Working Papers and Documents
of the

IJCAI-ECAI-2018 Workshop on

Learning and Reasoning:

Principles & Applications to
Everyday Spatial and Temporal Knowledge

L & **R** - 2018

July 13 - 14, 2018
Stockholm (Sweden)

Preface

Knowledge representation and reasoning (KRR), on the one hand, and machine learning (ML), on the other hand, have largely been developed as independent research trends in artificial intelligence (AI). Human reasoning, however, is often based on an intricate combination of processes that are related to learning (e.g. induction or extrapolation) and processes that are closer to deductive reasoning. Similarly, we can expect that progress in AI will increasingly need to rely on hybrid approaches that combine the explainability and teachability of KRR methods with the robustness and data-driven nature of ML methods. The ambitious aim of truly integrating reasoning and learning beyond one-way linkage raises many new questions, which this workshop hopes to explore.

Beyond a study of the underlying principles, this workshop also focuses on applications, with a particular emphasis on the use of spatial and temporal knowledge in everyday tasks.

The workshop serves as a forum for researchers from different fields (including Automated Theorem Proving, Cognitive Computing, Cognitive Robotics, Commonsense Reasoning, Constraint Solving, Logic, Mathematics, Machine Learning, Natural Language Processing, Theoretical Computer Science, Qualitative Reasoning) to discuss open problems, methodology, and recent advancements in the field. It also provides a forum for early career researchers to present their current work and to build up networks.

This workshop is the first under this name and with this scope. However it is to a limited extent a follow-up of previous ECAI and IJCAI workshops (already co-organised by three of the co-organisers of the present workshop):

- the successful series of WL4AI workshops (Weighted Logics for Artificial Intelligence: ECAI-2012, IJCAI-2013, IJCAI-2015)
- the IJCAI-2017 workshop on Logical Foundations for Uncertainty and Learning (LFU)

L & R - 2018 looks broadly at the intersection of logical formalisms and learning, by unifying the themes of WL4AI and LFU, and additionally encouraged submissions touching on defeasible reasoning and nonmonotonic frameworks among other issues.

Finally, we would like to express our gratitude to:

- Jesse Davis, Kristian Kersting, Angelika Kimmig, Stephen Muggleton and Marco Schorlemmer for having accepted to give invited talks at this workshop,
- the program committee members for their commitment to the success of this event and for their work,
- the authors of LR-2018 for the quality of their contributions.

Vaishak Belle, Lluis Godo, Henri Prade, Jochen Renz, Steven Schockaert, Ute Schmid, and Diedrich Wolter

Workshop Chairs[1]

Vaishak Belle	University of Edinburgh, UK
Lluis Godo	IIIA - CSIC, Spain
Henri Prade	IRIT - CNRS & Univ. of Toulouse, France & QCIS, UTS, Sydney, Australia
Jochen Renz	The Australian National University, Australia
Steven Schockaert	Cardiff University, UK
Ute Schmid	University of Bamberg, Germany
Diedrich Wolter	University of Bamberg, Germany

[1]This workshop results from the fusion of two workshop proposals that were encouraged to merge by the organizers of the joint workshop program of the Federated AI Meeting of AAMAS, ICML, and IJCAI (FAIM), leading to an unusually large set of Workshop Chairs.

Program Committee

Franz Baader	Technical University of Dresden, Germany
Stefano Borgo	Laboratory for Applied Ontology, Trento, Italy
Gerhard Brewka	University of Leipzig, Germany
Eliseo Clementini	University of L'Aquila, Italy
William Cohen	Carnegie Mellon University, USA
Antoine Cornuejols	AgroParisTech, France
Fabio Cozman	University of São Paulo, Brazil
Jesse Davies	KU Leuven, Belgium
Didier Dubois	IRIT, France
Esra Erdem	Sabanci University, Turkey
Marcelo Finger	University of São Paulo, Brazil
Uli Furbach	Koblenz University, Germany
Manfred Jaeger	Aalborg University, Denmark
Souhila Kaci	University Montpellier, France
Gabriele Kern-Isberner	Dortmund University, Germany
Roni Khardon	Tufts University, USA
Alex Kirsch	Tübingen University/Intuity, Germany
Ondrej Kuželka	KU Leuven, Belgium
Jrme Lang	Paris-Dauphine University, France
Jae Hee Lee	Cardiff University, UK
Sanjiang Li	University of Technology Sydney, Australia
Churn-Jung Liau	Academia Sinica,Taiwan
Emiliano Lorini	IRIT, France
Radu Marinescu	IBM Research, Ireland
Denis Maua	University of São Paulo, Brazil
Wannes Meert	KU Leuven, The Netherlands
Ron Petrick	University of Edinburgh, UK
Quoc-Sang Phan	Carnegie Mellon University, USA
Guilin Qi	Southeast University, China
Gilles Richard	IRIT, France
Thomas Schneider	University of Bremen, Germany
Guillermo Simari	Universidad Nacional del Sur, Argentina
Michael Sioutis	Örebro University, Sweden
Frieder Stolzenburg	Harz University of Applied Sciences, Germany
Umberto Straccia	ISTI-CNR, Pisa, Italy
Guy Van den Broeck	UCLA, USA
Christel Vrain	University of Orleans, France
Mary-Anne Williams	Univ. Tech. Sydney, Australia

