker updates its own value to the better one and drops its identified interim hypotheses. For example, Figs. 3 and 4 depict one learning situation of our new parallel ILP system. Worker Module 3 identifies an interim hypothesis and the evaluation is 20, the best evaluation of the module and others. Worker Module 3 then sends the value to all modules to share the best value (Fig. 3). Then, Worker Module 1, 2 and  $n$  receive the value that is better than their own, they update their own value to the better one and drops their own identified interim hypotheses (Fig. 4). This means that the new ILP system can avoid searching useless areas in the bounded hypothesis space. The new system can therefore speed up sufficiently, by increasing the number of computers used.



**Fig. 4.** When a worker module receives a value that is better than its own, its value is updated to the better value. Each worker module then shares the best evaluation, and all worker modules therefore have the same value.

## 4 Experiment and Results

We implemented the improved parallel ILP system using Java. A total of 30 computers (CPU Core i7 5820 K 6core/12thread 3.3 GHz 64 GB RAM) were used, as seen in Fig. 5, in an experiment to measure speedup using the implemented system. To estimate the speedup, we used data (e.g. hormone, feed, and activity) of dairy cattle to determine the successful conditions for artificial insemination [4].

### 4.1 Experiment Problems

We used two problems concerning dairy cattle to determine the successful conditions for artificial insemination [4] while estimating the speedup. The first was a large-scale problem that required approximately two days using one CPU. The second was a very large problem that required approximately two days using 10 CPUs. This problem could not be completed using one CPU, because it was too large.



**30 × CPU Core i7 5820K 6core/12thread 3.3GHz 64GB RAM**

**Fig. 5.** Experiment environment for the parallel ILP system. The Bio-oriented Technology Research Advancement Institution typically uses this environment for a daily cow project.



**Problem 1-Large Problem.** The first problem is a large one, consisting of the following learning data:

- Positive examples: 105
- Negative examples: 279
- Kinds of predicates: 6
- Background knowledge size: 49,757 lines

Table 1 lists the predicates and their mode declarations in the background knowledge of this problem. We also define several parameters.  $pLimit$  is 5,  $nLimit$  is 0, and max literals in learning is 8. After learning, we obtained 68 rules from this problem.

**Problem 2-Very-Large-Scale Problem.** The second problem is a very-large-scale one, consisting of the following learning data:

- Positive examples: 101
- Negative examples: 101

**Table 1.** Predicates and their mode declarations in the background knowledge of the large problem. Mode + indicates an input variable, - an output variable, and # a constant.

---

Predicates
progesterone(+cowID, +-preg_datetime, #val),
progesterone_diff(+cowID, +-preg_datetime, +-prog_datetime, #time, #val),
feed(+cowID, +-feed_datetime, #val),
feed_diff(+cowID, +-feed_datetime, +-feed_datetime, #time, #val),
sametime(+cowID, +-prog_datetime, +-feed_datetime),
birth_num(+cowID, #birth_number)

---

**Table 2.** Predicates and their mode declarations in the background knowledge of the very-large-scale problem. Mode + indicates an input variable, - an output variable, and # a constant.

---

Predicates
progesterone(+cowID, +-preg_datetime, #val),
progesterone_diff(+cowID, +-preg_datetime, +-prog_datetime, #time, #val),
preg_datetime_definition(+cowID, +-preg_datetime, #val),
feed(+cowID, +-feed_datetime, #val),
feed_diff(+cowID, +-feed_datetime, +-feed_datetime, #time, #val),
feed_datetime_definition(+cowID, +-feed_datetime, #val),
birth_num(+cowID, #birth_number),
activity(+cowID, +-activity_datetime, #val),
activity_diff(+cowID, +-activity_datetime, +-activity_datetime, #time, #val),
activity_datetime_definition(+cowID, +-activity_datetime, #val)

---

- Kinds of predicates: 10
- Background knowledge size: 48,049 lines

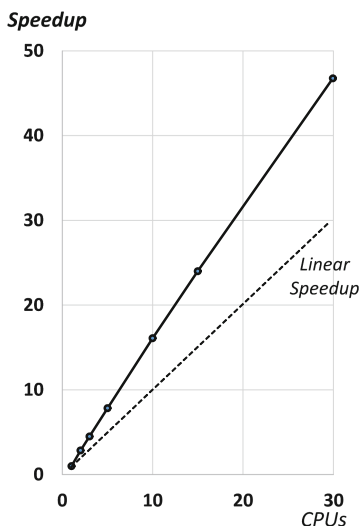
Table 2 lists the predicates and their mode declarations in the background knowledge of this problem. In addition, we define several parameters.  $pLimit$  is 10,  $nLimit$  is 5, and max literals in learning is 8. After learning, we obtained 168 rules from this problem.

## 4.2 Experiment Results

**Results of Problem 1.** Table 3 and Fig. 6 present the results of problem 1, demonstrating that the parallel ILP system accomplished superlinear speedup. In addition, we obtained identical rules using 1 to 30 CPUs.

**Table 3.** Results of parallel experiments of problem 1.

The Number of CPUs	Execution time (sec.)	Speedup
1	147,992	1.000
2	52,123	2.839
3	32,849	4.505
5	18,859	7.848
10	9,183	16.116
15	6,157	24.036
30	3,159	46.848



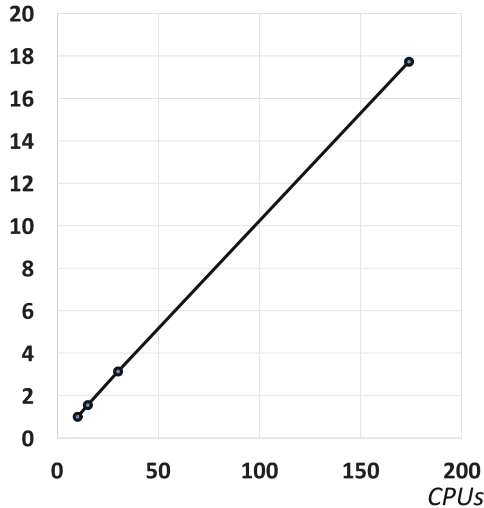
**Fig. 6.** Graph of parallel experiments of problem 1. The solid line is the result of our parallel ILP system, and the dashed line shows a linear speedup. This graph thus demonstrates that our system accomplished superlinear speedup.

**Results of Problem 2.** Table 4 and Fig. 7 present the results of problem 2. Our parallel ILP system accomplished a slight superlinear speedup based on 10 CPUs. This means we can apply our system to larger-scale problems by using more CPUs. In addition, we obtained identical rules using 10 to 174 CPUs. The 174 CPUs represent 29 computers with six CPUs each. One computer serves as the master module.

**Table 4.** Results of parallel experiments of problem 2. The 174 CPUs represent 29 computers six CPUs each. One computer is the master module.

The Number of CPUs	Execution time (sec.)	Speedup (base: 10 CPUs)
10	162,768	1.000
15	105,107	1.549
30	51,968	3.132
174	9,183	17.725

**Speedup (Base: 10 CPUs)**



**Fig. 7.** Graph of parallel experiments of problem 2. The results indicate that our system can be applied to very large problems by using more CPUs.

## 5 Discussion

### 5.1 Discussion of Speedup

Our original purpose was to attain linear speedup. However, we succeeded in attaining superlinear speedup and demonstrated it to be more useful in large

problems using our parallel ILP system. This demonstrates that all worker modules can divide a learning task and work ideally without searching useless areas in the bounded hypothesis space. In addition, we succeeded in obtaining identical rules by using one to 174 CPUs. This indicates that the rules identified are unchanged in a given problem, even if the order of finding hypotheses is changed.

Figure 8 provides the reason for the speedup of our parallel ILP system. The left side of Fig. 8 depicts the usual parallel method for ILP. This method only divides a space as subtasks among workers. Each worker searches its space only as in Fig. 8. Here, worker C worked much more than other workers. Thus, it is not effective. The right side of Fig. 8 depicts our parallel method, where each worker helps other workers when it completes searching its own space. Using our method, the workers communicate with each other and request or accept a portion of the subtasks. Finally, each worker simultaneously completes all tasks. In addition, our parallel mechanism adds a communication protocol for sharing the evaluation of rules identified. This means that our system can avoid searching useless areas in the bounded hypothesis space. Our system can therefore be speed up sufficiently by increasing the number of computers used.

### 5.2 Discussion of Speedup Stability

We checked the stability and robustness of our parallel ILP system in another experimental environment, using 48 computers (2vCPU:2.4 GH, 8GBRAM) on an ATLUS Cloud with 95 CPUs for workers and 1 CPU for the master module. In this experiment, we repeatedly (16 times) executed parallel learning using our system on problem 2. Table 5 presents the results of this experiment. The average time is 20,741.69s, but the standard deviation execution is only 29.16s. This means that our system is very stable for parallel ILP execution.

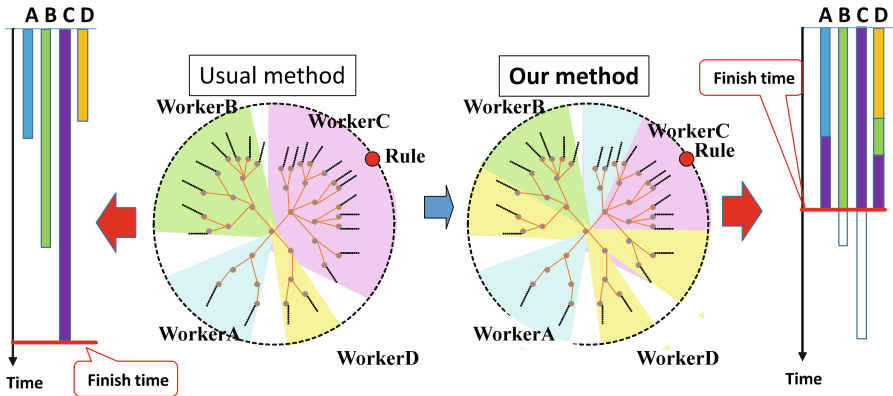


Fig. 8. Comparison between the usual method and our method. The left side depicts the usual method, and the right side depicts our method. Using our method, each worker completes all tasks at the same time.

**Table 5.** Result of repeated application of the parallel ILP system to problem 2. The result indicates that the speedup is stable.

Number of times	Execution time (sec.)
1	20773
2	20734
3	20747
4	20695
5	20758
6	20774
7	20787
8	20711
9	20755
10	20733
11	20705
12	20736
13	20715
14	20717
15	20738
16	20789
Average	20741.69
Standard deviation	29.16

## 6 Conclusions

In this study, we improved our parallel ILP system to enable superlinear speedup. This improvement redesigned several features of our ILP learning system and parallel mechanism. The redesigned ILP learning system searches and gathers all rules that have the same evaluation. The redesigned parallel mechanism included a communication protocol for sharing the evaluations of identified rules. To estimate the speedup, we used data from dairy cattle indicating the successful conditions for artificial insemination. Finally, we succeeded in superlinear speedup and demonstrated that it was more useful in large problems. We succeeded in obtaining identical rules using one to 174 CPUs. In addition, we applied our system to the problem of anomaly detection based on the log analysis of a company's server, and succeeded in obtaining valid rules in a realistic time [9]. Currently, we are designing an intelligent grid-search system using this parallel ILP to acquire better rules for dairy-cattle problems.

**Acknowledgments.** This research was supported by grants from the Project of the Bio-oriented Technology Research Advancement Institution, NARO (the special scheme project on advanced research and development for next-generation technology).

## References

1. Fidjeland, A., Luk, W., Muggleton, S.: Customisable multi-processor acceleration of inductive logic programming. In: Latest Advances in Inductive Logic Programming, pp. 123–141 (2014)
2. Fonseca, N.A., Silva, F.M.A., Camacho, R.: Strategies to parallelize ILP Systems. In: ILP 2005, pp. 136–153 (2005)
3. Katzouris, N., Artikis, A., Paliouras, G.: Distributed Online Learning of Event Definitions. Computing Research Repository (CoRR), abs/1705.02175 (2017)
4. Matsumoto, A., Kubota, C., Ohwada, H.: Extracting rules for successful conditions for artificial insemination in dairy cattle using inductive logic programming. In: Proceedings of the 9th International Conference on Machine Learning and Computing, pp. 6–10 (2017)
5. Mizoguchi, F., Ohwada, H.: Constrained relative least general generalization for inducing constraint logic programs. *New Gener. Comput.* **13**, 335–368 (1995)
6. Muggleton, S.: Inverse entailment and progol. *New Gener. Comput.* **13**(3, 4), 245–286 (1995)
7. Nishiyama, H., Ohwada, H.: Yet another parallel hypothesis search for inverse entailment. In: CEUR Workshop Proceedings of the Late Breaking Papers of the 25th International Conference on ILP, vol. 1636, pp. 86–94 (2016)
8. Ohwada, H., Nishiyama, H., Mizoguchi, F.: Concurrent execution of optimal hypothesis search for inverse entailment. In: Cussens, J., Frisch, A. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 165–173. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-44960-4\\_10](https://doi.org/10.1007/3-540-44960-4_10)
9. Shimada, T., Hatano, R., Nishiyama, H.: Server failure detection system based on inductive logic programming and random forest. In: Proceedings of 33rd International Conference on Computers and Their Applications, CATA 2018, March 2018
10. Skillicorn, D.B., Wang, Y.: Parallel and sequential algorithms for data mining using inductive logic. *Knowl. Inf. Syst.* **3**(4), 405–421 (2001)
11. Smith, R.G.: The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Trans. Comput.* **C-29**(12), 1104–1113 (1980)
12. Fonseca, N.A., Srinivasan, A., Silva, F., Camacho, R.: Parallel ILP for distributed-memory architectures. *Mach. Learn. J.* **74**(3), 257–279 (2009)



# Inductive Learning from State Transitions over Continuous Domains

Tony Ribeiro<sup>1</sup>(✉), Sophie Turret<sup>2</sup>, Maxime Folschette<sup>3</sup>, Morgan Magnin<sup>1</sup>,  
Domenico Borzacchiello<sup>5</sup>, Francisco Chinesta<sup>6</sup>, Olivier Roux<sup>1</sup>,  
and Katsumi Inoue<sup>4</sup>

<sup>1</sup> Laboratoire des Sciences du Numérique de Nantes (LS2N),  
1 rue de la Noë, 44321 Nantes, France  
[tony\\_ribeiro@ircyn.ec-nantes.fr](mailto:tony_ribeiro@ircyn.ec-nantes.fr)

<sup>2</sup> Max-Planck-Institut für Informatik, Saarland Informatics Campus,  
66123 Saarbrücken, Germany

<sup>3</sup> Univ Rennes, Inria, CNRS, IRISA, IRSET, 35000 Rennes, France

<sup>4</sup> National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku,  
Tokyo 101-8430, Japan

<sup>5</sup> Institut de Calcul Intensif, 1 rue de la Noë, 44321 Nantes, France

<sup>6</sup> PIMM, ENSAM ParisTech, 151 Boulevard de l'Hopital, 75013 Paris, France

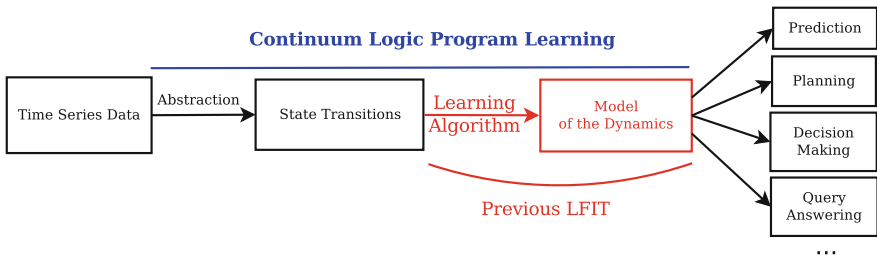
**Abstract.** Learning from interpretation transition (*LFIT*) automatically constructs a model of the dynamics of a system from the observation of its state transitions. So far, the systems that *LFIT* handles are restricted to discrete variables or suppose a discretization of continuous data. However, when working with real data, the discretization choices are critical for the quality of the model learned by *LFIT*. In this paper, we focus on a method that learns the dynamics of the system directly from continuous time-series data. For this purpose, we propose a modeling of continuous dynamics by logic programs composed of rules whose conditions and conclusions represent continuums of values.

**Keywords:** Continuous logic programming  
Learning from interpretation transition · Dynamical systems  
Inductive logic programming

## 1 Introduction

Learning the dynamics of systems with many interactive components becomes more and more important due to many applications, e.g., multi-agent systems, robotics and bioinformatics. Knowledge of system dynamics can be used by agents and robots for planning and scheduling. In bioinformatics, learning the dynamics of biological systems can correspond to the identification of the influence of genes and can help to understand their interactions. Dynamic system modeling based on time-series data can be classified into discrete and continuous approaches. Discrete and logic-based modeling methodologies assume that

the temporal evolution of each entity or variable describing the system occurs in synchronous discrete time steps. These methods seek to infer the regulation functions that update the state of each variable based on the states at previous time steps. In contrast with this approach, continuous models are defined by differential equations in which the rate of change of a given variable is related to the actual state of the system. Continuous approaches do not need the discretization of the real-valued measurement data. As a consequence, using real-valued parameters over a continuous timescale yields more reliable results, at least in theory, because it does not introduce any discretization related error. A review of both approaches, outlining their advantages and limitations, with specific applications to gene regulatory networks, is found in [8]. In this paper, we propose a logic-based modeling, which like continuous approaches, allows to deal with real-valued measured data. It is however assuming discrete time steps.



**Fig. 1. Existing work:** assumes a discretization of the time-series data used as input to LFIT. **New method:** no abstraction of the time-series data.

Learning from interpretation transition (*LFIT*) [7] has been proposed to automatically construct a model of the dynamics of a system from the observation of its state transitions. Figure 1 shows this learning process. Given some raw data, like time-series data of gene expression, a discretization of those data in the form of state transitions is assumed. From those state transitions, according to the semantics of the system dynamics, different inference algorithms that model the system as a logic program are proposed. The semantics of system dynamics can differ regarding the synchronism of its variables, the determinism of its evolution and the influence of its history. The LFIT framework proposes several modeling and learning algorithms to tackle those different semantics. So far the following systems are tackled: memory-less synchronous consistent systems [7], systems with memory [13], non-consistent systems [10].

So far, the discretization of the raw data was assumed given. Its quality is critical to obtain good quality models out of LFIT. For example when learning gene expression evolution, it means that the gene expression levels must be known beforehand. This information is usually unknown and statistical methods are used to guess it. Those methods rely most of the time on static state analysis to provide a division of the gene expressions into meaningful intervals [6,9].



But the levels of expressions and the dynamics are inseparable in biological models. Sometimes those levels of expressions are in fact what must be learned.

In this paper, we learn both at the same time (see Fig. 1). For this purpose, we extend LFIT to handle variables representing intervals instead of singletons. In the case of learning gene expression levels, a method that builds dynamical Bayesian networks has been proposed in [11], but it provides no theoretical guarantees. In contrast, our approach, that computes continuous logic programs, is generic and comes with soundness and completeness results. Time series data are considered in [14] but continuous data are statically discretized into interval predicate representing statistical properties and their goal is classification. The best-known logic handling intervals is temporal logic [4], which is solely concerned with temporal intervals. In [3] learning of temporal interval relationship defined by [1] (like meet or overlap) is considered. We rely on some of those relationships for rule comparison but learning those relations is not our concern. Temporal logic aside, the other related techniques focus on applying continuous functions on them [12]. The closest to our approach is interval constraint programming [2], but it handles static equational systems. All these techniques are thus unsuitable to solve the problem considered here, where the time is discrete, the values continuous and no continuous function is required.

The organization of the paper is as follows. Section 2 provides a formalization of continuum logic programs, the learning operations and their properties. Section 3 presents the ACEDIA learning algorithm and its experimental evaluation.

## 2 Continuum Logic and Program Learning

In this section, the concepts necessary to understand the learning algorithm are formalized. In Sect. 2.1 the basic notions of *continuum logic* (CL) and a number of important properties that the learned programs must have are presented. Then in Sect. 2.2 the operations that are performed during the learning, as well as results about the preservation of the properties introduced in Sect. 2.1 throughout the learning are exposed.

### 2.1 Continuum Logic Programs

Let  $\mathcal{V} = \{v_1, \dots, v_n\}$  be a finite set of  $n$  variables and  $\mathcal{I}_{\mathbb{R}}$  be the set of all intervals in  $\mathbb{R}$ . We use basic interval arithmetic operations such as intersection, hull and avoid. Formally for  $I_1, I_2 \in \mathcal{I}_{\mathbb{R}}$ ,  $I_1 \cap I_2 = \{x \in \mathbb{R} \mid x \in I_1 \wedge x \in I_2\}$ ,

$$\text{hull}(I_1, I_2) = \begin{cases} I_1 & \text{if } I_2 = \emptyset, \\ I_2 & \text{if } I_1 = \emptyset, \\ \{x \in \mathbb{R} \mid \exists y \in I_1, \exists z \in I_2, y \leq x \leq z \vee z \leq x \leq y\} & \text{otherwise.} \end{cases}$$

and  $\text{avoid}(I_1, I_2) = \{\{x \in I_1 \mid \forall x' \in I_2, x < x'\}, \{x \in I_1 \mid \forall x' \in I_2, x > x'\}\}$ .

The atoms of  $\mathcal{CL}$  are of the form  $v^I$  where  $v \in \mathcal{V}$  and  $I \in \mathcal{I}_{\mathbb{R}}$ . An atom  $v^I$  is *unit* when  $I = \{x\}$  and *empty* when  $I = \emptyset$ . A  $\mathcal{CL}$  rule is defined by:

$$R = v^I \leftarrow v_1^{I_1} \wedge \dots \wedge v_n^{I_n} \quad (1)$$

where  $v^I$  and  $v_i^{I_i}$  for  $1 \leq i \leq n$  are atoms in  $\mathcal{CL}$ . The atom on the left-hand side of the arrow is called the *head* of  $R$  and is denoted  $h(R)$ . The notation  $v_{h(R)}$  denotes the variable that occurs in  $h(R)$ . The conjunction on the right-hand side of the arrow is called the *body* of  $R$ , written  $b(R)$ . The conjunction  $b(R)$ , that contains a single occurrence of each variable in  $\mathcal{V}$ , is assimilated to the set  $\{v_1^{I_1}, \dots, v_n^{I_n}\}$  and we use set operations such as  $\in$  and  $\cap$  on it. A *continuum logic program* (CLP) is a set of  $\mathcal{CL}$  rules. Intuitively, the rule  $R$  has the following meaning: the variable  $v$  takes a value in  $I$  at the next step if each variable  $v_i$  takes a value in  $I_i$  at the current step.

The two following definitions introduce relations between atoms and between rules that are used further along.

**Definition 1 (Relations between atoms).** *Two atoms  $a = v^I$  and  $a' = v^{I'}$  that are based on the same variable  $v \in \mathcal{V}$  can have the following relationships with each other:*

- $a$  and  $a'$  overlap when  $I \cap I' \neq \emptyset$ , written  $a \sqcap a'$ ,
- $a$  subsumes  $a'$  when  $I' \subseteq I$ , written  $a' \sqsubseteq a$ .

*In the last case, we also write that  $a$  is more general than  $a'$  (resp.  $a'$  is more specific than  $a$ ). The notion of subsumption is straightforwardly extended to conjunctions of atoms  $B_1$  and  $B_2$  in the following way.  $B_1$  subsumes  $B_2$ , written  $B_2 \sqsubseteq B_1$  iff:*

$$\forall a \in B_1, \exists a' \in B_2, \text{ such that } a' \sqsubseteq a.$$

**Definition 2 (Rules Domination).** *Let  $R_1, R_2$  be two  $\mathcal{CL}$  rules. The rule  $R_1$  dominates  $R_2$ , written  $R_2 \leq R_1$  if  $h(R_1) \sqsubseteq h(R_2)$  and  $b(R_2) \sqsubseteq b(R_1)$ .*

**Proposition 1.** *Let  $R_1, R_2$  be two  $\mathcal{CL}$  rules. If  $R_1 \leq R_2$  and  $R_2 \leq R_1$  then  $R_1 = R_2$ .*

*Proof.* Let  $R_1, R_2$  be two  $\mathcal{CL}$  rules such that  $R_1 \leq R_2$  and  $R_2 \leq R_1$ . Then  $h(R_1) \sqsubseteq h(R_2)$  and  $h(R_2) \sqsubseteq h(R_1)$ , hence  $h(R_1) = v^{I_1}$  and  $h(R_2) = v^{I_2}$  and  $I_1 \subseteq I_2 \subseteq I_1$  thus  $I_1 = I_2$  and  $h(R_1) = h(R_2)$ . The same reasoning is applied on each variable to conclude  $b(R_1) = b(R_2)$ .  $\square$

Rules with more specific heads and more general bodies dominate the other rules. In practice, these are the rules we are interested in since they cover the most general cases and give the most accurate results.

The dynamical system that we want to learn the rules of is represented by a succession of *continuum states* as formally defined below.