LR-2018 Workshop Programme[2]

July 13 and 14, 2018, Stockholm, Sweden

Friday 13

14.30 - 14.10	**Welcome**
14.10 - 15.00	**Invited talk:** *Challenges in Real-World (Spatio-)Temporal Data: Case Studies in Anomaly Detection and Sports* Jesse Davies
15.00 - 15.10	Discussion
15.10 - 15.30	**Session 1:** *Inductive Logic Programming, Ontology Reasoning, and Spatial Knowledge: A Short Survey of 15-Years Research* Francesca Alessandra Lisi
	Coffee Break 15.30 – 16.00
16.00 - 16.50	**Invited talk:** *Meta-Interpretive Learning: Achievements and Challenges* Stephen Muggleton
16.50 - 17.00	Discussion
17.00 - 17.30	**Organiser talk:** *Towards a reconciliation between reasoning and learning: A position paper* Didier Dubois, Henri Prade
17.30 - 17.40	Discussion
17.40 - 18.30	**Invited talk:** *Reasoning at a Distance by Way of Conceptual Metaphors and Blends* Marco Schorlemmer
18.30 - 18.40	Discussion
	End of the afternoon session

[2]The contents of this programme includes a series of invited talks and (shorter) organiser talks together with regular contributions in order to provide multiple views of the ways learning and reasoning may interact. Each session is followed by substantial time for discussion among participants.

Saturday 14

9.00 - 9.50	**Invited talk:** *Probabilistic logic programming and learning* Angelika Kimmig
9.50 - 10.00	Discussion
	Coffee Break 10.00 – 10.30
10.30 - 11.00	**Organiser talk:** *Effective probabilistic logical reasoning in continuous domains* Vaishak Belle
11.00 - 11.10	Discussion
11.10 - 11.50	**Session 2:** *A Symbolic Approach for Explaining Errors in Image Classification Tasks* Marjan Alirezaie, Martin Längkvist, Michael Sioutis and Amy Loutfi *Sugeno Integral for Rule-Based Monotone Classification* Quentin Brabant, Miguel Couceiro, Didier Dubois, Henri Prade and Agnes Rico
11.50 - 12.20	**Organiser talk:** *Geometric representations of logical theories* Steven Schockaert
12.20 - 12.30	Discussion
	Lunch 12.30 – 14.00
14.00 - 15.15	**Session 3:** *Inducing Regular Grammars Using Recurrent Neural Networks* Mor Cohen, Avi Caciularu, Idan Rejwan and Jonathan Beran *Using Sequence to Sequence Neural Networks for Solving Similar Mathematical Problems* Ali Davody and Mihai Sebastian Baba *Refining Manually-Designed Symbol Grounding and High-Level Planning by Policy Gradients* Takuya Hiraoka, Takashi Onishi and Yoshimasa Tsuruoka *Spatio-temporal awareness for wireless telecommunicaiton networks* H. Joe Steinhauer, Tove Helldin and Gunnar Mathiason
15.15 - 15.30	Discussion
	Coffee Break 15.30 – 16.00
16.00 - 16.30	**Organiser talk:** *Integrating Qualitative Spatial Reasoning and Learning: Prospects and Problems* Diedrich Wolter
16.30 - 16.40	Discussion
16.40 - 17.30	**Invited talk:** *Computational modeling of complex AI systems that learn and think* Kristian Kersting
17.30 - 17.40	Discussion
17.40 - 18.10	**Organiser talk:** *Learning or Reasoning? Identifying Problems Where One Is Not Enough* Jochen Renz
18.10 - 18.30	Discussion and closing

Table of Contents

Challenges in Real-World (Spatio-)Temporal Data: Case Studies in Anomaly Detection and Sports

Jesse Davis

KU Leuven
Belgium

Abstract

An unifying theme in my research is the analysis of real-world (spatio-) temporal data. We investigate data generated from a variety of different sources such as sports matches, (recreational) athletes, retail stores, and airplanes among others. On the one hand, these domains are characterized by the presence of domain knowledge that is crucial to consider when designing solutions. On the other hand, they give rise to very rich and complicated data that confront an analyst with a variety of challenges such as the lack of ground truth labels, the need to construct relevant features, and changing contexts. In this talk, I will highlight some of the most important challenges that arise in this setting and how we have tackled them by discussing two different applications. In the first part of the talk, I will discuss the problem of attempting to identify anomalies in resource usage data from a retail environment. In the second part of the talk, I will describe our efforts to analyze data arising from sports matches.