**Definition 3 (Continuum State).** *A continuum state  $s$  is a function from  $\mathcal{V}$  to  $\mathbb{R}$ , i.e. it associates a real value to each variable in  $\mathcal{V}$ . It represents the set of unit atoms  $\{v_1^{\{x_1\}}, \dots, v_n^{\{x_n\}}\}$ . We write  $\mathcal{S}$  to denote the set of all continuum states and a pair of states  $(s, s') \in \mathcal{S}^2$  is called a transition.*

The following definitions and propositions describe the interactions between states and rules.

**Definition 4 (Rule-states matching).** *Let  $s \in \mathcal{S}$ . The CL rule  $R$  matches  $s$ , written  $R \sqcap s$ , if  $\forall a \in b(R), \exists a' \in s$  such that  $a \sqcap a'$ .*

A rule is activated only when it matches the current state of the system.

**Definition 5 (Cross-matching).** *Let  $R$  and  $R'$  be two CL rules. These rules cross-match, written  $R \sqcap R'$  when there exists  $s \in \mathcal{S}$  such that  $R \sqcap s$  and  $R' \sqcap s$ .*

Cross-matching can also be defined without the use of a matching state.

**Proposition 2 (Cross-matching).** *Let  $R$  and  $R'$  be two CL rules.*

$$R \sqcap R' \text{ iff } \forall (v^I, v^{I'}) \in b(R) \times b(R'), v^I \sqcap v^{I'}.$$

Proof. For the direct implication, assume given two CL rules  $R$  and  $R'$  such that  $R \sqcap R'$ . By definition, there exists  $s \in \mathcal{S}$  such that  $s$  matches both  $R$  and  $R'$ . Also by definition, for all  $(v^I, v^{I'}) \in b(R) \times b(R')$ , there exists  $a, a' \in s$  such that  $v^I \sqcap a$  and  $v^{I'} \sqcap a'$ . Moreover, by the definition of a state,  $a = a' = v^{\{x\}}$ . Thus  $x \in (I \cap I')$ , hence  $v^I \sqcap v^{I'}$ .

For the converse implication, it suffices to construct a suitable  $s \in \mathcal{S}$ , which can be done in the following way: For all  $v \in \mathcal{V}$ , there exists  $x_v \in I \cap I'$ , where  $v^I \in b(R)$  and  $v^{I'} \in b(R')$  since by Definition 1,  $I \cap I' \neq \emptyset$ . The state  $s = \{x_v \mid v \in \mathcal{V}\}$  is such that  $s \sqcap R$  and  $s \sqcap R'$ .  $\square$

The final program we want to learn should be complete and consistent within itself and with the observed transitions. The following definitions formalize these desired properties.

**Definition 6 (Rule and program realization).** *Let  $R$  be a CL rule and  $(s, s') \in \mathcal{S}^2$ . The rule  $R$  realizes the transition  $(s, s')$ , written  $s \xrightarrow{R} s'$ , if  $R \sqcap s$  and there exists  $a \in s'$  such that  $a \sqsubseteq h(R)$ . It realizes a set of transitions  $T \subseteq \mathcal{S}^2$ , written  $\xrightarrow{R} T$  if for all  $(s, s') \in T, s \xrightarrow{R} s'$ .*

A CLP  $P$  realizes  $(s, s')$ , written  $s \xrightarrow{P} s'$ , if for all  $v \in \mathcal{V}$ , there exists  $R \in P, s \xrightarrow{R} s'$ . It realizes  $T$ , written  $\xrightarrow{P} T$  if for all  $(s, s') \in T$  and all  $v \in \mathcal{V}$ , there exists  $R \in P$ , such that  $v_{h(R)} = v$  and  $s \xrightarrow{R} s'$ .

**Definition 7 (Conflicts).** Conflicts can occur between a CL rule  $R$  and a state  $(s, s') \in \mathcal{S}^2$  or between two CL rules  $R$  and  $R'$ . The first kind of conflict is when  $R \sqcap s$  and  $s \xrightarrow{R} s'$  and the second when  $v_{h(R)} = v_{h(R')}$ ,  $R \sqcap R'$  and neither  $h(R) \sqsubseteq h(R')$  or  $h(R') \sqsubseteq h(R)$ .

**Definition 8 (Consistent program).** A CLP  $P$  is strongly consistent if it does not contain conflicting rules, i.e. for all  $R, R' \in P$  such that  $v_{h(R)} = v_{h(R')}$  and  $R \sqcap R'$ , either  $h(R) \sqsubseteq h(R')$  or  $h(R') \sqsubseteq h(R)$ . It is consistent when for all conflicting  $R, R' \in P$ , the rule  $R'' = v_{h(R)}^\emptyset \leftarrow \{v^{I''} \mid v \in \mathcal{V}, v^I \in b(R), v^{I'} \in b(R') \text{ and } I \cap I' \subseteq I''\}$  belongs to  $P$ . Otherwise  $P$  is said to be non-consistent.

Note that in the definition of a consistent CLP, due to the conflict between  $R$  and  $R'$ ,  $I \cap I'$  is never empty. In case there is a blind spot in the observed transitions close to a frontier in the behavior of the system (which always happens to some degree due to the continuous nature of the rules and the discrete nature of the observed transitions) the rules with empty heads indicate the uncertainty of the behavior between the two closest observations.

**Definition 9 (Complete program).** *A CLP  $P$  is complete if for all  $s \in \mathcal{S}$  and all  $v \in \mathcal{V}$  there exists  $R \in P$  such that  $R \sqcap s$  and  $v_{h(R)} = v$ .*

*Example 1.* Let  $\mathcal{V} = \{v_1, v_2\}$  and consider the two rules  $R_1 = v_1^{[5;8]} \leftarrow v_1^{-\infty; \infty[ \wedge v_2^{]0;5[}$  and  $R_2 = v_1^{\{7\}} \leftarrow v_1^{-\infty; \infty[ \wedge v_2^{[4;9]}$ . The rules  $R_1$  and  $R_2$  cross-match but they do not conflict since  $v_{h(R_2)} \sqsubseteq v_{h(R_1)}$  and they do not dominate each other since  $b(R_1) \not\sqsubseteq b(R_2)$  and  $b(R_2) \not\sqsubseteq b(R_1)$ . They both realize the transition  $t = ((10; 4.5), (7; 1))$ , however the program  $P = \{R_1, R_2\}$  does not realize  $t$  because it contains no rule with  $v_2$  as its head variable.  $P$  is also not complete, while the CLP  $P' = \{v_1^{[1;2]} \leftarrow v_1^{-\infty, \infty[ \wedge v_2^{-\infty, \infty[}, v_2^\emptyset \leftarrow v_1^{-\infty, \infty[ \wedge v_2^{-\infty, \infty[}\}$  is complete.

## 2.2 Learning Operations

The three following definitions describe formally the main operation performed by the learning algorithm, which is to adapt a CLP to realize a new transition with a minimal amount of changes in the dynamics of the program.

**Definition 10 (Rule least specialization).** *Let  $R$  be a CL rule and  $(s, s') \in \mathcal{S}^2$  such that  $R$  and  $(s, s')$  are conflicting. The least specialization of  $R$  by  $(s, s')$  is:*

$$P_{\text{spe}}(R, (s, s')) = \bigcup_{v_s^{\{x\}} \in s} \{h(R) \leftarrow (\{v_s^{I'}\} \cup b(R) \setminus \{v_s^{I_s}\})\},$$

where  $v_s^{I_s} \in b(R)$ ,  $I' \in \text{avoid}(I_s, \{x\})$ .

**Definition 11 (Rule least generalization).** *Let  $R$  be a CL rule and  $(s, s') \in \mathcal{S}^2$  such that  $R$  and  $(s, s')$  are conflicting. The least generalization of  $R$  by  $(s, s')$  is:*

$$P_{\text{gen}}(R, (s, s')) = \{v^{I''} \leftarrow b(R) \mid h(R) = v^I, v^{\{x\}} \in s', I'' = \text{hull}(I, \{x\})\}$$

Note that  $P_{\text{gen}}(R, (s, s'))$  contains a single rule while the number of rules in  $P_{\text{spe}}(R, (s, s'))$  depends on the relationship between the variables in  $b(R)$  and in  $s$ .

**Definition 12 (Rule least revision).** *Let  $R$  be a CL rule and  $t \in \mathcal{S}^2$ . The least revision of  $R$  by  $t$  is:*

$$P_{\text{rev}}(R, t) = \begin{cases} P_{\text{spe}}(R, t) \cup P_{\text{gen}}(R, (s, s')) & \text{when } R \text{ and } t \text{ are conflicting} \\ \{R\} & \text{otherwise.} \end{cases}$$

The least revision of a CLP  $P$  by a transition  $t \in \mathcal{S}^2$  is  $P_{\text{rev}}(P, t) = \bigcup_{R \in P} P_{\text{rev}}(R, t)$ .

The intuition behind the least revision is that when a rule is conflicting with a considered transition it is for two possible reasons. Either the conclusion of the rule is correct but the conditions are too general, or the conditions of the rule are correct but the conclusion is too specific.

The following theorem collects properties of the least revision that make it suitable to be used in the learning algorithm.

**Theorem 1.** *Let  $R$  be a CL rule and  $(s, s') \in \mathcal{S}^2$ . Assume  $R$  and  $(s, s')$  are conflicting, and let  $S_R = \{s'' \in \mathcal{S} \mid R \sqcap s''\}$  and  $S_{\text{spe}} = \{s'' \in \mathcal{S} \mid \exists R' \in P_{\text{spe}}(R, (s, s')), R' \sqcap s''\}$ . The following results hold:*

1.  $S_{\text{spe}} = S_R \setminus \{s\}$ ,
2.  $s \xrightarrow{P_{\text{gen}}(R, (s, s'))} s'$ ,
3.  $P_{\text{rev}}(R, (s, s'))$  is strongly consistent and contains no rule conflicting with  $R$  and  $(s, s')$ .

Proof.

1. First, let  $s'' \in S_R \setminus \{s\}$ . Then there exists  $v^{\{x''\}} \in s''$ , such that  $v^{\{x\}} \in s$ ,  $v^I \in b(R)$ ,  $x' \in I$  and  $x \neq x'$ . Since  $x \in I$  because  $R$  and  $(s, s')$  conflict with each other, we can assume that  $x' < x$  (the proof in the case  $x' > x$  is symmetrical). Thus, the rule  $R' = h(R) \leftarrow b(R) \setminus \{v^I\} \cup \{v^{\{x'' \in I \mid x'' < x\}}\}$  is such that  $R' \sqcap s''$  and  $R' \in P_{\text{spe}}(R, (s, s'))$  hence  $s'' \in S_{\text{spe}}$ .  
Now consider  $s'' \in S_{\text{spe}}$ . By the definition of  $S_{\text{spe}}$  there exists  $R' \in P_{\text{spe}}(R, (s, s'))$  such that  $R' \sqcap s''$  thus there exists  $I^{\{x\}} \in s''$  such that  $v^I \in b(R)$ ,  $v^{\{x\}} \in s$  and  $x' \in I$ ,  $x' \neq x$ . Hence  $R \sqcap s''$  but  $s'' \neq s$  thus  $s'' \in S_R \setminus \{s\}$ .
2. Let  $P_{\text{gen}}(R, (s, s')) = \{R'\}$ . Since  $(s, s')$  and  $R$  are conflicting,  $R \sqcap s$ . Moreover given  $h(R') = v^I$ , by the definition of  $P_{\text{gen}}$  and the hull function, there exists  $v^{\{x\}} \in s'$  such that  $x \in I = \text{hull}(I, \{x\})$ , hence  $s \xrightarrow{R'} s'$ .
3. Let  $R_1, R_2 \in P_{\text{spe}}(R, (s, s'))$ . By the definition of  $P_{\text{spe}}(R, (s, s'))$ ,  $h(R_1) = h(R_2)$ , thus  $R_1$  and  $R_2$  cannot conflict. Now let  $R_3 \in P_{\text{gen}}(R, (s, s'))$ . Again  $R_1$  and  $R_3$  cannot conflict because  $h(R_1) \sqsubseteq h(R_3)$ . Thus  $P_{\text{rev}}(R, R')$  is free of conflicts. In addition, for all  $R' \in P_{\text{spe}}(R, (s, s'))$ ,  $h(R') = h(R)$  and for  $\{R'\} = P_{\text{gen}}(R, (s, s'))$ ,  $h(R) \sqsubseteq h(R')$  by the definition of the hull function. Finally,  $P_{\text{rev}}(R, (s, s'))$  does not conflict with  $(s, s')$  due to the two previous points of this theorem.

□

*Example 2.* Let  $\mathcal{V} = \{v_1, v_2\}$ ,  $R = v_1^{\{5\}} \leftarrow v_1^{]-\infty, \infty[} \wedge v_2^{]-\infty, 8[}$  and  $t = ((0; 1), (4, 2))$ . Then  $P_{\text{rev}}(R, t) = \{v_1^{[4; 5]} \leftarrow v_1^{]-\infty, \infty[} \wedge v_2^{]-\infty; 8[}, v_1^{\{5\}} \leftarrow v_1^{]-\infty; 0[} \wedge v_2^{]-\infty; 8[}, v_1^{\{5\}} \leftarrow v_1^{]0; \infty[} \wedge v_2^{]-\infty; 8[}, v_1^{\{5\}} \leftarrow v_1^{]-\infty, \infty[} \wedge v_2^{]-\infty, 1[}, v_1^{\{5\}} \leftarrow v_1^{]-\infty, \infty[} \wedge v_2^{]1, 8[}\}$ . The first rule in  $P_{\text{rev}}(R, t)$  is the least generalization of  $R$ .

The following definition groups all the properties that we want the learned program to have.

**Definition 13. (Suitable and optimal program).** Let  $T \subseteq \mathcal{S}^2$ . A CLP  $P$  is suitable for  $T$  when:

- $P$  is consistent,
- $P$  is complete,
- $P$  realizes  $T$ ,
- there is no rule with empty head in  $P$  that matches a  $s$  such that  $(s, s') \in T$ ,
- for all CL rules  $R$  not conflicting with a  $s$  such that  $(s, s') \in T$ , there exists  $R' \in P$  such that  $R \leq R'$ .

If in addition, for all  $R \in P$ , all the CL rules  $R'$  belonging to CLP suitable for  $T$  are such that  $R \leq R'$  implies  $R' \leq R$  then  $P$  is called optimal.

**Proposition 3.** Let  $T \subseteq \mathcal{S}^2$ . The CLP optimal for  $T$  is unique and denoted  $P_{\mathcal{O}}(T)$ .

Proof. Let  $T \subseteq \mathcal{S}^2$ . Assume the existence of two distinct CLPs optimal for  $T$ , denoted by  $P_{\mathcal{O}_1}(T)$  and  $P_{\mathcal{O}_2}(T)$  respectively. Then w.l.o.g. we consider that there exists a CL rule  $R$  such that  $R \in P_{\mathcal{O}_1}(T)$  and  $R \notin P_{\mathcal{O}_2}(T)$ . If  $R$  is conflicting with  $T$ , since  $P_{\mathcal{O}_1}(T)$  realizes  $T$  there exists  $R' \in P_{\mathcal{O}_1}(T)$  and  $(s, s') \in T$  such that  $R \sqcap s$ ,  $R' \sqcap s$ ,  $s \xrightarrow{R} s'$ ,  $s \xrightarrow{R'} s'$  and  $v_{h(R)} = v_{h(R')}$ . Hence  $R$  and  $R'$  are conflicting, thus there exists  $R'' = v_{h(R)}^{\emptyset} \leftarrow \{v^{I''} \mid v^I \in b(R), v^{I'} \in b(R'), I \cap I' \subseteq I''\}$ . But then  $R'' \sqcap s$  and  $P_{\mathcal{O}_1}(T)$  is not suitable for  $T$ , a contradiction. Thus  $R$  is not conflicting with  $T$  and there exists a CL rule  $R_2 \in P_{\mathcal{O}_2}(T)$ , such that  $R \leq R_2$ . By the definition of an suitable program, there exists  $R_1 \in P_{\mathcal{O}_1}(T)$  such that  $R_2 \leq R_1$  since  $R_2$  is not conflicting with  $T$ . Thus  $R \leq R_1$  and by the definition of an optimal program  $R_1 \leq R$ . By Proposition 1,  $R_1 = R$  and thus  $R \leq R_2 \leq R$  hence  $R_2 = R$ , a contradiction.  $\square$

The starting point of the learning algorithm is  $P_{\mathcal{O}}(\emptyset)$ , described in the following proposition.

**Proposition 4.**  $P_{\mathcal{O}}(\emptyset) = \{v^{\emptyset} \leftarrow \{v^{I']^{-\infty, \infty[} \mid v' \in \mathcal{V}\} \mid v \in \mathcal{V}\}$ .

Proof. Let  $P = \{v^{\emptyset} \leftarrow \{v^{I']^{-\infty, \infty[} \mid v' \in \mathcal{V}\} \mid v \in \mathcal{V}\}$ . The CLP  $P$  is consistent and complete by construction. Like all CLPs,  $\overset{\emptyset}{\leftarrow} P$  and there is no transition in  $\emptyset$  to match with the rules in  $P$ . In addition, by construction, the rules of  $P$  dominate all CL rules.  $\square$

The CLP optimal for a set of transitions can be obtained from any CLP suitable for  $T$  by removing all the dominated rules from it, as stated in the following proposition. This means that it suffices to compute a CLP suitable for  $T$  to obtain  $P_{\mathcal{O}}(T)$  by getting rid of the dominated rules.

**Proposition 5.** Let  $T \subseteq \mathcal{S}^2$ . If  $P$  is a CLP suitable for  $T$ , then  $P_{\mathcal{O}}(T) = \{R \in P \mid \forall R' \in P, R \leq R' \text{ implies } R' \leq R\}$ .

**Proposition 6.** Let  $P$  be a consistent CLP and  $(s, s') \in \mathcal{S}^2$ . The CLP  $P_{\text{rev}}(P, (s, s'))$  is consistent.

Proof. Let us assume there exists two CL rules  $R_1, R_2 \in P_{\text{rev}}(P, (s, s'))$ . Since by Theorem 1, for all  $R \in P$ ,  $P_{\text{rev}}(R, (s, s'))$  is strongly consistent, necessarily there exists two distinct  $R'_1, R'_2 \in P$  such that  $R_1 \in P_{\text{rev}}(R'_1, (s, s'))$  and  $R_2 \in P_{\text{rev}}(R'_2, (s, s'))$ . The fact that  $R_1$  and  $R_2$  conflict implies that  $v_{h(R_1)} = v_{h(R_2)} = v$ . It also implies that  $R_1 \sqcap R_2$ . Whether  $R_1$  and  $R_2$  are obtained by least specialization or least generation, the following relationships hold by construction:

1.  $b(R_1) \sqsubseteq b(R'_1)$  and  $b(R_2) \sqsubseteq b(R'_2)$ ,
2.  $h(R'_1) \sqsubseteq h(R_1)$  and  $h(R'_2) \sqsubseteq h(R_2)$ .

Due to point 1,  $R'_1 \sqcap R'_2$ . Since in addition  $P$  is consistent and  $v_{h(R'_1)} = v_{h(R'_2)}$ , either  $h(R'_1) \sqsubseteq h(R'_2)$  or  $h(R'_2) \sqsubseteq h(R'_1)$ . If  $v_{h(R'_1)}$  and  $v_{h(R'_2)}$  are not empty, due to point 2, the same relationship also holds between  $R_1$  and  $R_2$ , a contradiction with the fact that there is a conflict between  $R_1$  and  $R_2$ . Otherwise, one of  $v_{h(R_1)}$  and  $v_{h(R_2)}$  is empty, thus its least generalization's head is sure to be a singleton. Assume w.l.o.g. that  $v_{h(R'_1)}$  is empty. Since  $R_1$  and  $R_2$  conflict with each other, their heads cannot be empty or subsume each other, thus  $\{R_1\} = P_{\text{gen}}(R'_1, (s, s'))$  and  $R_2 \in P_{\text{spe}}(R'_2, (s, s'))$ . Thus  $b(R_2)$  avoids  $s$  on a variable  $v_*$  and there is a rule  $R \in P_{\text{spe}}(R'_1, (s, s'))$  that avoids  $s$  on the same variable and in the same way (either over or under it). The rule  $R$  has an empty head since  $R'_1$  also has one and for all  $v \in \mathcal{V}$ , if  $v^{I_1} \in b(R_1)$ ,  $v^{I_2} \in b(R_2)$  and  $v^I \in b(R)$  then  $I_1 \cap I_2 \subseteq I$  since  $I$  coincides with  $I_1$  except on  $v_*$  where  $I \in \text{avoid}(I_1, \{x\})$  and  $I$  overlaps with  $I_2$  from its bound at  $x$  and until the bound of  $I_1$ , thus covering  $I_1 \cap I_2$  entirely.  $\square$

The following theorem in association with the three previous results gives a method to iteratively compute  $P_{\mathcal{O}}(T)$  for any  $T \subseteq \mathcal{S}^2$ , starting from  $P_{\mathcal{O}}(\emptyset)$ .

**Theorem 2.** *Let  $T \subseteq \mathcal{S}^2$  and  $(s, s') \in \mathcal{S}^2$ .  $P_{\text{rev}}(P_{\mathcal{O}}(T), (s, s'))$  is a CLP suitable for  $T \cup \{(s, s')\}$ .*

Proof. Let  $P = P_{\text{rev}}(P_{\mathcal{O}}(T), (s, s'))$ . Since  $P_{\mathcal{O}}(T)$  is consistent, by Proposition 6,  $P$  is also consistent. Since  $P_{\mathcal{O}}(T)$  is complete, by the two first points of Theorem 1,  $P$  is also complete. By Theorem 1,  $P$  is not in conflict with the rules of  $P_{\mathcal{O}}(T)$ , and since  $P$  is also complete,  $\xrightarrow{P} T$ . In addition, since  $P_{\mathcal{O}}(T)$  is complete, for each  $v \in \mathcal{V}$ , there exists a CL rule  $R \in P_{\mathcal{O}}(T)$  such that  $v_{h(R)} = v$  and  $R \sqcap s$ .

By Theorem 1, it means that for each of these rules,  $s \xrightarrow{P_{\text{gen}}(R, (s, s'))} s'$ , hence  $s \xrightarrow{P} s'$  and  $\xrightarrow{P} (T \cup \{(s, s')\})$ . Assume the existence of a rule  $R \in P$  with empty head and matching a state  $s''$  where  $(s'', s''') = t \in T \cup \{(s, s')\}$ . If  $R \in P_{\mathcal{O}}(T)$  then  $t = (s, s')$  or  $P_{\mathcal{O}}(T)$  is not suitable for  $T$ . In this case, since  $R \sqcap t$  and  $R$  has not been revised,  $s \xrightarrow{R} s'$ , a contradiction with the emptiness of the head of  $R$ . Otherwise, there exists a CL rule  $R' \in P_{\mathcal{O}}(T)$  such that  $R \in P_{\text{spe}}(R', t)$  because a generalization cannot produce rules with empty heads. Then by Theorem 1,  $t \neq (s, s')$  and since  $R \sqcap s''$ , we also have  $R' \sqcap s''$  by the definition of the specialization operation. For the same reason, the head of  $R'$  is empty. Thus,  $P_{\mathcal{O}}(T)$  is not suitable for  $T$ , a contradiction. To prove that  $P$  verifies the last point of the

definition of a suitable CLP, let  $R$  be a CL rule not conflicting with  $T \cup \{(s, s')\}$ . Since  $R$  is also not conflicting with  $T$ , there exists  $R' \in P_{\mathcal{O}}(T)$  such that  $R \leq R'$ . If  $R'$  is not conflicting with  $(s, s')$ , then  $R' \in P$ . Otherwise,  $R \leq R'$  and  $R'$  is in conflict with  $(s, s')$  (but  $R$  is not). Thus there exists at least one variable  $v \in \mathcal{V}$  such that  $v^I \in b(R)$ ,  $v^{I'} \in b(R')$ ,  $v^{\{x\}} \in s$   $x \in I'$ ,  $x \notin I$  and  $I \subseteq I'$ . Then one of the intervals in  $\text{avoid}(I', \{x\})$  contains  $I$  by the definition of the function  $\text{avoid}$ . Let us denote this interval by  $I''$ . The rule  $R'' \in P_{\text{spe}}(R', (s, s'))$  such that  $v^{I''} \in b(R'')$  verifies  $R \leq R''$  because  $v^I \sqsubset v^{I''}$  and  $R''$  coincides with  $R'$  on all other body and head variables.  $\square$

### 3 ACEDIA

In this section we present **ACEDIA**, the Abstraction-free Continuum Environment Dynamics Inference Algorithm and its experimental evaluation.