Computational modeling of complex AI systems that learn and think

Kristian Kersting

TU Darmstadt
Darmstad, Germany

Abstract

Our minds make inferences that appear to go far beyond standard machine learning. Whereas people can learn richer representations and use them for a wider range of learning tasks, machine learning algorithms have been mainly employed in a stand-alone context, constructing a single function from a table of training examples. In this talk, I shall touch upon a view on machine learning and AI that can help capturing these human learning aspects by combining high-level languages and databases with statistical learning, optimisation, and deep learning. High-level features such as relations, quantifiers, functions, and procedures provide declarative clarity and succinct characterisations of the data science problem at hand. This helps reducing the cost of modelling and solving it. Putting deep probabilistic learning into the machine learning stack, it even paves the way towards one of my dreams, the automatic data scientist an AI that makes data analysis and reporting accessible to a broader audience of non-experts in machine learners.

This talk is based on joint works with many people such as Vaishak Belle, Carsten Binnig, Martin Grohe, Zoubin Ghahramani, Samuel Kolb, Parisa Kordjamshidi, Martin Mladenov, Alejandro Molina, Sriraam Natarajan, Robert Peharz, Cristopher Re, Dan Roth, Scott Sanner, Karl Stelzner, Martin Trapp, Isabel Valera, and Antonio Vergari.

Probabilistic logic programming and learning

Angelika Kimmig

Cardiff University
Cardiff, UK

Abstract

Reasoning with relational data, learning, and dealing with uncertainty are central to many aspects of AI. Their combination is studied under a variety of names, and a broad range of languages and tools have been developed. Probabilistic logic programming achieves this combination by extending the representation and reasoning capabilities of logic programming to settings with uncertain data. This talk provides a gentle introduction to the field, and also touches upon applications and challenges.

Meta-Interpretive Learning: Achievements and Challenges

Stephen H. Muggleton

Imperial College London
London, UK

Abstract

Meta-Interpretive Learning (MIL) is a recent Inductive Logic Programming technique aimed at supporting learning of recursive definitions. A powerful and novel aspect of MIL is that when learning a predicate definition it automatically introduces sub-definitions, allowing decomposition into a hierarchy of reuseable parts. MIL is based on an adapted version of a Prolog meta-interpreter. Normally such a meta-interpreter derives a proof by repeatedly fetching first-order Prolog clauses whose heads unify with a given goal. By contrast, a meta-interpretive learner additionally fetches higher-order meta-rules whose heads unify with the goal, and saves the resulting meta-substitutions to form a program. This talk will overview theoretical and implementational advances in this new area including the ability to learn Turing computabale functions within a constrained subset of logic programs, the use of probabilistic representations within Bayesian meta-interpretive and techniques for minimising the number of meta-rules employed. The talk will also summarise applications of MIL including the learning of regular and context-free grammars, learning from visual representations with repeated patterns, learning string transformations for spreadsheet applications, learning and optimising recursive robot strategies and learning tactics for proving correctness of programs. The talk will conclude by pointing to the many challenges which remain to be addressed within this new area.

Reasoning at a Distance by Way of Conceptual Metaphors and Blends

Marco Schorlemmer

IIIA - CSIC
Barcelona, Spain

Abstract

Cognitive scientists of the embodied cognition tradition have been providing evidence that a large part of our creative reasoning and problem-solving processes are carried out by means of conceptual metaphor and blending, grounded on our bodily experience with the world. In this talk I shall aim at fleshing out a mathematical model that has been proposed in the last decades for expressing and exploring conceptual metaphor and blending with greater precision than has previously been done. In particular, I shall focus on the notion of aptness of a metaphor or blend and on the validity of metaphorical entailment. Towards this end, I shall use a generalisation of the category-theoretic notion of colimit for modelling conceptual metaphor and blending in combination with the idea of reasoning at a distance as modelled in the Barwise-Seligman theory of information flow. I shall illustrate the adequacy of the proposed model with an example of creative reasoning about space and time for solving a classical brain-teaser. Furthermore, I shall argue for the potential applicability of such mathematical model for ontology engineering, computational creativity, and problem-solving in general.

Effective probabilistic logical reasoning in continuous domains

Vaishak Belle

University of Edinburgh
Edinburgh, UK

Abstract

Weighted model counting (WMC) is the problem of com- puting the mass of a function over the set of models of a propositional theory and lies at the heart of probabilistic ar- tificial intelligence, where a core issue is to quantify uncer- tainty over logically-structured worlds. Many state-of-the-art algorithms dealing with discrete Bayesian networks, factor graphs, probabilistic programs , and probabilistic databases reduce their inference problem to a WMC computation. While a typical WMC inference task is to compute the partition functions and marginals of factored probability distributions, it has also been used as a subroutine for more general tasks such as automated planning.

In this talk, we report on a new computational abstraction called weighted model integration that extends WMC to continuous and mixed discrete continuous domains. We discuss various strategies for solving the task effectively.

Towards a reconciliation between reasoning and learning A position paper

Didier Dubois[1] and **Henri Prade**[1,2]
1. IRIT, University of Toulouse, France
2. QCIS, University of Technology, Sydney, Australia
{dubois,prade}@irit.fr

Abstract

The paper first examines the contours of artificial intelligence (AI) at its beginnings, more than sixty years ago, and points out the important place that machine learning already had at that time. The ambition of AI of making machines capable of performing any information processing task that the human mind can do, means that AI should cover the two modes of human thinking: the instinctive (reactive) one and the deliberative one. This also corresponds to the difference between mastering a skill without being able to articulate it and holding some pieces of knowledge that one can use to explain and teach. In case a functional representation applies to a considered AI problem, the respective merits of learning a universal approximation of the function vs. a rule-based representation are discussed, with a view to better draw the contours of AI. Moreover, the paper reviews the relative positions of knowledge and data in reasoning and learning, and advocates the need for bridging the two tasks. Some examples are outlined. The paper is also a plea for a unified view of the various facets of AI as a science.

1 Introduction

What is artificial intelligence (AI) about? What are the research topics that belong to AI? What are the topics that stand outside? In other words, what are the contours of AI? Answers to these questions may have evolved with time, as did the issue of the proper way (if any) of doing AI.

Indeed over time, AI has been successively dominated by logical approaches (until the mid 1990's) giving birth to the so-called "symbolic AI", then by (Bayesian) probabilistic approaches, and since recently by another type of numerical approach, artificial neural networks. This state of facts has contributed to developing antagonistic feelings between different schools of thought, including claims of supremacy of some methods over others, rather than fostering attempts to understand the potential complementarity of approaches.

Moreover, when some breakthrough takes place in some sector of AI such as expert systems in the 1980's, or fuzzy logic in the 1990's (outside mainstream AI), or yet deep learning [LeCun *et al.*, 2015] nowadays, it is presented through its technological achievements rather than its actual scientific results. So we may even - provocatively - wonder: Is AI a science, or just a bunch of engineering tools? In fact, AI has developed over more than sixty years in several directions, and many different tools have been proposed for a variety of purposes. This increasing diversity, rather than being a valuable asset, may be harmful for an understanding of AI as a whole, all the more so as most AI researchers are highly specialized in some area and are largely ignoring the rest of the field.

Besides, beyond the phantasms and fears teased by the phrase '*artificial* intelligence', the meaning of words such as 'intelligence', 'learning', or 'reasoning' has a large spectrum and may refer to quite different facets of human mind activities, which contributes to blur the meaning of what we say when we are using the acronym AI. Starting with 'intelligence', it is useful to remember the dichotomy popularized by [Kahneman, 2011] between two modes of thinking: "System 1" which is fast, instinctive and emotional, while "System 2" is slower, more deliberative, and more logical. See [Raufaste, 2001] for an illustration of similar ideas in the area of radiological diagnosis, where "super-experts" provide correct diagnosis, even on difficult cases, without any deliberation, while "ordinary experts" may hesitate, deliberate on the difficult cases and finally make a wrong diagnosis. Still, a "super-expert" is able to explain to an "ordinary expert" why went wrong and what was important to notice in the difficult cases.

Darwiche [2017] has recently pointed out that what is achieved by deep leaning corresponds to tasks that do not require much deliberation, at least for a top expert, and is far from covering all that may be expected from AI. In other words, the system is mastering skills rather than being also able to elaborate knowledge for thinking and communicating about its skills. This is the difference between an excellent driver (without teaching capability) and a driving instructor.

The intended purpose of this note is to advocate in favor of a unified view of AI both in terms of problems and in terms of methods. The paper is organized as follows. First, in Section 2 a reminder on the history of the early years of AI emphasizes the idea that the diversity of AI has been there from its inception. Then Section 3 first discusses relations between a

functional view and a rule-based view of problems, in relation with "modeling versus explaining" concerns. The main paradigms of AI are then restated and the need for a variety of approaches ranging from logic to probability and beyond is highlighted. Section 4 reviews the roles of knowledge and data both in reasoning and in machine learning. Then, Section 6 points out problems where bridging reasoning and learning might be fruitful. Section 7 calls for a unified view of AI, a necessary condition for letting it become a mature science.