#### 3.1 Algorithm

**ACEDIA** learns a CLP from time-series data over continuous domains. Those time-series data are observations of a system's state transitions ( $\mathcal{S}^2$ ). Given as input a set of transitions  $T \subseteq \mathcal{S}^2$ , **ACEDIA** iteratively constructs a model of the system by applying the method formalized in the previous section to compute  $P_{\mathcal{O}}(T)$  as follows:

#### ACEDIA: Abstraction-free Continuum Environment Dynamics Inference Algorithm

- **INPUT:** a set of transitions  $T \subseteq \mathcal{S}^2$ .
- Initialize  $P$  as  $P_{\mathcal{O}}(\emptyset)$ .
- For each transition  $(s, s') \in T$ 
  - Extract each rule  $R$  of  $P$  that conflicts with  $(s, s')$ .
  - For each rule  $R$ 
    - \* Compute its least revision  $P' = P_{\text{rev}}(R, (s, s'))$ .
    - \* Remove all the rules in  $P'$  dominated by a rule in  $P$  or  $P'$ .
    - \* Remove all the rules in  $P$  dominated by a rule in  $P'$ .
    - \* Add all remaining rules in  $P'$  to  $P$ .
- **OUTPUT:**  $P = P_{\mathcal{O}}(T)$ .

Algorithms 1 and 2 provide the detailed pseudocode of the algorithm. Lines 3–6 of Algorithm 1 realize the computation of  $P_{\mathcal{O}}(\emptyset)$  as defined in Proposition 4. Then the learning is performed iteratively on each transition  $t \in T$  by applying the least-revision operation (lines 7–17) as defined in Definition 12 and removing dominated rules (lines 18–20) to ensure the optimality of the obtained program as stated in Proposition 5. The rules obtained by specialization that have empty intervals in their bodies are discarded since they cannot match or realize anything. Another noteworthy difference between the theory and the implementation is that the variables are not necessarily defined over all of  $\mathbb{R}$  but may be



constrained over an interval in  $\mathcal{I}_{\mathbb{R}}$  encompassing all the values associated to each variable in the observed transitions. This does not impact the theoretical results stated below.

**Theorem 3 (ACEDIA Termination, soundness, completeness, optimality).** *Let  $T \subseteq \mathcal{S}^2$ . The call  $\text{ACEDIA}(T)$  terminates and  $\text{ACEDIA}(T) = P_{\mathcal{O}}(T)$*

Proof. Let  $T \subseteq \mathcal{S}^2$ . The call  $\text{ACEDIA}(T)$  terminates because all loops iterate on finite sets.

To prove that  $\text{ACEDIA}(T) = P_{\mathcal{O}}(T)$ , and is thus sound, complete and optimal, it suffices to prove that the main loop (Algorithm 1 line 7–20) preserves the invariant  $P = P_{\mathcal{O}}(T_i)$  after the  $i^{\text{th}}$  iteration where  $T_i$  is the set of transitions already selected line 8 after the  $i^{\text{th}}$  iteration for all  $i$  from 0 to  $|T|$ .

Lines 3–6 initialize  $P$  to  $\{v^{\emptyset} \leftarrow \{v^j\}^{-\infty, \infty} \mid v^j \in \mathcal{V}\} \mid v \in \mathcal{V}$ . Thus by Proposition 4, after line 6,  $P = P_{\mathcal{O}}(\emptyset)$ .

Let us assume that before the  $i + 1^{\text{th}}$  iteration of the main loop,  $P = P_{\mathcal{O}}(T_i)$ . Through the loop lines 11–14,  $P' = \{R \in P_{\mathcal{O}}(T_i) \mid R \text{ does not conflict with } (s, s')\}$  is computed. Then the set  $P'' = \bigcup \{P_{\text{rev}}(R, (s, s')) \mid R \in P_{\mathcal{O}}(T_i) \setminus P'\}$  is iteratively build through the calls to **least\_revision** line 17 and the dominated rules are pruned as they are detected by the loop lines 18–20. Thus by Theorem 2 and Proposition 5,  $P = P_{\mathcal{O}}(T_{i+1})$  after the  $i + 1^{\text{th}}$  iteration of the main loop.  $\square$

**Theorem 4 (ACEDIA Complexity).** *Let  $T \subseteq \mathcal{S}^2$  be a set of transitions and  $|\mathcal{V}| = n$ . The worst-case time complexity of **ACEDIA** when learning from  $T$  belongs to  $\mathcal{O}(|T|^{2n} \times n^5)$  and its worst-case memory use belongs to  $\mathcal{O}(|T|^{2n} \times n^2)$ .*

Proof. Let  $x_i$  be the different values a variable  $v_i$  takes in  $T$ . Each  $x_i$  can be the minimum or maximum value of a continuum. After the initialization of  $P$  there is a rule with empty head for each  $x_i$ . According to the definition of the

---

**Algorithm 1. ACEDIA( $T$ )**


---

```

1: INPUT:  $T \subseteq \mathcal{S}^2$ 
2: OUTPUT:  $P_{\mathcal{O}}(T)$ 

3: // 1) Initialization of P
4:  $P = \emptyset$  // The empty logic program
5: for each  $v \in \mathcal{V}$  do
6:    $P := P \cup \{v^{\emptyset} \leftarrow \{v^j\}^{-\infty, \infty} \mid v^j \in \mathcal{V}\}$ 

7: // 2) Revision of P for each transition
8: for each  $(s, s') \in T$  do :
9:    $\text{conflicts} := \emptyset$ 
10:  //2.1) Extraction of conflicting rules
11:  for each  $R \in P$  do
12:    if  $b(R)$  conflicts with  $(s, s')$  then
13:       $P := P \setminus \{R\}$ 
14:       $\text{conflicts} := \text{conflicts} \cup \{R\}$ 
15:  //2.2) Revision of conflicting rules
16:  for each  $R \in \text{conflicts}$  do
17:     $LR := \text{least\_revision}(R, (s, s'))$ 
18:    for each  $R' \in LR$  do
19:      if  $\nexists R'' \in P, R' \leq R''$  then
20:         $P := P \setminus \{R''\} \cup \{R'\} \cup \{R'\}$ 

21: return P
```

---



---

**Algorithm 2. least\_revision( $R, t$ )**


---

```

1: INPUT: a rule  $R$  and a transition  $t = (s, s')$ ,
2: OUTPUT:  $LR = P_{\text{rev}}(R, t)$ .

3: // 1) Least generalization
4:  $x = I, v^I \in s'$ 
5: if  $h(R) = v^{\emptyset}$  then
6:    $h = v\{x\}$ 
7: else
8:   if  $x < \min(h(R))$  then
9:      $h := v[x, \max(h(R))]$ 
10:  else
11:     $h := v[\min(h(R)), x]$ 
12:  $LR := \{h \leftarrow b(R)\}$ 

13: // 2) Least specialization
14: for each  $v^I \in b(R)$  do
15:    $R_{\min} :=$ 
16:    $h(R) \leftarrow (b(R) \setminus \{v^I\} \cup \{v\}^{\min(I), x} \mid \min(I) \neq x)$ 
17:    $R_{\max} :=$ 
18:    $h(R) \leftarrow (b(R) \setminus \{v^I\} \cup \{v\}^{x, \max(I)} \mid \max(I) \neq x)$ 
19:    $LR := LR \cup \{R_{\min}, R_{\max}\}$ 

20: return LR
```

---

least generalization (Definition 11), those continuum will only be extended to hull a  $x_i$ , i.e. their min/max will always be a  $x_i$ . Thus the number of possible head continuums of a learned rule is  $1 + |x_i|^2$  which belongs to  $\mathcal{O}(1 + |T|^2)$ . Similarly for the rule body, according to the definition of the least specialization (Definition 10), a continuum in a rule body is only reduced to avoid one of the  $x_i$ . The only other possible values are  $\infty$  and  $-\infty$ , thus the number of possible continuums for each body variable belongs to  $\mathcal{O}((|x_i| + 2)^2) \sim \mathcal{O}((|T| + 2)^2)$ . The possible bodies of a rule are all the combinations of these continuums, thus belong to  $\mathcal{O}(((|T| + 2)^2)^n)$ . Hence, the total number of possible rules learned by **ACEDIA** belongs to  $\mathcal{O}(n \times (1 + |T|^2) \times ((|T| + 2)^{2n}))$ . The heads of rules are represented by an integer that encodes the variable it refers to and a continuum represented by two real numbers. The body of a rule is a vector of  $n$  pairs of variable/continuum encoded in the same way. Thus the memory use of a rule belongs to  $\mathcal{O}(3 + 3 \times n)$ . **Conclusion:** the memory use of **ACEDIA** belongs to  $\mathcal{O}(|P|) \sim \mathcal{O}(n \times ((1 + |T|^2) \times ((|T| + 2)^{2n}) \times (3 + 3 \times n))$ , i.e.  $\mathcal{O}(|P|) \sim \mathcal{O}(|T|^{2n} \times n^2)$ .

**ACEDIA** starts with the generation of the logic program  $P_{\mathcal{O}}(\emptyset)$ , containing  $n$  rules. A rule  $R$  has one head variable and its body contains each variable: building a rule belongs to  $\mathcal{O}(|R|) \sim \mathcal{O}(1 + n)$ . Thus the initialization of  $P$  belongs to  $\mathcal{O}(n \times |R|)$ . Afterward, the algorithm checks each transition of  $T$  iteratively and extracts conflicting rules from  $P$ . Checking conflict between two rules requires to compare their heads and all their body atoms: it belongs to  $\mathcal{O}(|R|)$ . Each rule of  $P$  is checked, thus extracting conflicting rules belongs to  $\mathcal{O}(|P| \times |R|)$ . Each conflicting rule is revised using least revision (Definition 12). This operation generates one rule by least generalization (one head continuum extension) and  $2n$  rules by least revision (2 for each atom in the body: one that avoids the value from below and the other that avoids it from the top), it belongs to  $\mathcal{O}(|LR|) \sim \mathcal{O}((1 + 2n) \times |R|)$ . Each revision is then compared to the rules of  $P$  to check domination (Definition 2). Checking domination requires to compare head continuum and all body continuum: it belongs to  $\mathcal{O}(|R|)$ . Thus removing the dominated revisions belongs to  $\mathcal{O}(|LR| \times |R| \times |P|)$ . This is repeated for each transition of  $T$ , thus the complete process belongs to  $\mathcal{O}(n \times |R| + |T| \times |LR| \times |R| \times |P|) \sim \mathcal{O}(n \times (1 + n) + |T| \times (1 + 2n) \times (1 + n) \times (1 + n) \times n \times (1 + |T|^2) \times (|T| + 2)^{2n} \times (3 + 3 \times n)) \sim \mathcal{O}(|T| \times (1 + |T|^2) \times (|T| + 2)^{2n} \times n^5)$  **Conclusion:** the complexity of **ACEDIA** when learning from  $T$  belongs to  $\mathcal{O}(|T|^{2n} \times n^5)$ .  $\square$

### 3.2 Evaluation

In this section, the benefits from **ACEDIA** are demonstrated on a case study and its scalability is assessed w.r.t. the input size and the number of variables. All experiments were conducted on an Intel Core I7 (6700, 3.4 GHz) with 32 Gb of RAM and can be accessed via the hyperlink given in footnote<sup>1</sup>.

The first evaluation is a case study on learning a **CLP** equivalent to a Boolean network of 3 variables. For the purpose of this experiment the levels of expression

<sup>1</sup> Experiments sources: [http://tonyribeiro.fr/data/experiments/ILP\\_2017.zip](http://tonyribeiro.fr/data/experiments/ILP_2017.zip).

$ \begin{array}{l} p^{[0, \mathbf{P1}] \leftarrow q^{[0, \mathbf{Q}]}. \\ p^{[\mathbf{P1}, 1]} \leftarrow q^{[\mathbf{Q}, 1]}. \\ \\ q^{[0, \mathbf{Q}] \leftarrow p^{[0, \mathbf{P1}]}. \\ q^{[0, \mathbf{Q}] \leftarrow r^{[0, \mathbf{R}]}. \\ q^{[\mathbf{Q}, 1]} \leftarrow p^{[\mathbf{P1}, 1]}, r^{[\mathbf{R}, 1]}. \\ \\ r^{[0, \mathbf{R}] \leftarrow p^{[\mathbf{P2}, 1]}. \\ r^{[\mathbf{R}, 1]} \leftarrow p^{[0, \mathbf{P2}]}. \end{array} $	$ \begin{array}{l} p^{[0, 0.5[ \leftarrow q^{[0, 0.5[}. \\ p^{[0.5, 1]} \leftarrow q^{[0.5, 1]}. \\ \\ q^{[0, 0.5[ \leftarrow p^{[0, 0.5[}. \\ q^{[0, 0.5[ \leftarrow r^{[0, 0.5[}. \\ q^{[0.5, 1]} \leftarrow p^{[0.5, 1]}, r^{[0.5, 1]}. \\ \\ r^{[0, 0.5[ \leftarrow p^{[0.75, 1]}. \\ r^{[0.5, 1]} \leftarrow p^{[0, 0.75[}. \end{array} $	$ \begin{array}{l} p^{[0, 0.25]} \leftarrow q^{[0, 0.5[}. \\ p^{[0.5, 1]} \leftarrow q^{[0.25, 1]}. \\ p^{[0, 1]}. \\ \\ q^{[0, 0.25]} \leftarrow p^{[0, 0.5[}. \\ q^{[0, 0.25]} \leftarrow r^{[0, 0.5[}. \\ q^{[0.5, 1]} \leftarrow p^{[0.25, 1]}, r^{[0.25, 1]}. \\ q^{[0, 1]}. \\ \\ r^{[0, 0.25]} \leftarrow p^{[0.5, 1]}. \\ r^{[0.5, 1]} \leftarrow p^{[0, 0.75[}. \\ r^{[0, 1]}. \end{array} $
---	--	---

(a) *CLP* with editable levels of expression in bold. (b) *CLP* with levels of expression set to  $P1=Q=R=0.5$  and  $P2=0.75$ . (c) **ACEDIA** output from all the transitions of the *CLP* in Figure 2b

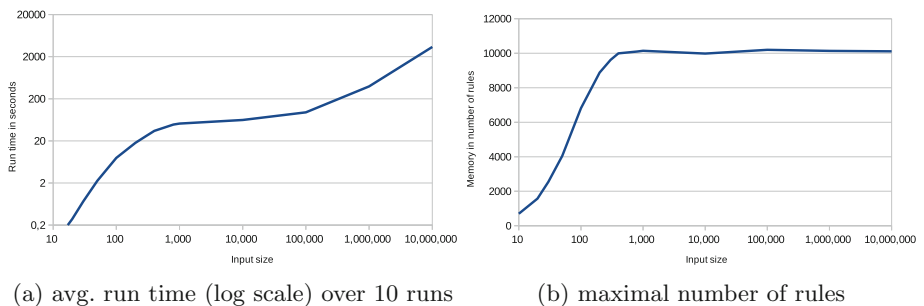
**Fig. 2.** Experimentation on a *CLP* with three variables.

can be changed by setting the condition/conclusion intervals as shown in Fig. 2a and b. In the rules body,  $q$  and  $r$  have a unique expression level but the level of  $p$  differs in the dynamics of  $q$  and  $r$ : to activate  $q$ ,  $p = 0.5$  is enough but  $p = 0.75$  is necessary to inhibit  $r$ . This is done to show that **ACEDIA** can learn different behaviors and different expression levels for the same variable, while previous versions of **LFIT** assumed the same discretization in all rules. The domain of each variable is restricted to  $[0, 1]$ , which can be considered like a normalization of the time-series in practice. For readability reasons, in the body of the rules, variables of which the corresponding value is the whole domain are omitted. Considering a precision of 0.25 for each variable value, 150 possible states are generated. The algorithm computes the approximately 1700 possible transitions according to the *CLP*. From those transitions, **ACEDIA** learns the original rules to capture the dynamics of the system and finds the expression level of the variables. **ACEDIA**'s output is shown in Fig. 2c. The rules are ordered and grouped to follow Fig. 2b and the rules with empty heads are omitted. They encode all non-observed states, e.g.  $p^\emptyset \leftarrow q^{[0, 0.25[}$  and  $r^\emptyset \leftarrow q^{[0.25, 0.75[}$ . Such minor differences have been highlighted in bold in Fig. 2c. The first rule is equivalent to the first rule of  $p$  of the original program: the total range of value a variable can take is  $[0, 1]$ , thus conditions on this continuum can be ignored. By looking closely to the first rule in Fig. 2c, it appears to be different from the first rule in Fig. 2b: the head of the former is equal to  $[0, 0.25]$  while the latter is equal to  $[0, 0.5[$ . This is as close an approximation as is possible with a precision of 0.25 in the states, and the fact that only closed bounds (respectively open bounds) can be created in the head (respectively body) of a rule. The second rule is equivalent to the second rule of  $p$ : here the target continuum  $[0.5, 1]$  is approximated by  $]0.25, 1]$ . The third, as the most general conclusion and conditions, just provides

the domain of the variable; it is an optimal rule of the observations that can be expected to be learned. Here we see that the influence of  $q$  over  $p$  is correctly learned as well as the level of expression of  $q$  to influence  $p$  which is 0.5. Regarding the dynamics of  $q$ , the rules are equivalent to the three original rules of  $q$  if we follow the same reasoning as before. The dynamics of the influences of  $p$  and  $r$  over  $q$  is also learned correctly, as well as the level of expression of  $p$  and  $r$ . Again line 7 we have a similar rule that just gives the domain of  $q$  and can be ignored. The rest of the rules are equivalent to the rules of  $r$  of the original program. Here we see that the specific level of expression of  $p$  to inhibit  $r$ ,  $P2 = 0.75$  is approximated by the continuum  $]0.5, 1]$ . This experiment shows that the dynamics of the system and the expression level of each variable are approximated as well as possible by **ACEDIA**.

Figure 3 shows the run time (Fig. 3a) and memory use (Fig. 3b) of **ACEDIA** on learning the CLP of Fig. 2b (middle) with regards to the number of input transitions. In this experiment, the precision of the value of each variable is 0.1 in the interval  $[0,1]$ , thus there are 11 possible values for a variable and  $11 \times 11 = 121$  possible continuums. In theory, more than 200 millions of rules ( $121$  heads times  $121^3$  bodies) could be learned, but this number never exceeds much more than 10,000 at a given time. The domination relation (Definition 2) allows to reduce the number of candidate rules at each step of the learning process. After approximately 400 transitions, the real rules of the system are learned and most of the computation consists in checking those rules against the new transitions and eliminating the remaining rules that are still specific enough to survive. This experiment shows that when the observed system has a small number of variables, the algorithm can be fed with as many observations as wanted.

Table 1 shows the run times and memory use of **ACEDIA** on learning partial mammalian cell Boolean networks [5] by varying the number of variables considered. The original number of variables is 10. To reduce the variables to  $n < 10$ , we removed the occurrences of  $10 - n$  variables in all original rules, thus creating a new system of  $n$  variables. In this experiment the exponential evolution of run



**Fig. 3.** Evaluation of **ACEDIA**'s scalability w.r.t. input size (log scale) on learning the CLP of Fig. 2b.

**Table 1.** Evaluation of **ACEDIA** scalability over 10 runs: average run time and memory use (in number of rules) on learning mammalian cell Boolean Network evolving the number of variables. Time Out (T.O.) is 10 h.

Variables	Run time/Rules w.r.t. input transitions ( $ T $ )						
	$ T  = 10$	$ T  = 25$	$ T  = 50$	$ T  = 100$	$ T  = 250$	$ T  = 500$	$ T  = 1000$
2	0.015 s/137	0.031 s/296	0.067 s/425	0.105 s/437	0.143 s/420	0.217 s/422	0.337 s/456
3	0.034 s/464	0.667 s/3 K	3.082 s/5 K	10 s/8 K	31 s/11 K	56 s/13 K	73 s/13 K
4	0.322 s/2 K	16 s/14 K	127 s/36 K	1081 s/86 K	T.O.	T.O.	T.O.
5	2.8 s/7 K	239 s/58 K	4,522 s/203 K	T.O.	T.O.	T.O.	T.O.
6	15 s/21 K	4,063 s/217 K	T.O.	T.O.	T.O.	T.O.	T.O.
7	73 s/45 K	22,616 s/596 K	T.O.	T.O.	T.O.	T.O.	T.O.
8	424 s/120 K	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.
9	2,239 s/228 K	T.O.	T.O.	T.O.	T.O.	T.O.	T.O.

time caused by the exponential explosion of the number of generated rules can be seen. For now, **ACEDIA** cannot handle systems with more than 9 variables in a reasonable amount of time and memory when considering 10 transitions and it tends to be limited to 4 variables when more than 10 input transitions are studied. However, as in the previous experiment, we observe the convergence of the number of rules and thus run time for 2 and 3 variables, which hints that such behavior should occur for more variables but with an exponentially greater input size. The current implementation is rather naïve. As with previous **LFIT** algorithms, we can expect better experimental results regarding scalability by developing dedicated data structures and learning heuristics. Such improvements remain as future work.

## 4 Conclusions

In the previous **LFIT** algorithms, it was assumed that the discretization of raw input data was performed by some third-party agent. Such a hypothesis is rather naïve, and does not match with real-life systems since the dynamics of a system is defined by both the levels of expression of variables and their influences on each other. In this paper, we introduce **ACEDIA**, an algorithm to learn the dynamics of a system directly from continuous time-series data. For this purpose we propose a modeling of continuous dynamics by continuum logic programs. As far as we know, this approach is completely new and its strengths and weaknesses are shown through theoretical results and practical evaluations. Similarly to continuous approaches, the modeling we propose allows to deal with real-valued measured data. It however assumes discrete time steps. One of our future works will thus address continuous time dynamics in the **LFIT** framework. It is important to note that this method can also be applied to discrete data like previous **LFIT** algorithms. In the case of multi-valued discrete data, **ACEDIA** learns more compact and expressive rules. Indeed, multiple conditions over different contiguous discrete levels can be expressed by one condition

over a continuum including those levels. Where previous **LFIT** algorithms need several rules to express those conditions, **ACEDIA** expresses them with a single one. The detailed comparison of **ACEDIA** with previous **LFITs** on this kind of data is out of the scope of this paper and remains as a future work.

This paper focuses on the theoretical bases of **ACEDIA** and we are now working on an efficient implementation of the algorithm, with the goal of applying it to real biological time-series data. The complexity of the current algorithm (see Theorem 4) limits its current usability to rather small systems as shown by the experimental results. However, the convergences observed gives us good hope about the practical use of the methods.

## References

1. Allen, J.F.: An interval-based representation of temporal knowledge. *IJCAI* **81**, 221–226 (1981)
2. Benhamou, F., Goualard, F., Granvilliers, L., Puget, J.-F.: Revising hull and box consistency. In: *Logic Programming: Proceedings of the 1999 International Conference on Logic Programming*, pp. 230–244. MIT press (1999)
3. do Carmo Nicoletti, M., de Sá Lisboa, F.O.S., Hruschka, E.R.: Learning temporal interval relations using inductive logic programming. In: Hruschka, E.R., Watada, J., do Carmo Nicoletti, M. (eds.) *INTECH 2011. CCIS*, vol. 165, pp. 90–104. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22247-4\\_8](https://doi.org/10.1007/978-3-642-22247-4_8)
4. Emerson, E.A.: Temporal and modal logic. In: *Handbook of Theoretical Computer Science, Formal Models and Semantics*, vol. B-5, pp. 995–10725 (1990)
5. Fauré, A., Naldi, A., Chaouiya, C., Thieffry, D.: Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics* **22**(14), e124–e131 (2006)
6. Friedman, N., Linial, M., Nachman, I., Pe’er, D.: Using Bayesian networks to analyze expression data. *J. Comput. Biol.* **7**(3–4), 601–620 (2000)
7. Inoue, K., Ribeiro, T., Sakama, C.: Learning from interpretation transition. *Mach. Learn.* **94**(1), 51–79 (2014)
8. Karlebach, G., Shamir, R.: Modelling and analysis of gene regulatory networks. *Nat. Rev. Mol. Cell Biol.* **9**(10), 770–780 (2008)
9. Kim, S.Y., Imoto, S., Miyano, S.: Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings Bioinf.* **4**(3), 228–235 (2003)
10. Martínez Martínez, D., Ribeiro, T., Inoue, K., Alenyà Ribas, G., Torras, V.: Learning probabilistic action models from interpretation transitions. In: *Proceedings of the Technical Communications of the 31st International Conference on Logic Programming (ICLP 2015)*, pp. 1–14 (2015)
11. Nachman, I., Regev, A., Friedman, N.: Inferring quantitative models of regulatory networks from expression data. *Bioinformatics* **20**(suppl 1), i248–i256 (2004)
12. Cloud, M.J., Moore, R.E., Kearfott, R.B.: *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics (2009)
13. Ribeiro, T., Magnin, M., Inoue, K., Sakama, C.: Learning delayed influences of biological systems. *Front. Bioeng. Biotechnol.* **2**, 81 (2015)
14. Rodríguez, J.J., Alonso, C.J., Boström, H.: Learning first order logic time series classifiers: rules and boosting. In: Zighed, D.A., Komorowski, J., Żytkow, J. (eds.) *PKDD 2000. LNCS (LNAI)*, vol. 1910, pp. 299–308. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45372-5\\_29](https://doi.org/10.1007/3-540-45372-5_29)



# Stacked Structure Learning for Lifted Relational Neural Networks

Gustav Šourek<sup>1</sup>(✉), Martin Svatoš<sup>1</sup>, Filip Železný<sup>1</sup>, Steven Schockaert<sup>2</sup>,  
and Ondřej Kuželka<sup>3</sup>

<sup>1</sup> Czech Technical University, Prague, Czech Republic  
{souregus,svatoma1,zelezny}@fel.cvut.cz

<sup>2</sup> School of CS and Informatics, Cardiff University, Cardiff, UK  
SchockaertS1@cardiff.ac.uk

<sup>3</sup> Department of Computer Science, KU Leuven, Leuven, Belgium  
ondrej.kuzelka@kuleuven.be

**Abstract.** Lifted Relational Neural Networks (LRNNs) describe relational domains using weighted first-order rules which act as templates for constructing feed-forward neural networks. While previous work has shown that using LRNNs can lead to state-of-the-art results in various ILP tasks, these results depended on hand-crafted rules. In this paper, we extend the framework of LRNNs with structure learning, thus enabling a fully automated learning process. Similarly to many ILP methods, our structure learning algorithm proceeds in an iterative fashion by top-down searching through the hypothesis space of all possible Horn clauses, considering the predicates that occur in the training examples as well as invented soft concepts entailed by the best weighted rules found so far. In the experiments, we demonstrate the ability to automatically induce useful hierarchical soft concepts leading to deep LRNNs with a competitive predictive power.