2 A short reminder of the beginnings of AI

To have a better understanding of AI, it may be useful to have a historical view of the emergence of the main ideas underling it [Marquis *et al.*, 2014b; 2014a; Nilsson, 2010]. We only focus here on its beginnings. Still it is worth mentioning that exactly three hundreds years before the expression 'artificial intelligence' was coined, the English philosopher Thomas Hobbes of Malmesbury (1588-1679) described human thinking as a symbolic manipulation of terms similar to mathematical calculation [Hobbes, 1839]. Indeed, he wrote "*Per Ratiocinationem autem intelligo computationem.*" (or in English one year later "*By ratiocination I mean computation*.") The text continues with "*Now to compute, is either to collect the sum of many things that are added together, or to know what remains when one thing is taken out of another. Ratiocination, therefore, is the same with addition and subtraction;*" One page after one reads: "*We must not therefore think that computation, that is, ratiocination, has place only in numbers, as if man were distinguished from other living creatures (which is said to have been the opinion of Pythagoras) by nothing but the faculty of numbering; for magnitude, body, motion, time, degrees of quality, action, conception, proportion, speech and names (in which all the kinds of philosophy consist) are capable of addition and subtraction.*" Such a description appears retrospectively quite consonant with what AI programs are trying to do!

In the late 1940's with the advent of cybernetics [Wiener, 1949], the introduction of artificial neural networks [McCulloch and Pitts, 1943], the principle of synaptic plasticity [Hebb, 1949] and the concept of computing machines [Turing, 1948] lead to the idea of thinking machines with learning capabilities. In 1950, the idea of machine intelligence appeared in a famous paper by Turing [1950], while Shannon [1950] was investigating the possibility of a program playing chess, and the young Zadeh [1950] was already suggesting multiple-valued logic as a tool for the conception of thinking machines.

As it is well-known, the official birthday act of AI corresponds to a research program whose application for getting a financial support, was written in the summer of 1955, and entitled "A proposal for the Dartmouth summer research project on artificial intelligence" (thus putting the name of the new field in the title!); it was signed by the two fathers of AI, John McCarthy (1927-2011), and Marvin Minsky (1927-2016), and their two mentors Nathaniel Rochester (1919-2001) (who designed the IBM 701 computer and was also interested in neural network computational machines), and Claude Shannon (1916-2001) [McCarthy *et al.*, 2006] (in 1950 he was already the founder of digital circuit design theory based on Boolean logic, the founder of information theory, but also the designer of an electromechanical mouse (Theseus) able to search through the corridors of a maze until reaching the target and to acquire and use knowledge from past experience). Then a series of meetings was organized at Dartmouth College (Hanover, New Hampshire, USA) during the summer of 1956. At that time, McCarthy was already interested in symbolic logic representations, while Minsky had already built a neural network learning machine (he was also a friend of Frank Rosenblatt [1958] the inventor of perceptrons).

The interests of the six other participants can be roughly divided into reasoning and learning concerns, they were on the one hand Herbert A. Simon (1916-2001), Allen Newell (1927-1992) [Newell and Simon, 1956] (together authors with John Clifford Shaw (1922-1991) of a program *The Logic Theorist* able to prove theorems in mathematical logic), and Trenchard More [1959] (a logician interested in natural deduction at that time), and on the other hand Arthur Samuel (1901-1990) [1959] (author of programs for checkers, and later chess games), Oliver Selfridge (1926-2008) [1959] (one of the fathers of pattern recognition), and Ray Solomonoff (1926-2009) [1956] (already author of a theory of probabilistic induction).

Interestingly enough, as it can be seen, these ten participants, with different backgrounds ranging from psychology to electrical engineering, physics and mathematics, were already carriers of the large variety of research directions that can still be observed to-date in AI from machine learning to knowledge representation and reasoning.

3 Representing functions and beyond

There are two modes of representation of knowledge, that call be called respectively functional and logical. The first mode consists in building a big function that produces a result when triggered by some input. The second mode consists of separate, possibly related, chunks of explicit knowledge, expressed in some language. The current dominant machine learning paradigm (up to noticeable exceptions) has adopted the functional approach, which ensures impressive successes in tasks requiring reactiveness, at the cost of losing explanatory power. Indeed, we can argue that what is learnt is know-how or skills, rather than knowledge. The other, logical, mode of representation, is much more adapted to encoding articulated knowledge, reasoning and producing explanations via deliberation, but its connection to learning from data is for the most part still in infancy.

A simple starting point for discussing relationships between learning and reasoning is to compare the machineries of a classifier and a rule-based expert system, for diagnosis for instance. In both cases, a functional view may apply. On the one hand, from a set of examples (of inputs and outputs of the function, such as pairs (symptoms, diseases)) one can easily predict the diseases corresponding to a new case via its input symptoms, after learning some function (e.g., using neural nets). On the other hand, one may have a set of expert rules stating that if the values of the inputs are such and such, the global evaluation should be in some subset. Such rules are mimicking the function. If collected from an expert, rules may turn out to be much less successful than the func-

tion learned from data. Clearly, the first view may provide better approximations and does not require the elicitation of expert rules, which is costly. However, the explanatory power will be poor in any case, because it will not be possible to answer "why not" questions and to articulate explanations based on causal relations. On the contrary, if causal knowledge is explicitly represented in the knowledge base, it has at least the merit of offering a basis for explanations (in a way that should be cognitively more appropriate for the end-user). It is moreover well-known that causal information cannot easily be extracted from data: only correlations can be laid bare if no extra information is obtained [Pearl, 2000].

The fuzzy set literature offers early examples of the replacement of an automatic control law by a set of rules. Indeed Zadeh [1973] proposed to use fuzzy expert rules for controlling complex non linear dynamic systems that might be difficult to model using a classical automatic control approach, while skilled humans can do the job. This was rapidly shown to be successful [Mamdani and Assilian, 1975]. The fact of using fuzzy rules, rather standard if-then rules, had the advantage of providing a basis for an interpolation mechanism, when an input was firing several rules to some degree. At that time, although the approach was numerical and quite far from the symbolic logic-based AI mainstream trend in those times, it was perceived as an AI-inspired approach, since it was relying on the representation of expert know-how by chunks of knowledge, rather than on the derivation of a control law from the modeling of the physical system to be controlled (which is the classical control engineering paradigm). After some time, it was soon recognized that fuzzy rules could be learnt rather than obtained from experts, while keeping excellent results thanks to the property of universal approximation possessed by sets of fuzzy rules. Mathematical models of such fuzzy rules are in fact closely related to neural network radial basis functions. But, fuzzy rules thus obtained by learning may become hardly intelligible. This research trend, known under the names of 'soft computing' or 'computational intelligence', thus drifted away from an important AI concern, the explainability power; see [Dubois and Prade, 1998] for a discussion.

The long term ambition of AI is to make machines capable of performing any information processing task the human mind can perform. This certainly includes recognition, identification, decision and diagnosis tasks (including sophisticated ones). They are "System 1" tasks (using Kahneman terminology) as long as we do not need to explain and reason about obtained results. But there are other problems that are not fully of this kind, even if machine learning may also play a role in their solving. Consider for instance the solving of quadratic equations. Even if we could determine, in a bounded domain, by machine learning techniques, whether an equation has zero, one or two solutions and what are their values (with a good approximation) from a large amount of examples, the solving of such equations by discovering their analytical solution(s), via factorization through symbolic calculations, seems to be a more powerful way of handling of the problem (the machine could then teach students).

AI problems cannot be always be viewed in terms of the functional view mentioned above. There are cases where we have not a function, but a multiple-valued function, e.g., finding all the solutions (if any) of a set of constraints. Apart from solving combinatorial problems, tasks such as reasoning about static or dynamical situations, or building action plans, or explaining results, communicating explanations pertaining to machine decisions in a way meaningful way to an end-user, or analyzing arguments and determining their possible weakness, or understanding what is going on in a text, a dialog in natural langage, in an image, a video, or finding relevant information and summarizing it are examples that may require capabilities beyond pure machine learning.

This is why AI, over the years, has developed general representation settings and methods capable of handling large classes of situations, while mastering computation complexity. For this purpose, at least five general paradigms have emerged in AI:

- **Knowledge representation** with symbolic or numerical structured settings for representing knowledge or preferences, such as logical languages, graphical representations like Bayesian networks, or domain ontologies describing taxonomy of concepts. Dedicated settings have been also developed for the representation of temporal or spatial information, of uncertain pieces of information, or of independence relations.
- **Reasoning and decision** Different types of reasoning tasks, beyond classical deduction, have been formalized such as: nonmonotonic reasoning for handling exception-tolerant rules in the presence of incomplete information, or reasoning from inconsistent information, or belief revision, belief updating, information fusion in the presence of conflicts, or formal argumentation handling pros and cons, or yet reasoning directly from data (case-based reasoning, analogical reasoning, interpolation, extrapolation). Models for qualitative (or quantitative) decision from compact representations have been proposed for decision under uncertainty, multiple criteria, or group decisions.
- **General algorithms for problem solving** This covers a panoply of generic tools ranging from heuristic ordered search methods, general problem solver techniques, methods for handling constraints satisfaction problems, to efficient algorithms for classical logic inference (e.g., SAT methods), or for deduction in modal and other non-classical logics.
- **Learning** The word 'learning' also covers different problems, from the classification of new items based on a set of examples (and counter-examples), the induction of general laws describing concepts, the synthesis of a functional relation by regression, the clustering of similar data (separating dissimilar data into different clusters) and the labelling of clusters, to reinforcement learning and to the discovery of regularities in data bases and data mining. Most of these problems can be handled by a variety of methods.
- **Multiple agent AI** Under this umbrella, there are quite different problems such as: the cooperation between human or artificial agents and the organization of tasks for

achieving collective goals, the modeling of BDI (Belief, Desire, Intention) agents, possibly in situations of dialogue (where, e.g., agents, which have different information items at their disposal, do not pursue the same goals, and try to guess the intentions of the other ones), or the study of the emergence of collective behaviors from the behaviors of elementary agents.