## 1 Introduction

Lifted Relational Neural Networks (LRNNs [15]) are weighted sets of first-order rules, which are used to construct feed-forward neural networks from relational structures. A central characteristic of LRNNs is that a different neural network is constructed for each learning example, but crucially, the weights of these different neural networks are shared. This allows LRNNs to use neural networks for learning in relational domains, despite the fact that training examples may vary considerably in size and structure.

In previous work, LRNNs have been learned from hand-crafted rules. In such cases, only the weights of the first-order rules have to be learned from training data, which can be accomplished using a variant of back-propagation. The use of hand-crafted rules offers a natural way to incorporate domain knowledge in the learning process. In some applications, however, (sufficient) domain knowledge is lacking and both the rules and their weights have to be learned from data. To this end, in this paper we introduce a structure learning method for LRNNs.

Our proposed structure learning method proceeds in an iterative fashion. In each iteration, it may either learn a set of rules that intuitively correspond to a new layer of a neural network template or to learn a set of rules that intuitively correspond to creating new connections among existing layers, a strategy which we refer to as stacked structure learning. The rules that are added in a given iteration either define one of the target predicates, or they define a new predicate that may depend on predicates that were ‘invented’ at earlier layers as well as on predicates from the considered domain. Since the actual meaning of these predicates depends on both the learned rules and their associated weights, structure learning is alternated with weight learning. Intuitively, this means that the definitions of predicates defined in earlier layers can be fine-tuned based on the rules which are added to later layers.

We present experimental result which show that the resulting LRNNs perform comparably to LRNNs that have been learned from hand-crafted rules. We believe that this makes LRNNs a particularly convenient framework for learning in relational domains, without any need for prior knowledge nor for any extensive hypertuning. Somewhat surprisingly, we find that LRNNs with learned rules are often more compact than those with hand-crafted rules. Finally, we also present some initial results which suggest that the use of logical rules enable LRNNs to efficiently learn concepts which neural networks normally struggle with.

The remainder of the paper is structured as follows. In the next section, we first provide the required background on LRNNs. In Sect. 3, we then present the proposed structure learning method, after which we discuss our experimental results in Sect. 4.

## 2 Preliminaries

In this section, we briefly recall the LRNN framework from [15].

**LRNN Structure.** A lifted relational neural network (LRNN)  $\mathcal{N}$  is a set of weighted definite clauses, i.e. a set of pairs  $(R_i, w_i)$  where  $R_i$  is a definite clause and  $w_i \in \mathbb{R}$ . For a LRNN  $\mathcal{N}$ , we write  $\mathcal{N}^*$  to denote the corresponding set of definite clauses, i.e.  $\mathcal{N}^* = \{C \mid (C, w) \in \mathcal{N}\}$ . The grounding  $\overline{\mathcal{N}}$  of a LRNN  $\mathcal{N}$  is defined as  $\overline{\mathcal{N}} = \{(C\theta, w) \mid (C, w) \in \mathcal{N}, C\theta \in G(\mathcal{N}^*)\}$ , where  $G(\mathcal{N}^*)$  is the restriction of the grounding of  $\mathcal{N}^*$  to those clauses that correspond to active rules, i.e. rules whose antecedent is satisfied in the least Herbrand model of  $\mathcal{N}^*$ . The neural network corresponding to  $\overline{\mathcal{N}}$  contains the following types of neurons:

- For each ground atom  $h$  occurring in  $\overline{\mathcal{N}}$ , there is a neuron  $A_h$ , called an *atom neuron*.
- For each ground fact  $(h, w) \in \overline{\mathcal{N}}$ , there is a neuron  $F_{(h,w)}$ , called a *fact neuron*.
- For every ground rule  $(c\theta \leftarrow b_1\theta \wedge \dots \wedge b_k\theta, w) \in \overline{\mathcal{N}}$ , there is a neuron  $R_{(c\theta \leftarrow b_1\theta \wedge \dots \wedge b_k\theta, w)}$ , called a *rule neuron*.
- For every (possibly non-ground) rule  $(c \leftarrow b_1 \wedge \dots \wedge b_k, w) \in \mathcal{N}$  and every grounding  $h = c\theta$  of  $c$  that occurs in  $\mathcal{H}$ , there is a neuron  $Agg_{(c \leftarrow b_1 \wedge \dots \wedge b_k, w)}^h$ , called an *aggregation neuron*.



**Forward Propagation.** Intuitively, the neural network computes for each ground atom  $h$  a truth value, which is given by the output of the atom neuron  $A_h$ . To obtain these truth values, the network propagates values in a way which closely mimics the immediate consequence operator from logic programming. In particular, when using the immediate consequence operator, there are two ways in which  $h$  can become true: if  $h$  corresponds to a fact, or if  $h$  is the head of a rule whose body is already satisfied. Similarly, the inputs of the atom neuron  $A_h$  consist of the fact neurons of the form  $F_{(h,w)}$  and aggregation neurons of the form  $Agg_{(c \leftarrow b_1 \wedge \dots \wedge b_k, w)}^h$ . The output of an atom neuron with inputs  $i_1, \dots, i_m$  is given by  $g_{\vee}(i_1, \dots, i_m)$ , where  $g_{\vee}$  is an activation function that maps the inputs to a real-valued output. In this paper we will use

$$g_{\vee}(b_1, \dots, b_k) = \text{sigm} \left( a \cdot \left( \sum_{i=1}^k b_i + b_0 \right) \right)$$

where  $\text{sigm}$  is the sigmoid function  $\text{sigm}(x) = 1/(1+e^{-x})$ . We set the parameters  $a = 6$  and  $b_0 = -0.5$ , as  $g_{\vee}$  then closely approximates the Łukasiewicz fuzzy disjunction [7] (see right panel in Fig. 1). This helps with the interpretability of LRNNs, as it means that we can intuitively think of the activation functions as logical connectives, and of LRNNs as (fuzzy) logic programs.

A fact neuron  $F_{(h,w)}$  has no input and has the value  $w$  as its output. The output of the aggregation neuron  $Agg_{(c \leftarrow b_1 \wedge \dots \wedge b_k, w)}^h$  intuitively expresses how strongly  $h$  can be derived using the rule  $c \leftarrow b_1 \wedge \dots \wedge b_k$ . The inputs of the aggregation neuron  $Agg_{(c \leftarrow b_1 \wedge \dots \wedge b_k)}^h$  are all rule neurons  $R_{(c\theta \leftarrow b_1\theta \wedge \dots \wedge b_k\theta, w)}$  for which  $c\theta = h$ . The output of this aggregation neuron is given by  $w \cdot g_*(i_1, \dots, i_m)$ , where  $i_1, \dots, i_m$  are its inputs,  $g_*$  is an activation function, and  $w$  is the weight of the corresponding rule. We will use

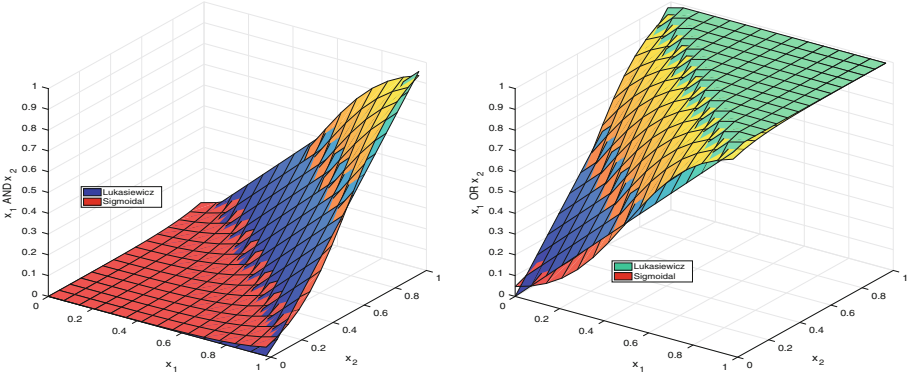
$$g_*(b_1, \dots, b_m) = \frac{1}{m} \sum_{i=1}^m b_i.$$

The rule neuron  $R_{(c\theta \leftarrow b_1\theta \wedge \dots \wedge b_k\theta, w)}$  intuitively needs to fire if the atoms  $b_1\theta, \dots, b_k\theta$  are all true. Accordingly, its inputs  $i_1, \dots, i_k$  are given by the atom neurons  $A_{b_1\theta}, \dots, A_{b_k\theta}$ , and its output is  $g_{\wedge}(i_1, \dots, i_k, w)$ , with  $g_{\wedge}$  a third type of activation function. In this paper we will use the activation function

$$g_{\wedge}(b_1, \dots, b_k) = \text{sigm} \left( a \cdot \left( \sum_{i=1}^k b_i - k + 1 + b_0 \right) \right)$$

where we set  $a = 6$  and  $b_0 = -0.5$ , which approximates Łukasiewicz fuzzy conjunction [7] (see left panel in Fig. 1).

**Weight Learning.** In applications, we usually consider LRNNs of the form  $\mathcal{N} \cup \mathcal{E}$ , where  $\mathcal{N}$  is a weighted set of first-order rules and  $\mathcal{E}$  is a weighted set of ground facts. In particular, each  $\mathcal{E}$  represents an example, while  $\mathcal{N}$  acts as a



**Fig. 1.** An approximation of Łukasiewicz conjunction (left) and disjunction (right) by sigmoidal activation functions  $g_{\wedge}$  and  $g_{\vee}$  for the use in LRNNs.

template for constructing feed-forward neural networks, with  $\overline{\mathcal{N} \cup \mathcal{E}}$  being the network corresponding to example  $\mathcal{E}$ . While the weights of  $\mathcal{E}$  are given, the weights of  $\mathcal{N}$  typically need to be learned from training data, as follows.

We are given a list of examples  $\mathcal{E} = (\mathcal{E}^1, \dots, \mathcal{E}^m)$  where each  $\mathcal{E}^j$  is a LRNN, typically containing only weighted ground facts, and a list of training queries  $\mathcal{Q} = (\{(q_1^1, t_1^1), \dots, (q_{k_1}^1, t_{k_1}^1)\}, \dots, \{(q_1^m, t_1^m), \dots, (q_{k_m}^m, t_{k_m}^m)\})$  where each  $q_i^j$  is a ground atom, which we call a *training query atom*, and  $t_i^j$  is its *target value*. For a query atom  $q_i^j$ , let  $y_i^j$  denote the output of the atom neuron  $A_{q_i^j}$  in the ground neural network of  $\overline{\mathcal{N} \cup \mathcal{E}^j}$ . The goal of the learning process is to find the weights  $w_h$  of the rules (and possibly facts) in  $\mathcal{N}$  for which the loss  $J$  on the training query atoms  $J(\mathcal{Q}) = \sum_{j=1}^m \sum_{i=1}^{k_j} \text{loss}(y_i^j, t_i^j)$  is minimized. This loss function is then optimized using standard stochastic gradient descent algorithm [2]. For details about weight learning of LRNNs, see [15].

### 3 Structure Learning

In this section we describe a structure learning algorithm for LRNNs. The algorithm receives a list of training examples and a list of training queries, and it produces a LRNN. For simplicity, we will assume that constants are only used as identifiers of objects. In particular, we will assume that attribute values are represented using unary literals, e.g. we would use  $\text{red}(o)$  instead of  $\text{color}(o, \text{red})$ . Besides that we do not put any restrictions on the structure of the training examples.

#### 3.1 Structure of the Learned LRNNs

The structure learning algorithm will create LRNNs having a generic “stacked” structure which we now describe. First, there are rules that define  $d$  new predicates, representing *soft clusters* [17] of unary predicates from the dataset.

These can be thought of as the first layer of the LRNN, where the weighted facts from the dataset comprise the zeroth layer. For instance, if the unary predicates in the dataset are  $A, B, \dots, Z$  then the LRNN will contain the following rules:

$$\begin{array}{llll} w_{a_1} : \alpha_1^1(X) \leftarrow A(X) & w_{b_1} : \alpha_1^1(X) \leftarrow B(X) & \dots & w_{z_1} : \alpha_1^1(X) \leftarrow Z(X) \\ w_{a_2} : \alpha_2^1(X) \leftarrow A(X) & w_{b_2} : \alpha_2^1(X) \leftarrow B(X) & \dots & w_{z_2} : \alpha_2^1(X) \leftarrow Z(X) \\ \dots & \dots & \dots & \dots \\ w_{a_d} : \alpha_d^1(X) \leftarrow A(X) & w_{b_d} : \alpha_d^1(X) \leftarrow B(X) & \dots & w_{z_d} : \alpha_d^1(X) \leftarrow Z(X) \end{array}$$

Here each  $\alpha_j^i$  is a latent predicate representing a soft cluster, the index  $i$  denotes the layer in which it appears (in this case, the first layer) and  $j$  indexes the individual soft clusters in that level.

In general, the second layer will consist of two types of rules. First, there may be rules introducing new latent predicates. In contrast to the unary predicates that were introduced in the first layer, here the latent predicates could be also of higher arity, although in practice an upper bound will be imposed for efficiency reasons. In the body of these rules, we may find predicates from the dataset itself, or latent predicates that were introduced in the first layer. The new latent predicates introduced in these rules may then be used in the bodies of rules in subsequent layers. Second, there may also be rules that have a predicate from the dataset in their head. These will typically be rules that were learned to predict the target predicates that we want to learn.

*Example 1.* For instance, in datasets of molecules, unary predicates can be used to represent types of atoms, such as *carbon* or *hydrogen*. An example of a possible second layer rule is:

$$w_{p_1} : p_1(X, Y) \leftarrow \text{bond}(X, Y) \wedge \alpha_1^1(X) \wedge \alpha_2^1(Y)$$

Here  $p_1$  is assumed to be one of the predicates from the dataset. Second layer rules that introduce a new latent predicate could look as follows.

$$\begin{array}{ll} w_{1,1}^2 : \alpha_1^2(V1, V2) \leftarrow \text{bond}(V1, V2) \wedge \alpha_1^1(V1) \wedge \alpha_1^1(V2) \\ w_{1,2}^2 : \alpha_1^2(V1, V3) \leftarrow \text{bond}(V1, V2) \wedge \text{bond}(V2, V3) \wedge \alpha_1^1(V1) \wedge \alpha_1^1(V3) \end{array}$$

The actual intuitive meaning of the predicate  $\alpha_1^2$  will depend on the weights  $w_{1,1}^2$ ,  $w_{1,2}^2$ . For instance, if both are large enough, the (atom neurons corresponding to the) predicate will have high output whenever its arguments correspond to two atoms which are either one or two steps apart from each other in the molecule, and which have sufficiently high membership in the soft cluster  $\alpha_1^1$ .

Any higher layers have a similar structure to the second layer, where the  $n^{\text{th}}$  layer contains rules whose bodies only contain predicates from layers 0 to  $n - 1$ , and whose heads either contain a target predicate or introduce a new latent predicate.

**Algorithm 1.** General schema of structure learning

---

```

1:  $\mathcal{E} \leftarrow$  learning examples
2:  $d \leftarrow$  latent concepts' dimension
3:  $\mathcal{W}, \mathcal{V}, \mathcal{R} \leftarrow \emptyset$ 
4:  $\mathcal{R} \leftarrow \text{createLayer1Rules}(\mathcal{E}, d)$ 
5:  $\mathcal{W} \leftarrow \text{initWeights}(R)$ 
6:  $(\mathcal{F}, \mathcal{V}) \leftarrow \text{weightedFacts}(\mathcal{E}, R, \mathcal{W})$ 
7: while  $\neg \text{StoppingCriterion}$  do
8:    $\text{bestRule} \leftarrow \text{ruleLearning}(\mathcal{F}, \mathcal{V}, \mathcal{R})$ 
9:    $\text{bestRules} \leftarrow \text{predicateInvention}(\text{bestRule})$ 
10:   $\mathcal{R} \leftarrow \mathcal{R} \cup \text{bestRules}$ 
11:   $\mathcal{W} \leftarrow \text{trainWeights}(\mathcal{R}, \mathcal{E}, \mathcal{W})$ 
12:   $(\mathcal{F}, \mathcal{V}) \leftarrow \text{weightedFacts}(\mathcal{E}, \mathcal{R}, \mathcal{W})$ 
13: end while
14: return  $(\mathcal{R}, \mathcal{W})$ 

```

---

**3.2 Structure Learning Algorithm**

The structure learning algorithm (Algorithm 1) iteratively constructs LRNNs that have the structure described in the previous section. It alternates weight learning steps with rule learning steps<sup>1</sup>. In the weight learning steps, the algorithm uses stochastic gradient descent to minimise the squared loss of the LRNN by optimising the weights of the rules, as described in Sect. 2. In the rule learning steps, the algorithm fixes the weights of all rules which define latent predicates and it searches for some *good* rule  $R$ . This rule  $R$  should be such that the squared loss of the LRNN decreases after we add  $R$  to it and after we retrain the weights of all rules with non-latent head predicates. Next we describe this algorithm in detail.

The first step of the structure learning algorithm (lines 4–5) is the construction of the first level of the LRNN, which defines the unary predicates representing soft clusters of object properties, as described in Sect. 3.1.

After the first step, the algorithm repeats the following procedure for a given number of iterations or until no suitable rules can be found anymore. It fixes the weights of all rules defining latent predicates (line 6). Then it runs a beam search algorithm searching through the space of possible rules<sup>2</sup> (line 8). The scoring function which is used by the beam search algorithm is computed as follows. Given a rule  $R$ , the algorithm creates a copy of the current LRNN to which the given candidate rule  $R$  is added. It then optimises the log-loss of this new LRNN (which corresponds to maximum-likelihood estimation for logistic regression), training just the non-fixed weights, i.e. the weights of the rules with non-latent predicates in their heads. The score of the rule  $R$  is then defined to

<sup>1</sup> Variants of this strategy are employed by many structure learning algorithms in the context of statistical relational learning, e.g. [4, 5, 8].

<sup>2</sup> The space of rules is defined by two user-specified constraints: maximum rule length and maximum number of variables in a rule.

be the log-loss after training the non-fixed weights. The reason why we do not retrain all weights of the LRNN when checking score of a rule  $R$  are efficiency considerations because training the weights of the whole LRNN corresponds to training a deep neural network. After the beam search algorithm finishes, the rule  $R^*$  that it returned is added to the original LRNN.

Note that  $R^*$  contains one of the target predicates in its head. However, in addition to adding  $R^*$ , we also add a set of related rules that have latent predicates in their head (line 9), as follows. Here, we will assume for simplicity that all latent predicates have the same arity  $k$ , but the same method can still be used when the latent predicates are allowed to have different arities. Let  $i$  be the highest index such that  $R^*$  contains a latent predicate of the form  $\alpha_j^i$  (i.e. a latent predicate from layer  $i$ ) in its body, where we assume  $i = 1$  if  $R^*$  does not contain any latent predicates. Then for each latent predicate  $\alpha_j^{i+1}$  from the  $(i + 1)$ -th layer, the algorithm adds to the LRNN all rules which have  $\alpha_j^{i+1}(V_1, \dots, V_k)$  in the head and which can be obtained by unifying  $V_1, \dots, V_k$  with the variables in  $R^*$ . This process is illustrated in the following example.

*Example 2.* Revisiting the example of molecular datasets, let  $R^* = p(A, B) \leftarrow \text{bond}(A, B) \wedge \alpha_2^1(A) \wedge \alpha_5^2(B)$  and let  $k = 1$ . Then the algorithm will add the following latent-predicate rules:

$$\begin{array}{lll}
 w_{1,1}^3 : & \alpha_1^3(V_1) & \leftarrow \text{bond}(V_1, B) \wedge \alpha_2^1(V_1) \wedge \alpha_5^2(B) \\
 w_{1,2}^3 : & \alpha_1^3(V_1) & \leftarrow \text{bond}(A, V_1) \wedge \alpha_2^1(A) \wedge \alpha_5^2(V_1) \\
 w_{2,1}^3 : & \alpha_2^3(V_1) & \leftarrow \text{bond}(V_1, B) \wedge \alpha_2^1(V_1) \wedge \alpha_5^2(B) \\
 w_{2,2}^3 : & \alpha_2^3(V_1) & \leftarrow \text{bond}(A, V_1) \wedge \alpha_2^1(A) \wedge \alpha_5^2(V_1) \\
 \dots & \dots & \dots \\
 w_{d,1}^3 : & \alpha_d^3(V_1) & \leftarrow \text{bond}(V_1, B) \wedge \alpha_2^1(V_1) \wedge \alpha_5^2(B) \\
 w_{d,2}^3 : & \alpha_d^3(V_1) & \leftarrow \text{bond}(A, V_1) \wedge \alpha_2^1(A) \wedge \alpha_5^2(V_1)
 \end{array}$$

Note that the algorithm has to add the new rules to the layer 3 because  $R^*$  already contained predicates from the layer 2.

After the LRNN has been extended by all these rules obtained from  $R^*$ , the weights of all the rules, including those corresponding to latent predicates, are retrained using stochastic gradient descent (line 11). Note that typically there will be some latent predicates which are not used in any rules; their weights are not considered during training. Subsequently, the algorithm again fixes the weights of the rules corresponding to the latent predicates, and repeats the same process to find an additional rule. This is repeated until a given stopping condition is met.

## 4 Experiments

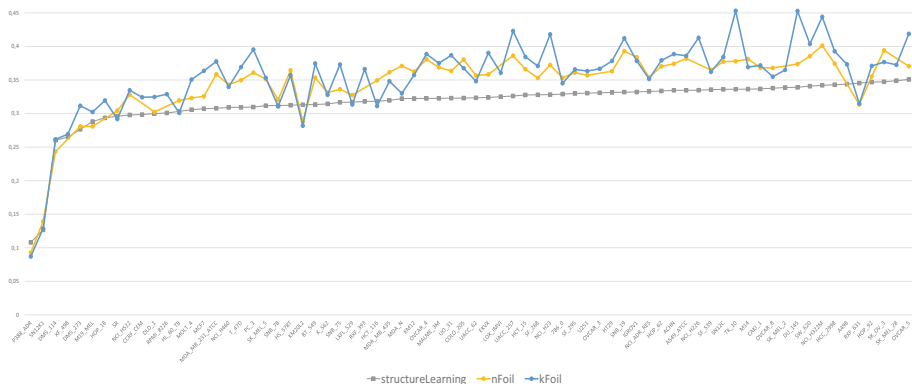
In this section we describe the results of experiments performed with the structure learning algorithm on a real-life molecular dataset and on a difficult artificial learning problem.

## 4.1 Molecular Datasets

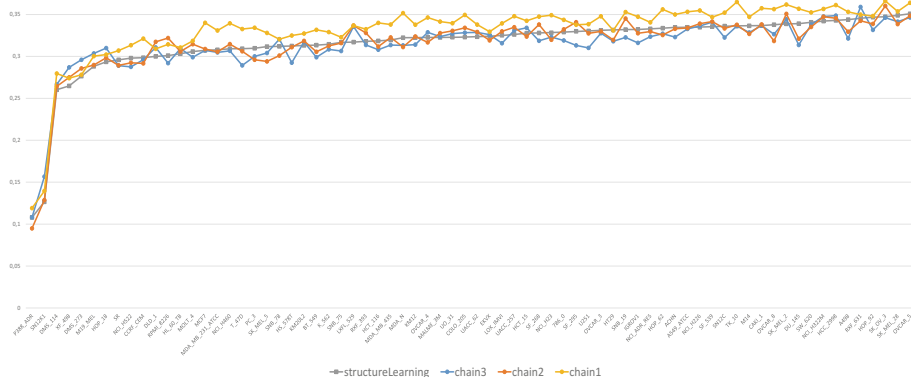
We performed experiments on 72 NCI datasets [13], each of which contains several thousands of molecules, labeled by their ability to inhibit the growth of different types of tumors. We compare the performance of the proposed LRNN structure learning method with the best previously published LRNNs, which contain large generic, yet manually constructed weighted rule sets [15]. For further comparison we include the relational learners kFOIL [10] and nFOIL [9], which respectively combine relational rule learning with support vector machines and with naive Bayes learning.

The results are shown in Figs. 2 and 3. The automatically learned LRNNs outperform both kFOIL and nFOIL in terms of predictive accuracy (measured using cross-validation). The learned LRNNs are also competitive with the manually constructed LRNNs from [15,16], although they do not outperform them. They are slightly worse than the largest of the manually constructed LRNNs, based on graph patterns with 3 vertices, enumerating all possible combinations of soft cluster types of the three atoms and soft cluster types of the two bonds connecting them. Figure 4 displays statistics of the learned LRNN rule sets. These statistics show that the structure learner turned out to produce quite complex LRNNs having multiple layers of invented latent predicates.

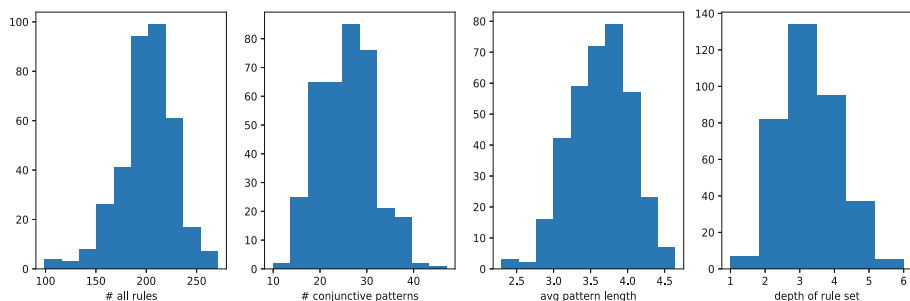
The weights of the rules defining the latent predicates in the first layer of the LRNN can be interpreted as coordinates of a vector-space embedding of the properties (atom types in our case). In Fig. 5, we plot the evolution of these embeddings as new rules are being added by the structure learning algorithm. The left panel of Fig. 5 displays the evolution of the embeddings of atom types after these have been pre-trained using an unsupervised method which was originally used for statistical predicate invention in [17]. The right panel of the same figure displays the evolution of the embeddings when starting from random initialization without any unsupervised pre-training. What can be seen from these



**Fig. 2.** Comparison of crossvalidated test errors of LRNNs produced by structure learning with nFoil and kFoil learners as baselines.



**Fig. 3.** Comparison of test errors of LRNNs produced automatically by structure learning with 3 handcrafted LRNNs with varying lengths of chain patterns from [15].

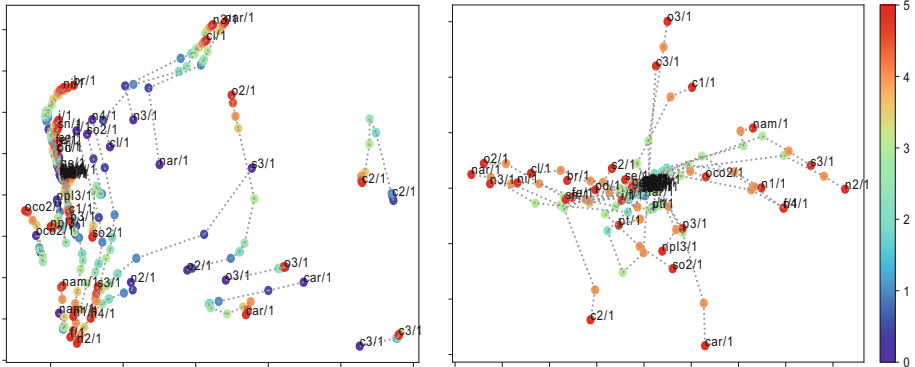


**Fig. 4.** Statistics of the learned LRNN rule sets from experiments with the 72 NCI datasets. We display (i) the number of rules (including zeroth layer soft clusters), (ii) the number of conjunctive rules (patterns) learned, (iii) the average length of these rules (patterns), and (iv) the overall number of layers (depth of template).

figures is how, as the model becomes more complex, the atom types start to make more visible clusters. Interestingly and perhaps somewhat against intuition, the use of the unsupervised pre-training seemed to consistently decrease predictive performance (we omit details due to limited space).

## 4.2 A Hard Artificial Problem

We consider the following variant of graph colorability, which can be seen as a relational generalization of the problem of learning the XOR function. For a graph, where each node may take on different “shades”  $\{sh_1 \dots sh_n\}$  of colors  $\{col_1 \dots col_m\}$  that are not observed (i.e. it is not given to which color each shade corresponds), the task is to learn to classify graphs that are correctly colored, i.e. where each edge in the graph connects two nodes of shades of different colors. In this problem, learning a correct representation of the colors (as sets of shades)



**Fig. 5.** PCA projection of evolution of atom embeddings during first 6 iterations (denoted by colors) of structure learning of a LRNN, with initialization based on unsupervised pre-training (left) and with completely random initialization (right). (Color figure online)

is completely decorrelated from the target, but to correctly learn to classify correctly colored graphs, we need to learn some such color concepts. An ideal learned LRNN correctly solving the problem could be very compact, for instance for 3 colors it might look like

$$\begin{aligned} w_1 : \text{notCorrectlyColored} &\leftarrow cl_0(X), \text{edge}(X, Y), cl_0(Y) \\ w_2 : \text{notCorrectlyColored} &\leftarrow cl_1(X), \text{edge}(X, Y), cl_1(Y) \\ w_3 : \text{notCorrectlyColored} &\leftarrow cl_2(X), \text{edge}(X, Y), cl_2(Y) \end{aligned}$$

together with rules defining the color concepts  $cl_0$ ,  $cl_1$  and  $cl_2$ .

Initial experiments with these artificial problems showed that the structure learning algorithm is able to learn appropriate LRNNs. The rule sets were typically different from the rule set shown above but they also encoded correct solutions that were comparably short. The performance results, shown in Table 1, suggest that the LRNN structure learning method is able to efficiently produce accurate and compact solutions without extensive hyper-tuning.

## 5 Related Work

LRNNs are related to many older works on using neural networks for relational learning such as [1] and more recent approaches such as [3, 14]. The structure learning strategy that we employ in the methods presented in this paper is in many respects similar to structure learning methods from statistical relational learning such as [4, 5, 8]. However, what clearly distinguishes it from all these previous SRL approaches is its ability to automatically induce hierarchies of latent concepts. In this respect, it is also related to meta-interpretive learning [11]. However, meta-interpretive learning is only applicable to the learning of



**Table 1.** Results of the structure learning algorithm (SL) on the graph coloring classification problem. Reported statistics are majority train error, training error, and ratio of cases with zero learning error. All problems were run 10 times with different random initialization seeds.

#colors-#shades	Majority error	Training error	% of perfect solutions
2c-1s	0.5	0.025	0.9
2c-2s	0.5	0.0	1
3c-1s	0.33	0.0	1
3c-2s	0.33	0.014	0.6
3c-3s	0.33	0.111	0.4
4c-1s	0.25	0.1375	0.0
4c-2s	0.25	0.160	0.0
4c-3s	0.25	0.129	0.1
4c-4s	0.25	0.044	0.1

crisp logic programs. The structure learning approach is also related to works on refining architectures of neural networks [6, 12]. However, from these it differs in its ability to handle relational data.

## 6 Conclusions and Future Work

In this paper we have introduced a method for learning the structure of LRNNs, capable of learning deep weighted rule sets with invented latent predicates. The predictive accuracies obtained by the learned LRNNs were competitive with results that we obtained in our previous work using manually constructed LRNNs. The method presented in this paper therefore has the potential to make LRNNs useful in domains where it would otherwise be difficult to come up with a rule set manually. It also makes the adoption of LRNNs by non-expert users more straightforward, as the proposed method can learn competitive LRNNs without requiring any user input (besides the dataset).


**Acknowledgements.** GŠ, MS and FŽ acknowledge support by project no. 17-26999S granted by the Czech Science Foundation. This work was done while OK was with Cardiff University and supported by a grant from the Leverhulme Trust (RPG-2014-164). SS is supported by ERC Starting Grant 637277. Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme “Projects of Large Research, Development, and Innovations Infrastructures”.

## References

1. Blockeel, H., Uwents, W.: Using neural networks for relational learning. In: ICML 2004 Workshop on Statistical Relational Learning and Its Connection to Other Fields, pp. 23–28 (2004)
2. Bottou, L.: Stochastic gradient descent tricks. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) *Neural Networks: Tricks of the Trade*. LNCS, vol. 7700, pp. 421–436. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-35289-8\\_25](https://doi.org/10.1007/978-3-642-35289-8_25)
3. Cohen, W.W.: TensorLog: a differentiable deductive database. arXiv preprint [arXiv:1605.06523](https://arxiv.org/abs/1605.06523) (2016)
4. Davis, J., Burnside, E., de Castro Dutra, I., Page, D., Costa, V.S.: An integrated approach to learning Bayesian networks of rules. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) *ECML 2005*. LNCS (LNAI), vol. 3720, pp. 84–95. Springer, Heidelberg (2005). [https://doi.org/10.1007/11564096\\_13](https://doi.org/10.1007/11564096_13)
5. Dinh, Q.T., Exbrayat, M., Vrain, C.: Generative structure learning for Markov logic networks based on graph of predicates. In: *IJCAI Proceedings of International Joint Conference on Artificial Intelligence*, vol. 22, p. 1249 (2011)
6. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture (1989)
7. Hájek, P.: *Metamathematics of Fuzzy Logic*, vol. 4. Springer, Dordrecht (1998). <https://doi.org/10.1007/978-94-011-5300-3>
8. Kok, S., Domingos, P.: Learning the structure of Markov logic networks. In: *Proceedings of the 22nd International Conference on Machine Learning*, pp. 441–448 (2005)
9. Landwehr, N., Kersting, K., Raedt, L.D.: Integrating naive bayes and FOIL. *J. Mach. Learn. Res.* **8**, 481–507 (2007)
10. Landwehr, N., Passerini, A., De Raedt, L., Frasconi, P.: kFOIL: learning simple relational kernels. In: *AAAI 2006: Proceedings of the 21st National Conference on Artificial Intelligence*, pp. 389–394. AAAI Press (2006)
11. Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Mach. Learn.* **100**(1), 49–73 (2015)
12. Opitz, D.W., Shavlik, J.W.: Heuristically expanding knowledge-based neural networks. In: *IJCAI*, pp. 1360–1365 (1993)
13. Ralaivola, L., Swamidass, S.J., Saigo, H., Baldi, P.: Graph kernels for chemical informatics. *Neural Netw.* **18**(8), 1093–1110 (2005)
14. Rocktäschel, T., Riedel, S.: Learning knowledge base inference with neural theorem provers. In: *NAACL Workshop on Automated Knowledge Base Construction (AKBC)* (2016)
15. Šourek, G., Aschenbrenner, V., Železný, F., Kuželka, O.: Lifted relational neural networks. In: *Proceedings of the NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches* (2015)
16. Šourek, G., Aschenbrenner, V., Železný, F., Kuželka, O.: Lifted relational neural networks. arXiv preprint (2015). <http://arxiv.org/abs/1508.05128>
17. Šourek, G., Manandhar, S., Železný, F., Schockaert, S., Kuželka, O.: Learning predictive categories using lifted relational neural networks. In: Cussens, J., Russo, A. (eds.) *ILP 2016*. LNCS (LNAI), vol. 10326, pp. 108–119. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63342-8\\_9](https://doi.org/10.1007/978-3-319-63342-8_9)



# Pruning Hypothesis Spaces Using Learned Domain Theories

Martin Svatoš<sup>1</sup> , Gustav Šourek<sup>1</sup>, Filip Železný<sup>1</sup>, Steven Schockaert<sup>2</sup>,  
and Ondřej Kuželka<sup>3</sup>

<sup>1</sup> Czech Technical University, Prague, Czech Republic  
{svatoma1,souregus,zelezny}@fel.cvut.cz

<sup>2</sup> School of CS and Informatics, Cardiff University, Cardiff, UK  
SchockaertS1@cardiff.ac.uk

<sup>3</sup> Department of Computer Science, KU Leuven, Leuven, Belgium  
ondrej.kuzelka@kuleuven.be

**Abstract.** We present a method to prune hypothesis spaces in the context of inductive logic programming. The main strategy of our method consists in removing hypotheses that are equivalent to already considered hypotheses. The distinguishing feature of our method is that we use learned domain theories to check for equivalence, in contrast to existing approaches which only prune isomorphic hypotheses. Specifically, we use such learned domain theories to saturate hypotheses and then check if these saturations are isomorphic. While conceptually simple, we experimentally show that the resulting pruning strategy can be surprisingly effective in reducing both computation time and memory consumption when searching for long clauses, compared to approaches that only consider isomorphism.

## 1 Introduction

A key challenge for inductive logic programming (ILP) algorithms (e.g. Progol [12]) is the fact that they typically have to search through large hypothesis spaces. Methods for pruning this search space have the potential to dramatically improve the quality of learned hypotheses and/or the runtime of the algorithms. One way of doing this is by filtering isomorphic hypotheses, which is the strategy used, for instance, in the relational pattern mining algorithm Farmr [14]. However, pruning isomorphic hypotheses is often not optimal, in the sense that it may be possible to prune hypotheses which are not isomorphic, but which are nonetheless equivalent in the considered domain. For example, the hypothesis that “if  $X$  is the father of  $Y$  then  $X$  and  $Y$  have the same last name” is equivalent to the hypothesis that “if  $X$  is male and a parent of  $Y$  then  $X$  and  $Y$  have the same last name”.

In this paper, we introduce a method which explicitly tries to prune equivalent hypotheses that are created during the hypothesis search, while still maintaining completeness<sup>1</sup>. One important challenge is that we need a quick way of

<sup>1</sup> As we show later in the paper, the completeness requirement disqualifies relative subsumption [15] as a candidate for such a pruning method.

testing whether a new hypothesis is equivalent to a previously considered one. To this end, we propose a saturation method which, given a first-order-logic clause, derives a longer *saturated* clause that is equivalent to it modulo a domain theory. This saturation method has the important property that two clauses are equivalent, given a domain theory, whenever their saturations are isomorphic. This means that we can use saturations to detect equivalent hypotheses as follows. We compute saturations of all hypotheses as they are being constructed, as well as certain invariants<sup>2</sup> of these saturations. Then we use the invariants to compute hashes for the saturated hypotheses, which allows us to use hash tables to efficiently narrow down the set of previously constructed hypotheses against which equivalence needs to be tested. In this way, we can avoid the need to explicitly compare new hypotheses with all previously constructed ones, which would clearly be infeasible in spaces with possibly millions of hypotheses. Note that this technique crucially relies on the use of saturations, and would not be possible with e.g. just a notion of relative subsumption modulo a background theory. To avoid the need for any prior domain knowledge, our method learns the required domain theories from the training data.

We experimentally show that our method can be orders of magnitude faster than methods which merely check for isomorphism, even when taking into account the time needed for learning domain theories.

## 2 Preliminaries

In this section, we first give an overview of the notations and terminology from first-order logic that will be used throughout the paper, after which Sect. 2.2 describes the considered learning setting.

### 2.1 First-Order Logic

We consider a function-free first-order language, which is built from a finite set of constants, variables, and predicates in the usual way. A term is a variable or a constant. An atom is a formula of the form  $p(t_1, \dots, t_n)$ , where  $p$  is an  $n$ -ary predicate symbol and  $t_1, \dots, t_n$  are terms. A literal is an atom or the negation of an atom. A clause  $A$  is a universally quantified disjunction of literals  $\forall x_1 \dots \forall x_n : \phi_1 \vee \dots \vee \phi_k$ , such that  $x_1, \dots, x_n$  are the only variables occurring in the literals  $\phi_1, \dots, \phi_k$ . For the ease of presentation, we will sometimes identify a clause  $A$  with the corresponding set of literals  $\{\phi_1, \dots, \phi_k\}$ . The set of variables occurring in a clause  $A$  is written as  $vars(A)$  and the set of all terms as  $terms(A)$ . For a clause  $A$ , we define the *sign flipping* operation as  $\tilde{A} \stackrel{def}{=} \bigvee_{l \in A} \tilde{l}$ , where  $\tilde{a} = \neg a$  and  $\neg \neg a = a$  for an atom  $a$ . In other words, the sign flipping operation simply replaces each literal by its negation.

A substitution  $\theta$  is a mapping from variables to terms. For a clause  $A$ , we write  $A\theta$  for the clause  $\{\phi\theta \mid \phi \in A\}$ , where  $\phi\theta$  is obtained by replacing each occurrence

<sup>2</sup> We use invariants based on a generalized version of Weisfeiler-Lehman procedure [22].

in  $\phi$  of a variable  $v$  by the corresponding term  $\theta(v)$ . If  $A$  and  $B$  are clauses then we say that  $A$   $\theta$ -subsumes  $B$  (denoted  $A \preceq_{\theta} B$ ) if and only if there is a substitution  $\theta$  such that  $A\theta \subseteq B$ . If  $A \preceq_{\theta} B$  and  $B \preceq_{\theta} A$ , we call  $A$  and  $B$   $\theta$ -equivalent (denoted  $A \approx_{\theta} B$ ). Note that the  $\approx_{\theta}$  relation is indeed an equivalence relation (i.e. it is reflexive, symmetric and transitive). Clauses  $A$  and  $B$  are said to be *isomorphic* (denoted  $A \approx_{iso} B$ ) if there exists an injective substitution  $\theta$  such that  $A\theta = B$ . Finally, we say that  $A$  OI-subsumes  $B$  (denoted  $A \preceq_{OI} B$  [6]) if there is an injective substitution such that  $A\theta \subseteq B$ . Note that  $A$  is isomorphic to  $B$  iff  $A \preceq_{OI} B$  and  $B \preceq_{OI} A$ .

*Example 1.* Let us consider the following four clauses:

$$\begin{aligned} C_1 &= p_1(A, B) \vee \neg p_2(A, B) \\ C_2 &= p_1(A, B) \vee \neg p_2(A, B) \vee \neg p_2(A, C) \\ C_3 &= p_1(X, Y) \vee \neg p_2(X, Y) \vee \neg p_2(X, Z) \\ C_4 &= p_1(A, B) \vee \neg p_3(A, B) \end{aligned}$$

Then we can easily verify that  $C_1 \approx_{\theta} C_2 \approx_{\theta} C_3$  (and thus also  $C_i \preceq_{\theta} C_j$  for  $i, j \in \{1, 2, 3\}$ ). We also have  $C_1 \not\approx_{iso} C_2$ ,  $C_1 \not\approx_{iso} C_3$ ,  $C_2 \approx_{iso} C_3$ , as well as  $C_i \not\preceq_{\theta} C_4$  and  $C_4 \not\preceq_{\theta} C_i$  for any  $i \in \{1, 2, 3\}$ . Finally we also have  $C_1 \preceq_{OI} C_i$  for  $i \in \{1, 2, 3\}$ ,  $C_2 \preceq_{OI} C_3$  and  $C_3 \preceq_{OI} C_2$ .

A literal is ground if it does not contain any variables. A grounding substitution is a substitution in which each variable is mapped to a constant. Clearly, if  $\theta$  is a grounding substitution, then for any literal  $\phi$  it holds that  $\phi\theta$  is ground. An interpretation  $\omega$  is defined as a set of ground literals. A clause  $A = \{\phi_1, \dots, \phi_n\}$  is satisfied by  $\omega$ , written  $\omega \models A$ , if for each grounding substitution  $\theta$ , it holds that  $\{\phi_1\theta, \dots, \phi_n\theta\} \cap \omega \neq \emptyset$ . In particular, note that a ground literal  $\phi$  is satisfied by  $\omega$  if  $\phi \in \omega$ . The satisfaction relation  $\models$  is extended to (sets of) propositional combinations of clauses in the usual way. If  $\omega \models T$ , for  $T$  a propositional combination of clauses, we say that  $\omega$  is a model of  $T$ . If  $T$  has at least one model, we say  $T$  is satisfiable. Finally, for two (propositional combinations of) clauses  $A$  and  $B$ , we write  $A \models B$  if every model of  $A$  is also a model of  $B$ . Note that if  $A \preceq_{\theta} B$  for clauses  $A$  and  $B$  then  $A \models B$ , but the converse does not hold in general.

Deciding  $\theta$ -subsumption between two clauses is an NP-complete problem. It is closely related to constraint satisfaction problems with finite domains and tabular constraints [4], conjunctive query containment [3] and homomorphism of relational structures. The formulation of  $\theta$ -subsumption as a constraint satisfaction problem has been exploited in the ILP literature for the development of fast  $\theta$ -subsumption algorithms [8, 11]. CSP solvers can also be used to check whether two clauses are isomorphic, by using the primal CSP encoding described in [11] together with an *alldifferent* constraint [7] over CSP variables representing logical variables. In practice, this approach to isomorphism checking can be further optimized by pre-computing a directed hypergraph variant of Weisfeiler-Lehman coloring [22] (where terms play the role of hyper-vertices and literals the role of

directed hyper-edges) and by enriching the respective clauses by unary literals with predicates representing the labels obtained by the Weisfeiler-Lehman procedure, which helps the CSP solver to reduce its search space.

## 2.2 Learning Setting

In this paper we will work in the classical setting of *learning from interpretations* [16]. In this setting, examples are interpretations and hypotheses are clausal theories (i.e. conjunctions of clauses). An example  $e$  is said to be *covered* by a hypothesis  $H$  if  $e \models H$  (i.e.  $e$  is *covered* by  $H$  if it is a model of  $H$ ). Given a set of positive examples  $\mathcal{E}^+$  and negative examples  $\mathcal{E}^-$ , the training task is then to find a hypothesis  $H$  from some class of hypotheses  $\mathcal{H}$  which optimizes a given scoring function (e.g. training error). For the ease of presentation, we will restrict ourselves to classes  $\mathcal{H}$  of hypotheses in the form of clausal theories without constants, as constants can be emulated by unary predicates (since we do not consider functions).

The covering relation  $e \models H$  can be checked using a  $\theta$ -subsumption solver as follows. Each hypothesis  $H$  can be written as a conjunction of clauses  $H = C_1 \wedge \dots \wedge C_n$ . Clearly,  $e \not\models H$  if there is an  $i$  in  $\{1, \dots, n\}$  such that  $e \models \neg C_i$ , which holds precisely when  $C_i \preceq_{\theta} \neg(\bigwedge e)$ .

*Example 2.* Let us consider the following example, inspired by the Michalski’s East-West trains datasets [20]:

$$\begin{aligned} e = \{ & eastBound(car1), hasCar(car1), hasLoad(car1, load1), boxShape(load1), \\ & \neg eastBound(load1), \neg hasCar(load1), \neg hasLoad(load1, car1), \\ & \neg hasLoad(load1, load1), \neg hasLoad(car1, car1), \neg boxShape(car1) \} \end{aligned}$$

and two hypotheses  $H_1$  and  $H_2$

$$\begin{aligned} H_1 &= eastBound(C) \vee \neg hasLoad(C, L) \vee \neg boxShape(L) \\ H_2 &= \neg eastBound(C) \vee \neg hasLoad(C, L) \end{aligned}$$

To check if  $e \models H_i$ ,  $i = 1, 2$ , using a  $\theta$ -subsumption solver, we construct

$$\begin{aligned} \neg(\bigwedge e) &= \neg eastBound(car1) \vee \neg hasCar(car1) \vee \neg hasLoad(car1, load1) \\ &\vee boxShape(load1) \vee eastBound(load1) \vee hasCar(load1) \\ &\vee hasLoad(load1, car1) \vee hasLoad(load1, load1) \vee hasLoad(car1, car1) \\ &\vee boxShape(car1) \end{aligned}$$

It is then easy to check that  $H_1 \not\preceq_{\theta} \neg(\bigwedge e)$  and  $H_2 \preceq_{\theta} \neg(\bigwedge e)$ , from which it follows that  $e \models H_1$  and  $e \not\models H_2$ .

In practice, when using a  $\theta$ -subsumption solver to check  $C_i \preceq_{\theta} \neg(\bigwedge e)$ , it is usually beneficial to flip the signs of all the literals, i.e. to instead check  $\bar{C}_i \preceq_{\theta} \bigvee e$ , which is clearly equivalent. This is because  $\theta$ -subsumption solvers

often represent negative literals in interpretations implicitly to avoid excessive memory consumption<sup>3</sup>, relying on the assumption that most predicates in real-life datasets are sparse.

### 2.3 Theorem Proving Using SAT Solvers

The methods described in this paper will require access to an efficient theorem prover for clausal theories. Since we restrict ourselves to function-free theories without equality, we can rely on a simple theorem-proving procedure based on propositionalization, which is a consequence of the following well-known result<sup>4</sup> [13].

**Theorem 1 (Herbrand’s Theorem).** *Let  $\mathcal{L}$  be a first-order language without equality and with at least one constant symbol, and let  $\mathcal{T}$  be a set of clauses. Then  $\mathcal{T}$  is unsatisfiable iff there exists some finite set  $\mathcal{T}_0$  of  $\mathcal{L}$ -ground instances of clauses from  $\mathcal{T}$  that is unsatisfiable.*

Here  $A\theta$  is called an  $\mathcal{L}$ -ground instance of a clause  $A$  if  $\theta$  is a grounding substitution that maps each variable occurring in  $A$  to a constant from the language  $\mathcal{L}$ .