4 Reasoning with knowledge or with data

In the above research areas, knowledge and data are often handled separately. In fact, AI traditionally deals with knowledge rather than with data, with the important exception of machine learning, whose aim can sometimes be viewed as changing data into knowledge. Indeed, basic knowledge is obtained from data by induction, while prior background knowledge may help learning machineries. These remarks suggest that the joint handling of knowledge and data is a general issue, and that combining reasoning and learning methods should be seriously considered.

Knowledge-based systems, or ontologies expressed by means of description logics, or yet Bayesian networks, represent background knowledge that is useful to make prediction from facts and data. In these reasoning tasks, knowledge as well as data is often pervaded with uncertainty. This has been extensively investigated.

Data, provided that they are reliable, are positive in nature since their existence manifests the *actual* possibility of what is observed or reported. This contrasts with knowledge that delimit the extent of what is *potentially* possible by specifying what is impossible (which has thus a negative flavor). This is why reasoning from both knowledge and data goes much beyond the application of generic knowledge to factual data as in expert systems. It is is a complex issue, which has received little attention until now [Ughetto *et al.*, 1999].

As pointed out in [Prade, 2016], reasoning directly with data has been much less studied. The idea of similarity naturally applies to data and gives birth to specific forms of reasoning such as case-based reasoning, case-based decision, or even case-based argumentation. "Betweenness" and similarity are at the basis of interpolation mechanisms, while analogical reasoning, which may be both a matter of similarity and dissimilarity provides a mechanism for extrapolation.

Analogical reasoning, in particular analogical proportion-based inference, interpolation and extrapolation are forms of commonsense reasoning that can be applied to symbolic and numerical knowledge, but which might be also useful in machine learning, as recently shown in classification. Analogical proportions are statements of the form "a is to b as c is to d", often denoted $a : b :: c : d$, which expresses that "a differs from b as c differs from d and b differs from a as d differs from c". This translates into a Boolean logical expression [Miclet and Prade, 2009; Prade and Richard, 2013] which is true only for the 6 following patterns $(0,0,0,0)$, $(1,1,1,1)$, $(1,0,1,0)$, $(0,1,0,1)$, $(1,1,0,0)$, and $(0,0,1,1)$ for (a,b,c,d). Note that these patterns are also compatible with the arithmetic difference definition $a-b=c-d$, where $a-b \in \{-1,0,1\}$, which is not a Boolean logic definition. Analogical proportions straightforwardly extends to vectors such as $\vec{a}=(a_1,...,a_n)$, by stating $\vec{a} : \vec{b} :: \vec{c} : \vec{d}$ iff $\forall i \in [1,n],\ a_i : b_i :: c_i : d_i$. The basic analogical inference pattern [Stroppa and Yvon, 2005], is then

$$\frac{\forall i \in \{1,...,p\},\ \ a_i : b_i :: c_i : d_i \text{ holds}}{\forall j \in \{p+1,...,n\},\ \ a_j : b_j :: c_j : d_j \text{ holds}}$$

Thus analogical reasoning amounts to finding completely informed triples $(\vec{a},\vec{b},\vec{c})$ appropriate for inferring the missing value(s) in $\vec{d}$. When there exist several suitable triples, possibly leading to distinct conclusions, one may use a majority vote for concluding. This extends to analogical proportions between numerical values. It has been successfully applied, for Boolean, nominal or numerical attributes, to classification [Miclet *et al.*, 2008; Bounhas *et al.*, 2017] (then the class $cl(\vec{x})$ (viewed as a nominal attribute) is the unique solution, when it exists, such as $cl(\vec{a}) : cl(\vec{b}) :: cl(\vec{c}) : cl(\vec{x})$ holds), and more recently to preference learning [Pirlot *et al.*, 2016; Fahandar and Hüllermeier, 2018]. It has been theoretically established that analogical classifiers *always* yield exact prediction for Boolean affine functions (which includes x-or functions), and only for them [Couceiro *et al.*, 2017b]. Still good results can be obtained in other cases [Couceiro *et al.*, 2018].

Moreover, analogical inequalities [Prade and Richard, 2017] of the form "a is to b at least as much as c is to d" might be useful for describing relations between images, as in [Law *et al.*, 2017].

Besides, the ideas of interpolation and extrapolation closely related to analogical proportion-based inference, which are of crucial importance in many numerical domains, can be applied in symbolic settings in the case of propositional categorization rules, using relations of betweenness and parallelism respectively, with a conceptual spaces semantics [Schockaert and Prade, 2013]; see [Schockaert and Prade, 2011] for an illustration.