In particular, to decide if  $\mathcal{T} \models C$  holds, where  $T$  is a set of clauses and  $C$  is a clause (without constants and function symbols), we need to check if  $\mathcal{T} \wedge \neg C$  is unsatisfiable. Since Skolemization preserves satisfiability, this is the case iff  $\mathcal{T} \wedge \neg C_{Sk}$  is unsatisfiable, where  $\neg C_{Sk}$  is obtained from  $\neg C$  using Skolemization. Let us now consider the restriction  $\mathcal{L}_{Sk}$  of the considered first-order language  $\mathcal{L}$  to the constants appearing in  $C_{Sk}$ , or to some auxiliary constant  $s_0$  if there are no constants in  $C_{Sk}$ . From Herbrand’s theorem, we know that  $\mathcal{T} \wedge \neg C_{Sk}$  is unsatisfiable in  $\mathcal{L}_{Sk}$  iff the grounding of this formula w.r.t. the constants from  $\mathcal{L}_{Sk}$  is satisfiable, which we can efficiently check using a SAT solver. Moreover, it is easy to see that  $\mathcal{T} \wedge \neg C_{Sk}$  is unsatisfiable in  $\mathcal{L}_{Sk}$  iff this formula is unsatisfiable in  $\mathcal{L}$ <sup>5</sup>.

In practice, it is not always necessary to completely ground the formula  $\mathcal{T} \wedge \neg C_{Sk}$ . It is often beneficial to use an incremental grounding strategy similar to cutting plane inference in Markov logic [19]. To check if a clausal theory  $\mathcal{T}$  is satisfiable, this method proceeds as follows.

**Step 0:** start with an empty Herbrand interpretation  $\mathcal{H}$  and an empty set of ground formulas  $\mathcal{G}$ .

<sup>3</sup> This is true for the  $\theta$ -subsumption solver based on [8] which we use in our implementation.

<sup>4</sup> The formulation of Herbrand’s theorem used here is taken from notes by Cook and Pitassi: <http://www.cs.toronto.edu/~toni/Courses/438/Mynotes/page39.pdf>.

<sup>5</sup> Indeed, if  $\mathcal{T} \wedge \neg C_{Sk}$  is unsatisfiable in  $\mathcal{L}$ , then there is a set of corresponding  $\mathcal{L}$ -ground instances of clauses that are unsatisfiable. If we replace each constant appearing in these ground clauses which does not appear in  $C_{Sk}$  by an arbitrary constant that does appear in  $C_{Sk}$ , then the resulting set of ground clauses must still be inconsistent, since  $T$  does not contain any constants and there is no equality in the language, meaning that  $\mathcal{T} \wedge \neg C_{Sk}$  cannot be satisfiable in  $\mathcal{L}_{Sk}$ .

- Step 1:** check which groundings of the formulas in  $\mathcal{T}$  are not satisfied by  $\mathcal{H}$  (e.g. using a CSP solver). If there are no such groundings, the algorithm returns  $\mathcal{H}$ , which is a model of  $\mathcal{T}$ . Otherwise the groundings are added to  $\mathcal{G}$ .
- Step 2:** use a SAT solver to find a model of  $\mathcal{G}$ . If  $\mathcal{G}$  does not have any model then  $\mathcal{T}$  is unsatisfiable and the method finishes. Otherwise replace  $\mathcal{H}$  by this model and go back to Step 1.

### 3 Pruning Hypothesis Spaces Using Domain Theories

In this section we show how domain theories can be used to prune the search space of ILP systems. Let us start with two motivating examples.

*Example 3.* Let us consider the following two hypotheses for some target concept  $x$ :

$$H_1 = x(A) \vee \neg animal(A) \vee \neg cod(A)$$

$$H_2 = x(A) \vee \neg fish(A) \vee \neg cod(A)$$

Intuitively, these two hypotheses are equivalent since every *cod* is a *fish* and every *fish* is an *animal*. Yet ILP systems would need to consider both of these hypotheses separately because  $H_1$  and  $H_2$  are not isomorphic, they are not  $\theta$ -equivalent and neither of them  $\theta$ -subsumes the other.

*Example 4.* Problems with redundant hypotheses abound in datasets of molecules, which are widespread in the ILP literature. For instance, consider the following two hypotheses:

$$H_1 = x(A) \vee \neg carb(A) \vee \neg bond(A, B) \vee \neg bond(B, C) \vee \neg hydro(C)$$

$$H_2 = x(A) \vee \neg carb(A) \vee \neg bond(A, B) \vee \neg bond(C, B) \vee \neg hydro(C)$$

These two hypotheses intuitively represent the same molecular structures (a carbon and a hydrogen both connected to the same atom of unspecified type). Again, however, their equivalence cannot be detected without the domain knowledge that bonds in molecular datasets are symmetric<sup>6</sup>.

In the remainder of this section we will describe how background knowledge can be used to detect equivalent hypotheses. First, we introduce the notion of  *saturations*  of clauses in Sect. 3.1. Subsequently, in Sect. 3.2 we show why pruning hypotheses based on these saturations does not hurt the completeness of a refinement operator. In Sect. 3.3, we then explain how these saturations can be used to efficiently prune search spaces of ILP algorithms. In Sect. 3.4 we describe a simple method for learning domain theories from the given training data. Finally, in Sect. 3.5 we show why using relative subsumption is not sufficient.

<sup>6</sup> In the physical world, bonds do not necessarily have to be symmetric, e.g. there is an obvious asymmetry in polar bonds. However, it is a common simplification in data mining on molecular datasets to assume that bonds are symmetric.



### 3.1 Saturations

The main technical ingredient of the proposed method is the following notion of *saturation*.

**Definition 1 (Saturation of a clause).** *Let  $\mathcal{B}$  be a clausal theory and  $C$  a clause (without constants or function symbols). If  $\mathcal{B} \not\models C$ , we define the saturation of  $C$  w.r.t.  $\mathcal{B}$  to be the maximal clause  $C'$  satisfying: (i)  $\text{vars}(C') = \text{vars}(C)$  and (ii)  $\mathcal{B} \wedge C'\theta \models C\theta$  for any injective grounding substitution  $\theta$ . If  $\mathcal{B} \models C$ , we define the saturation of  $C$  w.r.t.  $\mathcal{B}$  to be  $\mathbf{T}$ , where  $\mathbf{T}$  denotes tautology.*

When  $\mathcal{B}$  is clear from the context, we will simply refer to  $C'$  as the saturation of  $C$ .

Definition 1 naturally leads to a straightforward procedure for computing the saturation of a given clause. Let  $\mathcal{P} = \{l_1, l_2, \dots, l_n\}$  be the set of all literals which can be constructed using variables from  $C$  and predicate symbols from  $\mathcal{B}$  and  $C$ . Let  $\theta$  be an arbitrary injective grounding substitution; note that we can indeed take  $\theta$  to be arbitrary because  $\mathcal{B}$  and  $C$  do not contain constants. If  $\mathcal{B} \not\models C$ , the saturation of  $C$  is given by the following clause:

$$\bigvee \{l \in \mathcal{P} : \mathcal{B} \models \neg l\theta \vee C\theta\} \quad (1)$$

This means in particular that we can straightforwardly use the SAT based theorem proving method from Sect. 2.3 to compute saturations. The fact that (1) correctly characterizes the saturation can be seen as follows. If  $C'$  is the saturation of  $C$  then  $\mathcal{B} \wedge C'\theta \models C\theta$  by definition, which is equivalent to  $\mathcal{B} \wedge \neg(C\theta) \models \neg(C'\theta)$ . We have  $\neg(C'\theta) = \bigwedge \{\tilde{l}\theta : \mathcal{B} \wedge \neg(C\theta) \models \tilde{l}\theta\} = \bigwedge \{\tilde{l}\theta : \mathcal{B} \wedge l\theta \models C\theta\}$ , and thus  $C'\theta = \bigvee \{l\theta : \mathcal{B} \wedge l\theta \models C\theta\}$ . Finally, since  $\theta$  is injective, we have<sup>7</sup>  $C' = (C'\theta)\theta^{-1} = \bigvee \{l : \mathcal{B} \wedge l\theta \models C\theta\}$ .

*Example 5.* Let us consider the following theory

$$\mathcal{B} = \{\neg \text{friends}(X, Y) \vee \text{friends}(Y, X)\}$$

which expresses the fact that friendship is a symmetric relation and a clause

$$C = \neg \text{friends}(X, Y) \vee \text{happy}(X).$$

To find the saturation of this clause, we first need a suitable injective substitution  $\theta$ ; let us take  $\theta = \{X \mapsto c_1, Y \mapsto c_2\}$ . Then we have

$$\begin{aligned} \mathcal{B} \cup \neg(C\theta) &= \mathcal{B} \cup \{\text{friends}(c_1, c_2) \wedge \neg \text{happy}(c_1)\} \\ &\models \text{friends}(c_1, c_2) \wedge \text{friends}(c_2, c_1) \wedge \neg \text{happy}(c_1), \end{aligned}$$

After negating the latter formula and inverting the substitution (noting that it is injective) we get the following saturation:

$$C' = \neg \text{friends}(X, Y) \vee \neg \text{friends}(Y, X) \vee \text{happy}(X).$$

<sup>7</sup> Note that we are slightly abusing notation here, as  $\theta^{-1}$  is not a substitution.

Now, let us consider another clause  $D = \neg\text{friends}(X, Y) \vee \text{happy}(Y)$ . This clause is not isomorphic to  $C$ . However, it is easy to see that its saturation

$$D' = \neg\text{friends}(X, Y) \vee \neg\text{friends}(Y, X) \vee \text{happy}(Y)$$

is isomorphic to the saturation  $C'$  of  $C$ .

The next proposition will become important later in the paper as it will allow us to replace clauses by their saturations when learning from interpretations.

**Proposition 1.** *If  $C'$  is a saturation of  $C$  w.r.t.  $\mathcal{B}$  then  $\mathcal{B} \wedge C' \models C$ .*

*Proof.* We have  $\mathcal{B} \wedge C' \models C$  iff  $\mathcal{B} \wedge C' \wedge \neg C$  is unsatisfiable. Skolemizing  $\neg C$ , this is equivalent to  $\mathcal{B} \wedge C' \wedge \neg(C\theta_{Sk})$  being unsatisfiable, where  $\theta_{Sk}$  is a substitution representing the Skolemization. As in Sect. 2.3, we find that the satisfiability of  $\mathcal{B} \wedge C' \wedge \neg(C\theta_{Sk})$  is also equivalent to the satisfiability of the grounding of  $\mathcal{B} \wedge C' \wedge \neg(C\theta_{Sk})$  w.r.t. the Skolem constants introduced by  $\theta_{Sk}$ . In particular, this grounding must contain the ground clause  $C'\theta_{Sk}$ . From the definition of saturation, we have that  $\mathcal{B} \wedge C'\theta_{Sk} \wedge \neg(C\theta_{Sk}) \models \mathbf{F}$ , where  $\mathbf{F}$  denotes falsity (noting that  $\theta_{Sk}$  is injective). It follows that  $\mathcal{B} \wedge C' \wedge \neg C \models \mathbf{F}$ , and thus also  $\mathcal{B} \wedge C' \models C$ .  $\square$

The next proposition shows that saturations cover the same examples as the clauses from which they were obtained, when  $\mathcal{B}$  is a domain theory that is valid for all examples in the dataset.

**Proposition 2.** *Let  $\mathcal{B}$  be a clausal theory such that for all examples  $e$  from a given dataset it holds that  $e \models \mathcal{B}$ . Let  $C$  be a clause and let  $C'$  be its saturation w.r.t.  $\mathcal{B}$ . Then for any example  $e$  from the dataset we have  $(e \models C) \Leftrightarrow (e \models C')$ .*

*Proof.* From the characterization of saturation in (1), it straightforwardly follows that  $C \models C'$ , hence  $e \models C$  implies  $e \models C'$ . Conversely, if  $e \models C'$ , then we have  $e \models \mathcal{B} \wedge C'$ , since we assumed that  $e \models \mathcal{B}$ . Since we furthermore know from Proposition 1 that  $\mathcal{B} \wedge C' \models C$ , it follows that  $e \models C$ .

Finally, we define *positive* and *negative saturations*, which only add positive or negative literals to clauses. Among others, this will be useful in settings where we are only learning Horn clauses.

**Definition 2.** *A positive (resp. negative) saturation of  $C$  is defined as  $C'' = C \cup \{l \in C' : l \text{ is a positive (resp. negative) literal}\}$  where  $C'$  is a saturation of  $C$ .*

Propositions 1 and 2 are also valid for positive or negative saturations; their proofs can be straightforwardly adapted. When computing the positive (resp. negative) saturation, we can restrict the set  $\mathcal{P}$  of candidate literals to the positive (resp. negative) ones. This can speed up the computation of saturations significantly.

### 3.2 Searching the Space of Saturations

In this section we show how saturations can be used together with refinement operators to search the space of clauses ordered by OI-subsumption<sup>8</sup>. Specifically, we show that if we have a refinement operator that can completely generate some set of clauses then we can use the same refinement operator, in combination with a procedure for computing saturations, to generate the set of all saturations of the considered set of clauses. Since this set of saturations is typically smaller than the complete set of clauses (as many clauses can lead to the same saturated clauses), this is already beneficial for reducing the size of the hypothesis space. In Sect. 3.3, we show that it also allows us to very quickly prune equivalent clauses. First we give a definition of *refinement operator* [21].

**Definition 3 (Refinement operator).** *Let  $\mathcal{L}$  be a first-order language. A refinement operator<sup>9</sup> on the set  $\mathcal{C}$  of all  $\mathcal{L}$ -clauses is a function  $\rho : \mathcal{C} \rightarrow 2^{\mathcal{C}}$  such that for any  $C \in \mathcal{C}$  and any  $D \in \rho(C)$  it holds  $C \preceq_{OI} D$ . A refinement operator  $\rho$  is complete if for any two clauses  $C$  and  $D$  such that  $C \preceq_{OI} D$ , a clause  $E$  isomorphic to  $D$  ( $D \approx_{iso} E$ ) can be obtained from  $C$  by repeated application of the refinement operator (i.e.  $E \in \rho(\rho(\dots\rho(C)\dots))$ ).*

Most works define refinement operators w.r.t.  $\theta$ -subsumption instead of OI-subsumption [21]. We need the restriction to OI-subsumption as a technical condition for Proposition 3 below. It should be noted, however, that our results remain valid for many refinement operators that are not specifically based on OI-subsumption, including all refinement operators that only add new literals to clauses. Also note that we do not use OI-subsumption as a covering operator but only to structure the space of hypotheses. Therefore there is no loss in what hypotheses can be learnt.

The next definition formally introduces the combination of refinement operators and saturations.

**Definition 4 (Saturated refinement operator).** *Let  $\mathcal{L}$  be a first-order language. Let  $\rho$  be a refinement operator on the set  $\mathcal{C}$  of all  $\mathcal{L}$ -clauses containing at most  $n$  variables. Let  $\mathcal{B}$  be a clausal theory. Let  $\sigma_{\mathcal{B}} : \mathcal{C} \rightarrow \mathcal{C}$  be a function that maps a clause  $C$  to its saturation  $C'$  w.r.t.  $\mathcal{B}$ . Then the function  $\rho_{\mathcal{B}} = \sigma_{\mathcal{B}} \circ \rho$  is called the saturation of  $\rho$  w.r.t.  $\mathcal{B}$ .*

Clearly, the saturation of a refinement operator w.r.t. some clausal theory  $\mathcal{B}$  is a refinement operator as well. However, it can be the case that  $\rho$  is complete whereas its saturation is not. As we will show next, this is not a problem for completeness w.r.t. the given theory  $\mathcal{B}$  in the sense that saturations of all clauses from the given class  $\mathcal{C}$  are guaranteed to be eventually constructed by the combined operator, when  $\rho$  is a complete refinement operator.

<sup>8</sup> Note that we only use OI-subsumption to partially order the constructed hypotheses, not to check the entailment relation.

<sup>9</sup> What we call refinement operator in this paper is often called downward refinement operator. Since we only consider downward refinement operators in this paper, we omit the word downward.

**Proposition 3.** *Let  $\mathcal{L}$  be a first-order language. Let  $\rho$  be a complete refinement operator on the set of  $\mathcal{L}$ -clauses  $\mathcal{C}$ ,  $\mathcal{B}$  be clausal theory,  $\sigma_{\mathcal{B}}$  a function that maps a clause  $C$  to its saturation  $C'$  w.r.t.  $\mathcal{B}$  and let  $\rho_{\mathcal{B}}$  be the saturation of  $\rho$  w.r.t.  $\mathcal{B}$ . Let  $C \in \mathcal{C}$  be a clause,  $\mathcal{S}_C$  and let  $\mathcal{S}_C^{\mathcal{B}}$  be the sets of clauses that can be obtained from  $C$  by repeated application of  $\rho$  and  $\rho_{\mathcal{B}}$ , respectively. Then for any clause  $D \in \mathcal{S}_C$  there is a clause  $D' \in \mathcal{S}_C^{\mathcal{B}}$  such that  $\sigma_{\mathcal{B}}(D) \approx_{iso} D'$ .*

*Proof.* We first note that if  $A \preceq_{OI} B$  then  $\sigma_{\mathcal{B}}(A) \preceq_{OI} \sigma_{\mathcal{B}}(B)$  (assuming an extended definition of OI-subsumption such that  $A \preceq_{OI} \mathbf{T}$  for any  $A$ ), which follows from the monotonicity of the entailment relation  $\models$ . Let us define  $\mathcal{X} = \{\sigma_{\mathcal{B}}(A) \mid A \in \mathcal{S}_C\}$ . Note that  $\mathcal{X}$  and  $\mathcal{S}_C^{\mathcal{B}}$  are not defined in the same way ( $\mathcal{X}$  is the set of saturations of clauses in  $\mathcal{S}_C$  whereas  $\mathcal{S}_C^{\mathcal{B}}$  is the set of clauses that can be obtained by the saturated refinement operator  $\rho_{\mathcal{B}}$  from the clause  $C$ ). We need to show that these two sets are *equivalent*. Clearly,  $\mathcal{S}_C^{\mathcal{B}} \subseteq \mathcal{X}$ . To show the other direction, let us assume (for contradiction) that there is a clause  $X \in \mathcal{X}$  for which there is no clause  $Y \in \mathcal{S}_C^{\mathcal{B}}$  which is isomorphic to  $X$ . Let us assume that  $X$  is a minimal clause with this property, meaning that for any clause  $X'$  contained in the set  $\mathcal{Z}_X = \{Z \in \mathcal{X} \mid Z \preceq_{OI} X \wedge X \not\approx_{iso} Z\}$  there is a clause  $Y' \in \mathcal{S}_C^{\mathcal{B}}$  which is isomorphic to  $X'$ . Clearly, if there is one such clause  $X$  then there is also a minimal one which follows from the fact that all the considered clauses are finite and  $\preceq_{OI}$  is a partial order. Let us take a clause  $X' \in \mathcal{Z}_X$  which is maximal<sup>10</sup> w.r.t. the ordering induced by  $\preceq_{OI}$  and let  $Y'$  be the respective isomorphic clause from  $\mathcal{S}_C^{\mathcal{B}}$ . Then  $\rho(Y')$  must contain a clause  $Y''$ ,  $Y' \not\approx_{iso} Y''$ , that OI-subsumes  $X$ , which follows from completeness of the refinement operator  $\rho$ . However, then  $\sigma_{\mathcal{B}}(Y'')$  must be contained in  $\mathcal{S}_C^{\mathcal{B}}$ . It must also hold that  $\sigma_{\mathcal{B}}(Y'') \preceq_{OI} \sigma_{\mathcal{B}}(X) = X$ . Here,  $\sigma_{\mathcal{B}}(Y'') \preceq_{OI} \sigma_{\mathcal{B}}(X)$  follows from the already mentioned observation that if  $A \preceq_{OI} B$  then  $\sigma_{\mathcal{B}}(A) \preceq_{OI} \sigma_{\mathcal{B}}(B)$ , and the equality  $\sigma_{\mathcal{B}}(X) = X$  follows from the idempotence of  $\sigma_{\mathcal{B}}$ , noting that  $X$  is already a saturation of some clause. However, this is a contradiction with the maximality of  $X'$  and the corresponding  $Y'$ .  $\square$

### 3.3 Pruning Isomorphic Saturations

When searching the space of clauses or, in particular, saturations of clauses, we should avoid searching through isomorphic clauses. It is easy to see that the sets of clauses generated by a (saturated) complete refinement operator  $\rho$  from two isomorphic clauses  $C$  and  $C'$  will contain clauses that are isomorphic (i.e. for any clause in the first set there will be an isomorphic clause in the second set and vice versa). Therefore it is safe to prune isomorphic clauses during the search.

When searching through the hypothesis space of clauses, most ILP algorithms maintain some queue of candidate clauses. This is the case, for instance, in algorithms based on best-first search (Progol, Aleph [12]). Other algorithms, e.g. those based on level-wise search, maintain similar data structures (e.g. Warmr [5]).

<sup>10</sup> If we ordered the set of clauses by  $\theta$ -subsumption instead of OI-subsumption then there would not have to exist a maximal clause with this property.

Many of the clauses that are stored in such queues or similar data structures will be equivalent, even if they are not isomorphic. Existing methods, even if they were removing isomorphic clauses during search<sup>11</sup>, have to consider each of these equivalent clauses separately, which may greatly affect their performance. This is where using saturations of clauses w.r.t. some background knowledge is most useful because it can replace the different implicitly equivalent clauses by their saturation.

In theory, one could try to test isomorphism of all pairs of clauses currently in the queue data structures. However, this would be prohibitively slow in most practical cases. To efficiently detect equivalences by checking isomorphism of saturations, we replace the queue data structure (or a similar data structure used by the given algorithm) by a data structure that is based on hash tables. When a new hypothesis  $H$  is constructed by the algorithm, we first compute its saturation  $H'$ . Then, we check whether the modified queue data structure already contains a clause that is isomorphic to  $H'$ . To efficiently check this, we use a straightforward generalization of the Weisfeiler-Lehman labeling procedure [22]. We then only need to check whether two clauses are isomorphic if they have the same hash value. We similarly check whether  $H'$  is isomorphic to a clause in the so-called closed set of previously processed hypotheses. If  $H'$  is neither isomorphic to a clause in the queue nor to a clause in the closed set, it is added to the queue.

*Example 6.* Let us again consider the two clauses from Example 3:  $H_1 = x(A) \vee \neg animal(A) \vee \neg cod(A)$  and  $H_2 = x(A) \vee \neg fish(A) \vee \neg cod(A)$ . Suppose that the theory  $\mathcal{B}$  encodes the taxonomy of animals and contains the rules  $\neg cod(X) \vee fish(X)$  and  $\neg fish(X) \vee animal(X)$ . Computing the saturations of  $H_1$  and  $H_2$ , we obtain  $H'_1 = x(A) \vee \neg animal(A) \vee \neg cod(A) \vee \neg fish(A)$  and  $H'_2 = x(A) \vee \neg fish(A) \vee \neg cod(A) \vee \neg animal(A)$ , which are isomorphic. Therefore both of them can be replaced by the same saturations while the corresponding algorithm keeps searching the hypothesis space.

Similarly as shown above for the two clauses from Example 3, saturations could be used to detect equivalence of the two clauses from Example 4 w.r.t. the corresponding background knowledge theory  $\mathcal{B}$ .

In addition to equivalence testing, saturations can be used to filter trivial hypotheses, i.e. hypotheses covering every example, without explicitly computing their coverage on the dataset (which would be very costly on large datasets). We illustrate this use of saturations in the next example.

*Example 7.* Consider a domain theory  $\mathcal{B} = \neg professor(X) \vee \neg student(X)$  which states that no one can be both a student and a professor. Let us also consider a hypothesis  $H = employee(X) \vee \neg professor(X) \vee \neg student(X)$ . If the domain theory  $\mathcal{B}$  is correct,  $H$  should cover all examples from the dataset and is thus trivial. Accordingly, the saturation of  $H$  contains every literal, and is in particular equivalent to  $\mathbf{T}$ .

<sup>11</sup> For instance, Farmr [14] or RelF [9] remove isomorphic clauses (or conjunctive patterns), but many existing ILP systems do not attempt removing isomorphic clauses.

### 3.4 Learning Domain Theories for Pruning

The domain theories that we want to use for pruning hypothesis spaces can be learned from the given training dataset. Every clause  $C$  in such a learned domain theory should satisfy  $e \models C$  for all examples  $e$  in the dataset. We construct such theories using a level-wise search procedure, starting with an empty domain theory. The level-wise procedure maintains a list of candidate clauses (modulo isomorphism) with  $i$  literals. If a clause  $C$  in the list of candidate clauses covers all examples (i.e.  $e \models C$  for all  $e$  from the dataset) then it is removed from the list and if there is no clause in the domain theory which  $\theta$ -subsumes  $C$ , then  $C$  is also added to the domain theory. Each of the remaining clauses in the list, i.e. those which do not cover all examples in the dataset, are then extended in all possible ways by the addition of a literal. This is repeated until a threshold on the maximum number of literals is reached. The covering of examples by the candidate clauses is checked using  $\theta$ -subsumption as outlined in Sect. 2.

It is worth pointing out that if we restrict the domain theories, e.g. to contain only clauses of length at most 2 or only Horn clauses, the saturation process will be guaranteed to run in polynomial time (which follows from the polynomial-time solvability of 2-SAT and Horn-SAT).

### 3.5 Why Relative Subsumption is Not Sufficient

Although the motivation behind relative subsumption [15] is similar to ours, relative subsumption has two main disadvantages that basically disqualify it for the purpose of pruning the hypothesis space. The first problem is that pruning hypotheses that are equivalent w.r.t. relative subsumption may not guarantee completeness of the search. This is the same issue as with pruning based on plain  $\theta$ -subsumption which, unlike pruning based on isomorphism, may lead to incompleteness of the search. Note that this is already the case in the more restricted setting of graph mining under homomorphism [18]. The second issue with relative subsumption is that it would need to be tested for all pairs of candidate hypotheses, whereas the pruning based on saturations and isomorphism testing allows us to use the more efficient hashing strategy based on the Weisfeiler-Lehman procedure.

## 4 Experiments

In this section we evaluate the usefulness of the proposed pruning method on real datasets. We test it inside an exhaustive feature construction algorithm which we then evaluate on a standard molecular dataset KM20L2 from the NCI GI 50 dataset collection [17]. This dataset contains 1207 examples (molecules) and 64404 facts.

## 4.1 Methodology and Implementation

The evaluated feature construction method is a simple level-wise algorithm which works similarly to the Warmr frequent pattern mining algorithm [5]. It takes two parameters: maximum depth  $d$  and maximum number of covered examples  $t$  (also called “maximum frequency”). It returns all connected<sup>12</sup> clauses which can be obtained by saturating clauses containing at most  $d$  literals, and which cover at most  $t$  examples. Unlike in frequent conjunctive pattern mining where minimum frequency constraints are natural, when mining in the setting of learning from interpretations, the analogue of the minimum frequency is the maximum frequency constraint<sup>13</sup>.

The level-wise algorithm expects as input a list of interpretations (examples) and the parameters  $t$  and  $d > 0$ . It proceeds as follows:

- Step 0:** set  $i := 0$  and  $L_0 := \{\square\}$  where  $\square$  denotes the empty clause.
- Step 1:** construct a set  $L_{i+1}$  by extending each clause from  $L_i$  with a negative literal (in all possible ways).
- Step 2:** replace clauses in  $L_{i+1}$  by their negative saturations and for each set of mutually isomorphic clauses keep only one of them.
- Step 3:** remove from  $L_{i+1}$  all clauses which cover more than  $t$  examples in the dataset.
- Step 4:** if  $L_{i+1}$  is empty or  $i + 1 > d$  then finish and return  $\bigcup_{j=0}^{i+1} L_j$ . Otherwise set  $i := i + 1$  and go to step 1.

As can be seen from the above pseudocode, we restricted ourselves to mining clauses which contain only negative literals. This essentially corresponds to mining positive conjunctive queries, which is arguably the most typical scenario. Nonetheless, it would be easy to allow the algorithm to search for general clauses, as the  $\theta$ -subsumption solver used in the implementation actually allows efficient handling of negations.

We implemented the level-wise algorithm and the domain theory learner in Java<sup>14</sup>. To check the coverage of examples using  $\theta$ -subsumption, we used an implementation of the  $\theta$ -subsumption algorithm from [8]. For theorem proving, we used an incremental grounding solver which relies on the Sat4j library [1] for solving ground theories and the  $\theta$ -subsumption engine from [8].

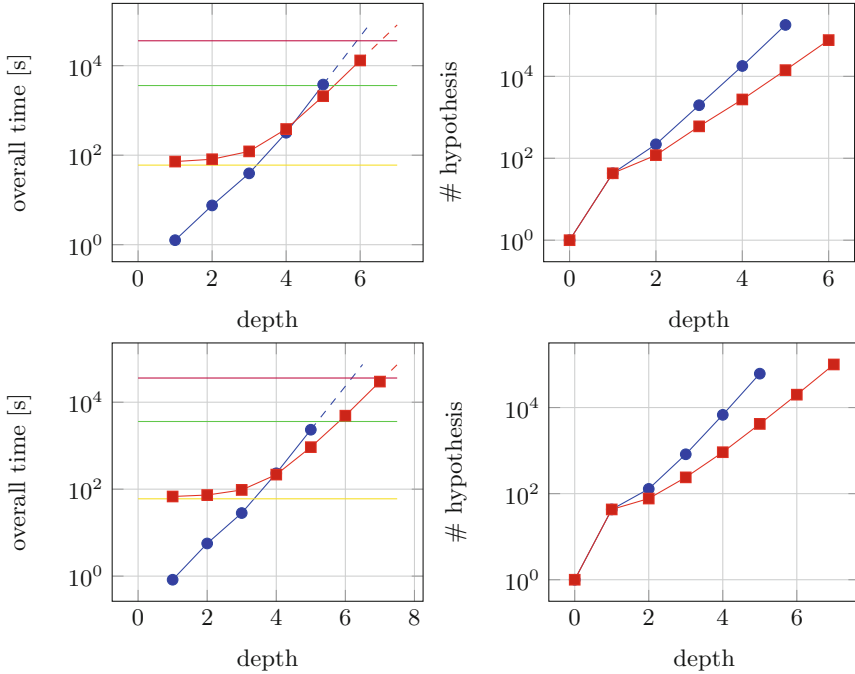
## 4.2 Results

We measured runtime and the total number of clauses returned by the level-wise algorithm without saturations and with saturations. Both algorithms were

<sup>12</sup> A clause is said to be connected if it cannot be written as disjunction of two non-empty clauses. For instance  $\forall X, Y : p_1(X) \vee p_2(Y)$  is not connected because it can be written also as  $(\forall X : p_1(X)) \vee (\forall Y : p_2(Y))$  but  $\forall X, Y : p_1(X) \vee p_2(Y) \vee p_3(X, Y)$  is connected. If a clause is connected then its saturation is also connected.

<sup>13</sup> Frequent conjunctive pattern mining can be emulated in our setting. It is enough to notice that the clauses that we construct are just negations of conjunctive patterns.

<sup>14</sup> Available from <https://github.com/martinsvat>.



**Fig. 1. Left panels:** Runtime of the level-wise algorithm using saturations for pruning (red) and without using saturations (blue). **Right panels:** Number of clauses constructed by the algorithm using saturations (red) and without using saturations (blue). Top panels display results for maximal number of covered examples equal to dataset size minus one and bottom panels for this parameter set to dataset size minus 50, which corresponds to minimum frequency of 50. One minute, one hour, and ten hours are highlighted by yellow, green, and purple horizontal lines. Runtimes are extrapolated by exponential function and shown in dashed lines. (Color figure online)

exactly the same, the only difference being that the second algorithm first learned a domain theory and then used it for computing the saturations. Note in particular that both algorithms used the same isomorphism filtering. Therefore any differences in computation time must be directly due to the use of saturations.

We performed the experiments reported here on the NCI dataset KM20L2. The learned domain theories were restricted to contain only clauses with at most two literals. We set the maximum number of covered examples equal to the number of examples in the dataset minus one (which corresponds to a minimum frequency constraint of 1 when we view the clauses as negated conjunctive patterns). Then we also performed an experiment where we set it equal to the number of examples in the dataset minus 50 (which analogically corresponds to a minimum frequency constraint of 50). We set the maximum time limit to 10 h.



The results of the experiments are shown in Fig. 1. The pruning method based on saturations turns out to pay off when searching for longer clauses where it improves the baseline by approximately an order of magnitude and allows it to search for longer hypotheses within the given time limit. When searching for smaller clauses, the runtime is dominated by the time for learning the domain theory, which is why the baseline algorithm is faster in that case. The number of generated clauses, which is directly proportional to memory consumption, also becomes orders of magnitude smaller when using saturations for longer clauses. Note that for every clause constructed by the baseline algorithm, there is an equivalent clause constructed by the algorithm with the saturation-based pruning. We believe these results clearly suggest the usefulness of the proposed method, which could potentially also be used inside many existing ILP systems.

## 5 Related Work

The works most related to our approach are those relying on a special case of Plotkin’s relative subsumption [15] called generalized subsumption [2]. Generalized subsumption was among others used in [10]. In Sect. 3.5 we discussed the reasons why relative subsumption is not suitable for pruning. Background knowledge was also used to reduce the space of hypotheses in the Progol 4.4 system [12], which uses Plotkin’s relative clause reduction. Note that the latter is a method for *removing* literals from bottom clauses, whereas in contrast our method is based on *adding* literals to hypotheses. Hence, the Progol 4.4 strategy is orthogonal to the methods presented in this paper. Another key difference is that our approach is able to learn the background knowledge from the training data whereas all the other approaches use predefined background knowledge. Finally, our approach is not limited to definite clauses, which is also why we do not use SLD resolution. On the other hand, as our method is rooted in first-order logic (due to the fact that we use the learning from interpretations setting) and not directly in logic programming, it lacks some of the expressive power of logic programming.

## 6 Conclusions

In this paper, we introduced a generally applicable method for pruning hypotheses in ILP, which goes beyond mere isomorphism testing. We showed that the method is able to reduce the size of the hypothesis space by orders of magnitudes, and also leads to a significant runtime reduction. An interesting aspect of the proposed method is that it combines induction (domain theory learning) and deduction (theorem proving) for pruning the search space. In future, it would be interesting to combine these two components of our approach more tightly.

**Acknowledgements.** MS, GŠ and FŽ acknowledge support by project no. 17-26999S granted by the Czech Science Foundation. This work was done while OK was with Cardiff University and supported by a grant from the Leverhulme Trust (RPG-2014-164). SS is supported by ERC Starting Grant 637277. Computational resources were provided by the CESNET LM2015042 and the CERIT Scientific Cloud LM2015085, provided under the programme “Projects of Large Research, Development, and Innovations Infrastructures”.

## References

1. Berre, D.L., Parrain, A.: The SAT4J library, release 2.2. *J. Satisfiability Boolean Model. Comput.* **7**, 50–64 (2010)
2. Buntine, W.L.: Generalized subsumption and its applications to induction and redundancy. *Artif. Intell.* **36**(2), 149–176 (1988)
3. Chekuri, C., Rajaraman, A.: Conjunctive query containment revisited. *Theor. Comput. Sci.* **239**(2), 211–229 (2000)
4. Dechter, R.: *Constraint Processing*. Elsevier Morgan Kaufmann, San Francisco (2003)
5. Dehaspe, L., De Raedt, L.: Mining association rules in multiple relations. In: Lavrač, N., Džeroski, S. (eds.) *ILP 1997*. LNCS, vol. 1297, pp. 125–132. Springer, Heidelberg (1997). [https://doi.org/10.1007/3540635149\\_40](https://doi.org/10.1007/3540635149_40)
6. Ferilli, S., Fanizzi, N., Di Mauro, N., Basile, T.M.: Efficient  $\theta$ -subsumption under object identity. In: 2002 AI\*IA Workshop, pp. 59–68 (2002)
7. van Hoeve, W.J.: The alldifferent constraint: A survey (2001). CoRR cs.PL/0105015. <http://arxiv.org/abs/cs.PL/0105015>
8. Kuželka, O., Železný, F.: A restarted strategy for efficient subsumption testing. *Fundam. Inform.* **89**(1), 95–109 (2008)
9. Kuželka, O., Železný, F.: Block-wise construction of tree-like relational features with monotone reducibility and redundancy. *Mach. Learn.* **83**(2), 163–192 (2011)
10. Malerba, D.: Learning recursive theories in the normal ILP setting. *Fundam. Inform.* **57**(1), 39–77 (2003)
11. Maloberti, J., Sebag, M.: Fast theta-subsumption with constraint satisfaction algorithms. *Mach. Learn.* **55**(2), 137–174 (2004)
12. Muggleton, S.: Inverse entailment and progol. *New Gen. Comput.* **13**(3–4), 245–286 (1995)
13. Newborn, M.: *Automated Theorem Proving - Theory and Practice*. Springer, New York (2001). <https://doi.org/10.1007/978-1-4613-0089-2>
14. Nijssen, S., Kok, J.N.: Efficient frequent query discovery in FARMER. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) *PKDD 2003*. LNCS (LNAI), vol. 2838, pp. 350–362. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-39804-2\\_32](https://doi.org/10.1007/978-3-540-39804-2_32)
15. Plotkin, G.D.: A note on inductive generalization. *Mach. Intell.* **5**(1), 153–163 (1970)
16. Raedt, L.D.: Logical settings for concept-learning. *Artif. Intell.* **95**(1), 187–201 (1997)
17. Ralaivola, L., Swamidass, S.J., Saigo, H., Baldi, P.: Graph kernels for chemical informatics. *Neural Netw.* **18**(8), 1093–1110 (2005)
18. Ramon, J., Roy, S., Jonny, D.: Efficient homomorphism-free enumeration of conjunctive queries. In: *Preliminary Papers ILP 2011*, p. 6 (2011)

19. Riedel, S.: Improving the accuracy and efficiency of MAP inference for markov logic. In: 24th Conference on Uncertainty in Artificial Intelligence, UAI 2008, pp. 468–475 (2008)
20. Stepp, R.E., Michalski, R.S.: Conceptual clustering: inventing goal-oriented classifications of structured objects. In: Machine Learning: An Artificial Intelligence Approach, vol. 2, pp. 471–498 (1986)
21. Tamaddoni-Nezhad, A., Muggleton, S.: The lattice structure and refinement operators for the hypothesis space bounded by a bottom clause. *Mach. Learn.* **76**(1), 37–72 (2009)
22. Weisfeiler, B., Lehman, A.: A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia* **2**(9), 12–16 (1968)



# An Investigation into the Role of Domain-Knowledge on the Use of Embeddings

Lovekesh Vig<sup>1</sup>(✉), Ashwin Srinivasan<sup>2</sup>, Michael Bain<sup>3</sup>, and Ankit Verma<sup>4</sup>

<sup>1</sup> TCS Research, New Delhi, India  
lovekesh.vig@tcs.com

<sup>2</sup> School of Computer Science and Information Systems, BITS Pilani,  
Goa Campus, Sancoale 403726, Goa, India

<sup>3</sup> School of Computer Science and Engineering, UNSW,  
Sydney, NSW 2052, Australia

<sup>4</sup> School of Computational and Integrative Sciences,  
Jawaharlal Nehru University, New Delhi, India

**Abstract.** Computing similarity in high-dimensional vector spaces is a long-standing problem that has recently seen significant progress with the invention of the *word2vec* algorithm. Usually, it has been found that using an embedded representation results in much better performance for the task being addressed. It is not known whether embeddings can similarly improve performance with data of the kind considered by Inductive Logic Programming (ILP), in which data apparently dissimilar on the surface, can be similar to each other given domain (background) knowledge. In this paper, using several ILP classification benchmarks, we investigate if embedded representations are similarly helpful for problems where there is sufficient amounts of background knowledge. We use tasks for which we have domain expertise about the relevance of background knowledge available and consider two subsets of background predicates (“sufficient” and “insufficient”). For each subset, we obtain a baseline representation consisting of Boolean-valued relational features. Next, a vector embedding specifically designed for classification is obtained. Finally, we examine the predictive performance of widely-used classification methods with and without the embedded representation. With sufficient background knowledge we find no statistical evidence for an improved performance with an embedded representation. With insufficient background knowledge, our results provide empirical evidence that for the specific case of using deep networks, an embedded representation could be useful.

## 1 Introduction

The idea of a distance between instances is at the heart of automated methods for classification. This is self-evident for clustering methods that use an explicit distance measure like the Euclidean distance. But even supervised classification methods can often be recast as techniques using more specialised distance

calculation. It is therefore not entirely unexpected that with data requiring a high-dimensional representation, the choice of distance measure becomes increasingly important for identifying groups of similar instances. In [1], the authors show that as the data dimension increases, distances can behave unexpectedly. Crucially, for certain obvious measures of distance (the mean-squared distance, for example; and more generally  $L_k$  norms for  $k > 1$ ), notions of nearest- and furthest-points become meaningless, and all points are approximately at the same distance from each other. The results concerning high-dimensional data are of relevance to one form of relational learning, in which the data are represented by relational (Boolean) features. In Inductive Logic Programming (ILP), there is a long history (starting from [11]) of addressing classification problems by first identifying Boolean functions of data instances, defined in terms of predicates provided as background knowledge. The values of the functions identified represent a relational instance as a Boolean vector (with the  $i$ -th entry for a relational instance  $x$  being 1 if the corresponding function  $F_i(x) = 1$ ), augmented with class value of the instance. A straightforward approach would then attempt to build a classification model using the class-augmented Boolean vectors as input data. Difficulties can arise, since the Boolean vectors may contain 100's or even 1000's of dimensions, thus bringing up the problems associated with high-dimensional data.

Ignoring for the moment the large body of work on feature-subset selection, at least two kinds of distance-relevant remedies have been proposed to address this issue: (1) In [1], “fractional norms” ( $L_k$  norms with  $k < 1$ ) have been found to be more useful as dimensionality increases; and (2) Distance computations use a representation of data instances as vectors of real-numbers with fewer dimensions. This is similar in spirit to what is done in principal components analysis (PCA), but the new dimensions are the result of non-linear transformations. One way of constructing this non-linear transformation, or embedding, of data instances is via the use of neural networks, most prominently embodied by the *word2vec* algorithm.

Developed by Mikolov *et al.* [15], *word2vec* refers to neural network models that are able to embed words into fixed dimensional vector spaces based on their context. The resulting embeddings have been shown to capture some desirable semantic properties with words sharing similar context being mapped to similar vectors. The technique is not restricted to text and embeddings can also be generated in any supervised classification setting in either a class-sensitive or class-agnostic fashion.

Recent methods to model relational data have attempted to map entities and relationships into an embedding space. Methods such as transE [2], assume that the relationship between two entities is a translation in the joint vector space of entities and relations, transH [26] allows for an entity to have multiple representations depending on the relationship, and TransR [12] allows for different embedding spaces for entities and relationships and performs translations in the relation space. However, these methods are directed towards the task of knowledge graph completion.

In this paper, we are interested in whether neural-net embeddings can assist the classification of relational data instances. Specifically, we investigate whether there is evidence for the following hypothesis:

**Embedding Hypothesis ( $H_1$ ).** Even for problems for which a sufficient amount of background knowledge exists, embedded representations are still useful.

Here, “useful” will mean predictive performance improves. The corresponding “null” hypothesis ( $H_0$ ) for experiments is that for problems with sufficient background knowledge, there is no statistical evidence for the utility of embeddings. We will also comment on a hypothesis related to  $H_1$ , and of practical interest, namely: if there isn’t sufficient background knowledge, then embedded representations are useful.

Two immediate difficulties arise when seeking to test the hypothesis  $H_1$ . First, how are we to judge whether background knowledge is sufficient? Secondly, how can we ensure a fair comparison of performance, against reports in the literature? The question of sufficiency of background knowledge is essentially a domain-specific statement, and therefore best provided as domain-knowledge. Comparisons with past results is clearly best done with the same inputs and data-splits used in those reports. For a small number of problems in the ILP literature, we are fortunate both to have domain-knowledge about the relevance of background predicates, and the same input/data-splits used, and we will use these datasets in the paper. Using the relevance-information, we first obtain a baseline representation without the use of embeddings. This consists of relational features defined in terms of the background knowledge.<sup>1</sup> The usual approach to constructing embeddings is class-label agnostic. Instead, we construct an embedding that specifically accounts for class-labels using networks that attempt to maximise the distances between classes (using the baseline representations of training instances in each class). Finally, we use three classification methods that have been widely successful (deep networks, boosted trees and support vector machines). With this choice of problems, methods, and embeddings we compare the predictive performance obtained with the vector-space embeddings against the performance obtained with the baseline representation. If there is evidence of an improvement in performance. For completeness, we also perform the same comparisons using the most widely-used class-agnostic embedding method.

The rest of the paper is organised as follows. In Sect. 2 we briefly describe how we obtain baseline- and vector-representations of relational data. Experimental evaluation of the conjecture is in Sect. 3. Section 4 concludes the paper.

---

<sup>1</sup> ILP practitioners will recognise this as a “propositionalisation” step. In fact, as will become apparent, our approach for obtaining features is simpler than most propositionalisation methods.

## 2 Baseline and Vector-Space Representations

### 2.1 First-Order Feature-Based Representation

We use the approach similar to [19] for obtaining a baseline feature representation for relational data. This does a random selection of features expressible within a mode language and using a weak-constraint of relevancy based on subsuming the most-specific clause of at least one data instance (details of the mode-language and the most-specific clause are as described in [17]). The procedure for random feature selection is in Fig. 1.

The ILP practitioner will recognise this as a simple randomised proposition-alisation method. The idea of obtaining a Boolean-feature representation from relational data has a long and useful history inspired by the initial work in LINUS [11] (see, for example, [4, 7, 18–20, 23] and of more immediate relevance here, [5, 13] for use of such features as inputs to neural networks).

### 2.2 Vector-Space Representation

The objective with obtaining an embedded representation is to generate representations of the data as fixed-sized real-valued vectors, that are usually of a lower dimension than the input representation. We start with the simplest way of

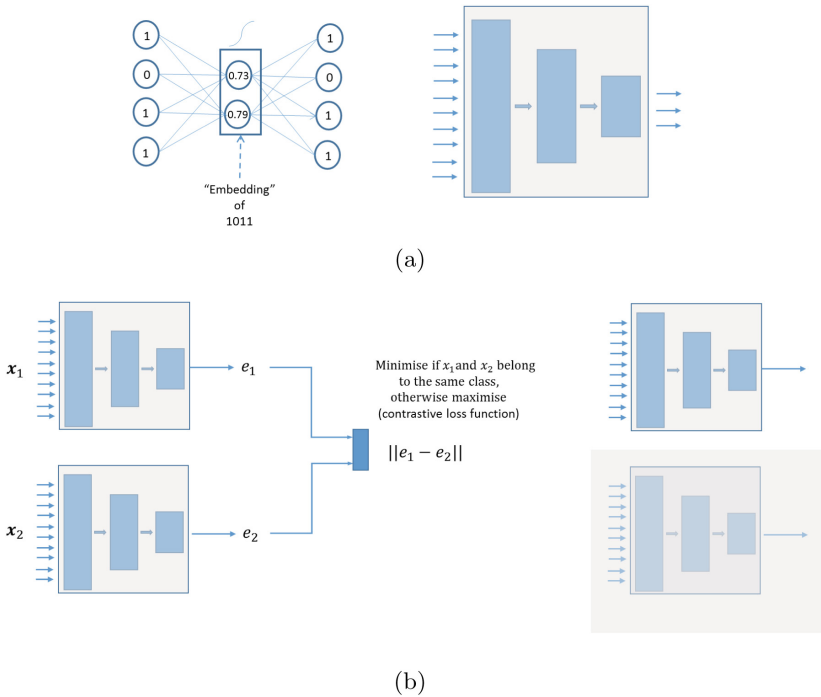
*DrawFeatures*( $B, M, E, \mathcal{L}, d, MaxDraws$ ) :

1. Let  $F$  be  $\langle \rangle$
2. Let  $draws = 0$
3. Let  $i = 1$
4. Let  $Drawn$  be  $\emptyset$
5. **while**  $draws \leq MaxDraws$  **do**
  - (a) Randomly draw with replacement an example  $e_i \in E$
  - (b) Let  $\perp_d(B, e_i)$  be the most specific rule in the depth-limited mode language  $\mathcal{L}_d(M)$  that subsumes  $\perp(B, e_i)$  (the most-specific clause that entails  $e$ , given  $B$ ).
  - (c) Randomly draw an clause  $C_i$  s.t.  $C_i \subseteq \perp_d(B, e_i)$
  - (d) **if** ( $C_i$  is not redundant given  $Drawn$ ) **then**
    - i. Let  $C_i = (Class(x, c) \leftarrow Cp_i(x))$
    - ii. Let  $f_i = (F_i(x) \leftarrow Cp_i(x))$
    - iii. Update sequence  $F$  with  $f_i$
    - iv.  $Drawn := Drawn \cup \{C_i\}$
    - v. increment  $i$
  - (e) increment  $draws$
6. **done**
7. return  $F$

**Fig. 1.** A procedure for obtaining features within a mode language, given background knowledge and data.

generating such a vector that is class-agnostic (that is, the representation is not specifically designed for classification tasks). We then describe a class-sensitive approach, which attempts to maximise the distance between vectors of different classes.

The class-agnostic representation is generated using an autoencoder. Here a low-dimensional real-vector representation of the data is obtained by using a feedforward neural network that is trained to reconstruct the input data, using one or more intermediate layers with significantly fewer nodes than the input. The class-sensitive embeddings are obtained by learning similarity across samples of the same class using a Siamese network [10]. A Siamese network takes pairs of samples as input and maps each sample into an embedding space via identical base networks. During training the euclidean distance in the embedding space between samples of the same class is minimized and that between samples belonging to different classes is maximized (see Fig. 2(a) and (b)). Networks for the autoencoder and Siamese embeddings need a feature-based representation



**Fig. 2.** (a) Class-agnostic; and (b) Class-sensitive vector representations. (a) is obtained from the hidden layers of an auto-encoder (that is, a network that simply reconstructs the input data: usually the auto-encoder will have more than one hidden layer, as shown in the shaded box, top-right); and (b) is obtained from the hidden layers of a Siamese network (that is, one of a matched pair of networks that are trained to separate instances  $x_{i,j}$  belonging to different classes).



to start with: we use the Boolean feature-vector representation of the relational instances as the input for the autoencoder or the Siamese network.

### 3 Empirical Evaluation

#### 3.1 Aims

We investigate the Embedding Hypothesis introduced in Sect. 1. Recast in the terminology of statistical hypothesis testing, the Embedding Hypothesis  $H_1$  and the corresponding null hypothesis are these:<sup>2</sup>

$H_1$ : The (median) predictive performance using an embedded representation is higher than that obtained using the baseline representation.

$H_0$ : There is no difference in the (median) predictive performances of the embedded and baseline representations.

#### 3.2 Materials

**Data and Background Knowledge.** We report results from experiments conducted using 7 well-studied real world problems from the ILP literature. These are: Mutagenesis [8]; Carcinogenesis [9]; DssTox [16]; and 4 datasets arising from the comparison of Alzheimer’s drugs denoted here as *Amine*, *Choline*, *Scop* and *Toxic* [22]. We have chosen these datasets for the following reasons: (1) Results are available in the ILP literature using standardised testing protocols; (2) We have access to orderings over sets of background knowledge (from less to more relevant).<sup>3</sup>

We refer the reader to the relevant ILP literature with details of the background knowledge available for each problem. Here we restrict ourselves to describing the two end-points of the relevancy ordering available for background knowledge:

**Least relevant.** For Mutagenesis, Carcinogenesis and DssTox, these are the predicates describing just the molecular (atom-bond) structure. For the Alzheimers datasets, these are predicates that simply include counts of the numbers and kinds of substitutions into template positions of the molecules.

<sup>2</sup> It is useful to clarify the choice of hypotheses. The null hypothesis  $H_0$  holds if there is no evidence for the utility of an embedded representation. This choice is motivated by the fact that obtaining an class-sensitive embedded representation requires substantial computational effort. Thus, the experiment is set up to be conservative: an embedded representation will only be obtained if there is statistical evidence to support it.

<sup>3</sup> Information of this nature is already available in the ILP literature for Mutagenesis and Carcinogenesis from [24]. The information on relevance in that paper was provided by Professor R.D. King. We thank him extending this to the other problems here.

**All.** For Mutagenesis and Carcinogenesis these include predicate definitions for molecular structure; higher-order functional groups; ring structures; bulk properties along with arithmetic predicates to deal with numbers. For the Alzheimer’s data, this includes additional predicates describing the actual substituents at template positions; and physico-chemical properties of the substituents (size, polarity, hydrophobicity *etc.*: the so-called “Hansch” predicates).

In this paper, we will take the least relevant subset of background as constituting insufficient domain-knowledge for the problem; and the “All” subset as constituting (a superset of) sufficient background knowledge. We note that of the 7 datasets, 6 have both sufficient and insufficient definitions for background knowledge. One dataset (DssTox) only has insufficient background knowledge (that is, “All” is the same as “Least Relevant”).

**Algorithms and Machines.** Random features were constructed on an Intel Core i7 laptop computer, using VMware virtual machine running Fedora 13, with an allocation of 2 GB for the virtual machine. The Prolog compiler used was Yap. Feature-construction uses the utilities provided by the Aleph ILP system [21] for constructing most-specific clauses in a depth-bounded mode language, and for drawing clauses subsuming such most-specific clauses. No use is made of any of the search procedures within Aleph. The deep networks were constructed using the Keras library with Theano as the backend, and were trained using an NVIDIA K-40 GPU card.

### 3.3 Method

The main steps of our experimental method are these:

For each problem  $p$ :

1. Let  $Tr$  denote the training set and  $Te$  denote the test set
2. Obtain a sequence of features  $F$  using  $Tr$
3. Let  $Tr_F$  denote the baseline feature-based representation of  $Tr$  using values for the  $F$  and  $Te_F$  be the corresponding feature-based representation of  $Te$  (we assume for simplicity that  $Tr_F$  and  $Te_F$  include the class-values of instances)
4. Let  $M_F$  be the classification model constructed with  $Tr_F$  and  $A_F$  be its performance on  $Te_F$
5. Let  $Tr_V$  be a vector-space representation of  $Tr$  obtained using  $Tr_F$  and a class-sensitive embedding  $e$ . Let  $Te_V$  be the vector-space representation of  $Te$
6. Let  $M_V$  be the classification model by a method  $m$  constructed with  $Tr_V$  and  $A_V$  be its performance on  $Te_V$
7. Compare  $A_F$  and  $A_V$

A number of points require further clarification:

- For obtaining features, we allow a maximum number of samples of 10,000. Thus, the maximum dimensionality of the baseline representation is bounded by this number.
- By a class-agnostic embedding, we mean an embedding obtained from using an auto-encoder of the Boolean feature-vectors obtained using the baseline representation of the data. By a class-sensitive embedding, we mean an embedding obtained by using a Siamese network trained to separate feature-vectors of positive and negative instances.
- For the Siamese networks used for embeddings we chose a deep feedforward network with two hidden layers, and an embedding dimension of 100. The training procedure involved creating 20 training batches comprising of 1000 positive and 1000 negative sample pairs (obtained by sampling with repetition) in addition to 500 positive and 500 negative validation pairs. After every 20 epochs of training, the worst performing class pairs on the validation set were identified and additional pairs from these classes were added to the training set. This was continued until performance converged on the validation set.
- We consider three kinds of classification methods. First, we use a straightforward deep neural network (DNN). This is a model with multiple, fully connected feedforward layers of rectified linear (ReLU) units followed by Dropout for regularization (see [6] for a description of these ideas). The model weights were initialized with a Gaussian distribution. The number of layers, number of units for each layer, the optimizers, and other training hyperparameters such as learning rate, were determined automatically using a validation set, which is part of the training data. Since the data is limited for the datasets under consideration, after obtaining the model which yields the best validation score, the chosen model is then retrained on the complete training set (this includes the validation set) until the training loss exceeds the training loss obtained for the chosen model during validation. The second method considered are gradient boosted trees as implemented by the XGBoost procedure [3], wherein fixed sized trees are usually the base models, and new tree models are fit on the error residuals at each step, and successively merged with the base model. We used a maximum tree depth of 6, with an L2 regularization parameter of 1, and a step size of 0.3 for our XGBoost model. Finally, we also consider support vector machines.
- For all problems, performance is assessed using 10-fold cross-validation. So the steps above are repeated 10 times for each train-test split. A standardised cross-validation split for these specific datasets is available from previous reports in the literature, which allows a direct comparison against those results. For the comparisons here, we will take an embedding-based representation to be useful (a “win” for the embedded representation) if the cross-validated accuracy with embeddings is greater than the corresponding accuracy with the baseline representation.
- The appropriate statistical test for testing the hypothesis in Sect. 3.1 is the one-sided sign test. The computation is simple: suppose the number of trials

on which an embedded representation does better is  $s$  out of  $n$ . Let  $h$  be  $s$  or  $n - s$ , whichever is smaller. The  $p$ -value is the binomial probability of observing at most  $h$  heads in  $n$  tosses of a fair coin. If this value exceeds some significance level  $\alpha$ , we cannot reject the null hypothesis. We will take  $\alpha = 0.05$ .

### 3.4 Results

The principal results of the empirical evaluation are tabulated in Figs. 3 and 4, and summarised in Fig. 5.

Problem	Accuracy	
	<i>Base</i>	<i>Embed</i>
<i>Mut188</i>	0.91(0.06)	0.91 (0.04)
<i>Canc330</i>	0.68(0.03)	0.60 (0.07)
<i>Amine</i>	0.89(0.04)	0.93 (0.03)
<i>Choline</i>	0.81(0.03)	0.81 (0.05)
<i>Scop</i>	0.80(0.05)	0.83 (0.08)
<i>Toxic</i>	0.93(0.03)	0.93 (0.03)

(a) Deep Network

Problem	Accuracy	
	<i>Base</i>	<i>Embed</i>
<i>Mut188</i>	0.91 (0.06)	0.89 (0.05)
<i>Canc330</i>	0.64 (0.10)	0.61 (0.08)
<i>Amine</i>	0.91 (0.02)	0.92 (0.02)
<i>Choline</i>	0.83 (0.03)	0.81 (0.04)
<i>Scop</i>	0.74 (0.02)	0.80 (0.09)
<i>Toxic</i>	0.93 (0.03)	0.93 (0.03)

(b) XGBoost

Problem	Accuracy	
	<i>Base</i>	<i>Embed</i>
<i>Mut188</i>	0.88 (0.05)	0.90 (0.05)
<i>Canc330</i>	0.60 (0.08)	0.55 (0.03)
<i>Amine</i>	0.93 (0.03)	0.93 (0.03)
<i>Choline</i>	0.83 (0.04)	0.81 (0.04)
<i>Scop</i>	0.76 (0.09)	0.81 (0.08)
<i>Toxic</i>	0.93 (0.03)	0.93 (0.03)

(c) SVM

**Fig. 3.** Results with sufficient background knowledge. Estimated predictive accuracies using the baseline (*Base*) and embedded (*Embed*) representations, obtained using a Siamese network. Note: *DssTox* does not appear in the list of problems here, since it is characterised as having insufficient background knowledge.

The  $p$ -value column in Fig. 5 is the probability of observing the result if the null hypothesis is true (that is, there is no difference in median performance when using an embedded representation). A statistical case can be made when using the one-sided sign test, or when numbers are small, of not ignoring ties that support the null hypothesis (see for example, [14]). The tabulations in Fig. 5 do this, although the conclusions we draw below do not change even if ties are discarded. The main observations that can be made from these tabulations are these:

- With sufficient background knowledge ( $B$ ), we do not have statistically significant evidence that an embedded representation ( $E$ ) is useful ( $p$ -values are all over  $\alpha = 0.05$ ).

Problem	Accuracy	
	<i>Base</i>	<i>Embed</i>
<i>Mut188</i>	0.82 (0.06)	0.84 (0.05)
<i>Canc330</i>	0.56 (0.08)	0.57 (0.09)
<i>DssTox</i>	0.70 (0.06)	0.74 (0.05)
<i>Amine</i>	0.73 (0.03)	0.75 (0.03)
<i>Choline</i>	0.70 (0.02)	0.72 (0.02)
<i>Scop</i>	0.59 (0.07)	0.60 (0.06)
<i>Toxic</i>	0.78 (0.03)	0.80 (0.03)

(a) Deep Network

Problem	Accuracy	
	<i>Base</i>	<i>Embed</i>
<i>Mut188</i>	0.82 (0.04)	0.83 (0.04)
<i>Canc330</i>	0.56 (0.07)	0.56 (0.06)
<i>DssTox</i>	0.75 (0.05)	0.73 (0.05)
<i>Amine</i>	0.74 (0.03)	0.75 (0.04)
<i>Choline</i>	0.71 (0.02)	0.70 (0.03)
<i>Scop</i>	0.62 (0.06)	0.59 (0.07)
<i>Toxic</i>	0.78 (0.02)	0.78 (0.03)

(b) XGBoost

Problem	Accuracy	
	<i>Base</i>	<i>Embed</i>
<i>Mut188</i>	0.67 (0.10)	0.67 (0.09)
<i>Canc330</i>	0.54 (0.02)	0.54 (0.02)
<i>DssTox</i>	0.75 (0.06)	0.69 (0.05)
<i>Amine</i>	0.72 (0.04)	0.72 (0.04)
<i>Choline</i>	0.70 (0.03)	0.70 (0.03)
<i>Scop</i>	0.60 (0.05)	0.59 (0.06)
<i>Toxic</i>	0.77 (0.04)	0.72 (0.05)

(c) SVM

**Fig. 4.** Results with insufficient background knowledge. Estimated predictive accuracies using the baseline (*Base*) and embedded (*Embed*) representations.

Back.	Successes		<i>p</i> value
	$\neg E$	<i>E</i>	
<i>B</i>	4	2	0.34
$\neg B$	0	7	0.01

(a) Deep Network

Back.	Embed. Wins		<i>p</i> value
	$\neg E$	<i>E</i>	
<i>B</i>	4	2	0.34
$\neg B$	5	2	0.22

(b) XGBoost

Back.	Embed. Wins		<i>p</i> value
	$\neg E$	<i>E</i>	
<i>B</i>	4	2	0.34
$\neg B$	6	1	0.062

(c) SVM

**Fig. 5.** Summary of results. Here *B* denotes sufficient background knowledge and *E* denotes the number of times the embedded representation performed better.  $\neg E$  denotes the number of times the embedded representation did not perform better.

- The *p*-values when background knowledge is insufficient ( $\neg B$ ) is lower than those obtained in the case above. However, only in the case of deep networks is there appear to be statistically significant evidence that an embedded representation improves performance.

We are thus able to make the following practical suggestions:

1. If background knowledge is deficient, then consider using a deep network with an embedded representation from a Siamese network for the relational data.

2. If background knowledge is sufficient, then an embedded representation will not be needed. Consider using either a deep networks or gradient boosted trees with the baseline representation for the relational data.<sup>4</sup>

There are some additional observations we provide on the experiments:

**General.** The following general comments concern the methods and problems:

- In common to all propositionalisation-based methods, there is a limitation to the kinds of problems that can be tackled. Suppose we consider the following categories of problems: (a) those for which relational features can be constructed, and are sufficient; (b) those for which relational features can be constructed, but are not sufficient; and (c) those for which relational features cannot be constructed. The results in the paper refer to categories (a) and (b); the latter being the case where the background knowledge is deliberately insufficient. Category (c) is not considered in this paper, and it is indeed instructive to note problems that come under this category. These include the following: those which require the use of recursion; and those that require higher-order (second-order or higher) constructs. The approach we have used cannot handle problems of this kind.
- A common feature of the data is that they are all drawn from the broad area of biochemistry. This raises a question of whether the results are only applicable to data from that domain. The data and problems are extremely varied: in fact, even apparently related problems of mutagenesis and carcinogenesis deal with very different sets of chemicals, and ultimately, have very different target concepts. It can therefore be somewhat misleading to think of the problems as being of the same kind, simply because they deal with different tasks in the broad area of chemical toxicology. The problems were chosen for this study for the following reasons. First, we have very well-established benchmarks for these in the ILP literature. This includes the results from the use of relational features with widely-used statistical learners and from parameter-optimised ILP learners. We also have the same data-splits on which these results were obtained, making a controlled comparison possible. Secondly, the background knowledge ranges from largely propositional (the Alzheimer’s data), to a mix of propositional and first-order relations (mutagenesis and carcinogenesis), to only first-order relations (DssTox). Thirdly, and most important, we have relevance-information about background predicates that we have been able to use to give meaning to notions of sufficient and insufficient background knowledge.

**Specific.** The following comments refer specifically to the results:

- A legitimate concern is whether the results are artifacts of the choice of embedding method, and a different embedding method would in fact result in improving performance. The tabulation in Fig. 6 using embeddings from

---

<sup>4</sup> Comparing the results in Figs. 3 and 7 suggests that this will perform better than using an ILP approach, even with optimised parameters.

autoencoders suggests that this may not be the case (the results are for a deep network: the other classifiers show the same trend).

- How do the results compare with ILP-based methods that use the relational representation directly, or statistical methods with a propositionalised representation? In Fig. 7 we tabulate some of the best predictive performances reported in the ILP literature on the same datasets (with the same cross-validation splits). It is evident that the predictive performances reported in Fig. 3 compare favourably in all cases.

Problem	Accuracy	
	<i>Auto</i>	<i>Siam</i>
<i>Mut188</i>	0.67 (0.07)	0.91 (0.04)
<i>Canc330</i>	0.54 (0.02)	0.60 (0.07)
<i>Amine</i>	0.63 (0.07)	0.93 (0.03)
<i>Choline</i>	0.50 (0.04)	0.81 (0.05)
<i>Scop</i>	0.51 (0.07)	0.83 (0.08)
<i>Toxic</i>	0.53 (0.06)	0.93 (0.03)

**Fig. 6.** Results with sufficient background knowledge using a deep network. *Auto* denotes embeddings obtained using an autoencoding, and *Siam* denotes embeddings obtained using a Siamese network.

Problem	Accuracy	
	<i>OptILP</i> [24]	<i>Stat</i> [18]
<i>Mut188</i>	0.88(0.02)	0.85(0.05)
<i>Canc330</i>	0.58(0.03)	0.60(0.02)
<i>DssTox</i>	0.73(0.02)	0.72(0.01)
<i>Amine</i>	0.80(0.02)	0.81(0.00)
<i>Choline</i>	0.77(0.01)	0.74(0.00)
<i>Scop</i>	0.67(0.02)	0.72(0.02)
<i>Toxic</i>	0.87(0.01)	0.84(0.01)

**Fig. 7.** Predictive accuracies of some of the best reported performances using ILP (*OptILP*) and statistical learners (*Stat*) in the ILP literature. All estimates are 10-fold cross-validation estimates that use the same splits as those used here, and with the background predicates used to obtain the results in Fig. 3.

## 4 Concluding Remarks

This paper has been concerned with the effect of background knowledge on the use of embedded vector-space representations of relational data. Our principal contribution has been to present evidence that for problems rich in domain knowledge, embedded representations of the data may be limited value. During the course of

our experiments, we also find evidence that for problems poor in domain knowledge, embedded representations may prove useful for deep networks.

Turning to the results, there are some cautionary notes. Even fixing the background knowledge and the baseline features, the results are still dependent on the choice of problems, classification techniques, and embedding methods used. These are the perils of an empirical study of course. We have tried to control for this by using three of the most powerful classification methods available at present. The use of Siamese networks is also considered one of the best ways of obtaining embeddings for classification (results, for example, with the standard class-agnostic embeddings obtained from an auto-encoder are substantially worse than with the class-sensitive method). To ensure we do not see a spurious effect, we also need to ensure our baseline is good. The results in Fig. 7 suggest that our baseline is at least as good or better, than the best reported in the literature. It is of course possible that with more experimentation, results may change. If so—as Keynes is reputed to have said—we would have to change our mind. In the meantime, the experiments here mark out a territory within which the conclusions drawn are plausible and the practical suggestions, probably useful.

**Acknowledgements.** A.S. is a Visiting Professor in the Department of Computer Science, University of Oxford; and Visiting Professorial Fellow, School of CSE, UNSW Sydney. A.S. is supported by the SERB grant EMR/2016/002766.

## References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 420–434. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44503-X\\_27](https://doi.org/10.1007/3-540-44503-X_27)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26, pp. 2787–2795. Curran Associates Inc, Red Hook (2013)
3. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. ACM (2016)
4. Faruque, T.A., Srinivasan, A., King, R.D.: Topic models with relational features for drug design. In: Riguzzi, F., Železný, F. (eds.) ILP 2012. LNCS (LNAI), vol. 7842, pp. 45–57. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38812-5\\_4](https://doi.org/10.1007/978-3-642-38812-5_4)
5. França, M.V.M., Zaverucha, G., d’Avila Garcez, A.S.: Fast relational learning using bottom clause propositionalization with artificial neural networks. *Mach. Learn.* **94**(1), 81–104 (2014)
6. Goodfellow, I.J., Bengio, Y., Courville, A.C.: Deep Learning. Adaptive Computation and Machine Learning. MIT Press, Cambridge (2016)



7. Joshi, S., Ramakrishnan, G., Srinivasan, A.: Feature construction using theory-guided sampling and randomised search. In: Železný, F., Lavrač, N. (eds.) *ILP 2008*. LNCS (LNAI), vol. 5194, pp. 140–157. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-85928-4\\_14](https://doi.org/10.1007/978-3-540-85928-4_14)
8. King, R.D., Muggleton, S.H., Srinivasan, A., Sternberg, M.J.: Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proc. Natl. Acad. Sci. U.S.A.* **93**(1), 438–442 (1996)
9. King, R.D., Srinivasan, A.: Prediction of rodent carcinogenicity bioassays from molecular structure using inductive logic programming. *Environ. Health Perspect.* **104**, 1031–1040 (1996)
10. Koch, G.: Siamese neural networks for one-shot image recognition (2015)
11. Lavrač, N., Džeroski, S., Grobelnik, M.: Learning nonrecursive definitions of relations with linus. In: Kodratoff, Y. (ed.) *EWISL 1991*. LNCS, vol. 482, pp. 265–281. Springer, Heidelberg (1991). <https://doi.org/10.1007/BFb0017020>
12. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: *AAAI* (2015)
13. Lodhi, H.: Deep relational machines. In: Lee, M., Hirose, A., Hou, Z.-G., Kil, R.M. (eds.) *ICONIP 2013*. LNCS, vol. 8227, pp. 212–219. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-42042-9\\_27](https://doi.org/10.1007/978-3-642-42042-9_27)
14. Marshall, J.B.: The sign test with ties included. *Appl. Math.* **5**, 1594–1597 (2014)
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*. Proceedings of a Meeting Held 5–8 December 2013, Lake Tahoe, Nevada, United States, pp. 3111–3119 (2013)
16. Muggleton, S.H., Santos, J.C.A., Tamaddoni-Nezhad, A.: TopLog: ILP using a logic program declarative bias. In: de la Garcia, M., Pontelli, E. (eds.) *ICLP 2008*. LNCS, vol. 5366, pp. 687–692. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-89982-2\\_58](https://doi.org/10.1007/978-3-540-89982-2_58)
17. Muggleton, S.: Inverse entailment and progol. *New Gener. Comput.* **13**(3&4), 245–286 (1995)
18. Ramakrishnan, G., Joshi, S., Balakrishnan, S., Srinivasan, A.: Using ILP to construct features for information extraction from semi-structured text. In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) *ILP 2007*. LNCS (LNAI), vol. 4894, pp. 211–224. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-78469-2\\_22](https://doi.org/10.1007/978-3-540-78469-2_22)
19. Saha, A., Srinivasan, A., Ramakrishnan, G.: What kinds of relational features are useful for statistical learning? In: Riguzzi, F., Železný, F. (eds.) *ILP 2012*. LNCS (LNAI), vol. 7842, pp. 209–224. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38812-5\\_15](https://doi.org/10.1007/978-3-642-38812-5_15)
20. Specia, L., Srinivasan, A., Joshi, S., Ramakrishnan, G., das Graças Volpe Nunes, M.: An investigation into feature construction to assist word sense disambiguation. *Mach. Learn.* **76**(1), 109–136 (2009)
21. Srinivasan, A.: *The Aleph Manual* (1999). <http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/>
22. Srinivasan, A., Muggleton, S.H., Sternberg, M.J.E., King, R.D.: Theories for mutagenicity: a study in first-order and feature-based induction. *Artif. Intell.* **85**(1–2), 277–299 (1996)

23. Srinivasan, A., King, R.D.: Feature construction with inductive logic programming: a study of quantitative predictions of biological activity by structural attributes. In: Muggleton, S. (ed.) *ILP 1996*. LNCS, vol. 1314, pp. 89–104. Springer, Heidelberg (1997). [https://doi.org/10.1007/3-540-63494-0\\_50](https://doi.org/10.1007/3-540-63494-0_50)
24. Srinivasan, A., King, R.D., Bain, M.: An empirical study of the use of relevance information in inductive logic programming. *J. Mach. Learn. Res.* **4**, 369–383 (2003)
25. Srinivasan, A., Ramakrishnan, G.: Parameter screening and optimisation for ILP using designed experiments. *J. Mach. Learn. Res.* **12**, 627–662 (2011)
26. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: *AAAI* (2014)

## Author Index

- Antanas, Laura 1  
Artikis, Alexander 78
- Bain, Michael 169  
Bekker, Jessa 16  
Blockeel, Hendrik 63  
Borzacchiello, Domenico 124
- Chinesta, Francisco 124  
Cohen, William 94  
Côte-Real, Joana 31
- Dai, Wang-Zhou 46  
Davis, Jesse 16  
De Raedt, Luc 1  
Dries, Anton 1  
Dumančić, Sebastijan 63  
Dutra, Inês 31
- Folschette, Maxime 124
- Inoue, Katsumi 124
- Katzouris, Nikos 78  
Kaur, Navdeep 94  
Kersting, Kristian 94  
Khot, Tushar 94  
Kunapuli, Gautam 94  
Kuželka, Ondřej 140, 152
- Magnin, Morgan 124  
Moreno, Plinio 1  
Muggleton, Stephen 46
- Natarajan, Sriraam 94  
Nishiyama, Hiroyuki 112
- Ohwada, Hayato 112
- Paliouras, Georgios 78
- Ribeiro, Tony 124  
Rocha, Ricardo 31  
Roux, Olivier 124
- Schockaert, Steven 140, 152  
Šourek, Gustav 140, 152  
Srinivasan, Ashwin 169  
Svatoš, Martin 140, 152
- Tamaddoni-Nezhad, Alireza 46  
Tourret, Sophie 124
- Verma, Ankit 169  
Vig, Lovekesh 169
- Wen, Jing 46
- Železný, Filip 140, 152  
Zhou, Zhi-Hua 46