5 Learning from incomplete data

The need for reasoning from incomplete, uncertain, vague, or inconsistent information, has led to the development of new approaches beyond logic and probability. Incompleteness is a well-known phenomenon in classical logic. However, many reasoning problems exceed the capabilities of classical logic (initially developed in relation with the foundations of mathematics where statements are true or false, and there is no uncertainty in principle). As for probability theory, single probability distributions, often modeled by Bayesian networks are not fully appropriate for handling incomplete information nor epistemic uncertainty. There are different, but related, frameworks for modeling ill-known probabilities that were developed in the last 50 years by the Artificial Intelligence community at large [Walley, 1996]: belief functions and evidence theory (which may be viewed as a randomization of the set-based approach to incomplete information), imprecise probability theory (which uses convex families of probability functions) and quantitative possibility theory (which is the simplest model since one of the lower and the upper probability bounds is trivial).

The traditional approach for going from data to knowledge is to resort to statistical inferential methods. However, these

methods used to assume data that are precise and in sufficient quantity. The recent concern with big data seems to even strengthen the relevance of probability theory and statistics. However there are a number of circumstances where data is missing or is of poor quality, especially if one tries to collect information for building machines or algorithms supposed to face very complex or unexpected situations (e.g. autonomous vehicles in crowded areas). The concern of Artificial Intelligence for reasoning about partial knowledge has led to a questioning of traditional statistical methods when data is of poor quality.

When data is missing or just imprecise (one then speaks of *coarse data*), statistical methods need to be adapted. In particular the question is whether one wishes to model the observed phenomenon *along with* the limited precision of the observations, or *despite* it. The latter trend comes down to complete the data in some way (using imputation methods). A well-known method that does it is the EM algorithm [Dempster *et al.*, 1977]: starting from a class of parameterized statistical models, the idea is to iteratively construct a precise model that fits the data as much as possible, by generating at each step a precise observation sample in agreement with the incomplete data (using a revision step on the current model) followed by the computation of a new model applying the maximum likelihood method to the last precise sample. This technique, including variants based on belief functions [Denoeux, 2013], has been extensively used in AI for clustering and learning Bayesian nets.

However the obtained result, where by virtue of the algorithm, data has become complete and precise, is not easy to interpret. If we want to be faithful to the data and its imperfections, one way is to build a model that accounts for the imprecision of observations, i.e., a set-valued model. This is the case if a belief function is obtained via maximum likelihood on imprecise observations: one optimizes the *visible likelihood function* [Couso and Dubois, 2018]. The idea is to cover all precise models that could have been derived, had the data been precise. Imprecise models are useful to lay bare ignorance when it is present, so as to urge finding more data, but it may be problematic for decision problems, when we have to act despite ignorance.

So we must try to optimize the likelihood function based on the actual values behind the imprecise observations. But such likelihood functions (either pertaining to the real phenomenon under study, or to both the phenomenon and its measurement) are ill-known in the case of coarse data [Couso and Dubois, 2018]. In that case we are bound

- to make assumptions on the measurement process so as to create a tight link between the hidden likelihood function pertaining to the outcomes of the real phenomenon, and the visible likelihood of the imprecise observations (for instance the CAR (coarsening at random) assumption [Heitjan and Rubin, 1991], or the superset assumption [Hüllermeier and Cheng, 2015]. In that case, the coarseness of the data can be in some sense ignored. See [Jaeger, 2005] for a general discussion.
- or to pick a suitable hidden likelihood function among the ones compatible with the imprecise data, for instance using an optimistic maximax approach that tends to disambiguate the data [Hüllermeier, 2014], or a maximin (robust) approach that favors statistical models with high variance to cover the partial ignorance [Guillaume *et al.*, 2017]. In that case, the measurement process is ignored.

See [Couso *et al.*, 2017] for more discussions about such methods for statistical inference with poor quality data.

Besides, a new cumulative entropy function [Serrurier and Prade, 2013] has been proposed, which is based on confidence intervals of frequency estimates that together considers the entropy of the probability distribution and the uncertainty around the estimation of its parameters. Such a function takes advantage of the ability of a possibility distribution to upper bound a family of probabilities previously estimated from a limited set of examples and of the link between possibilistic specificity order and entropy [Dubois and Hüllermeier, 2007]. This has been applied to the learning of decision trees to cope with the fact that as we go deeper downward the tree, the examples become rarer and the faithfulness of entropy decreases [Serrurier and Prade, 2015].

Finally, let us also mention that the fact that we may have to work with incomplete relational data and that knowledge may also be uncertain motivates the use of a new probabilistic programming language first called "Probabilistic Similarity Logic", and then "Probabilistic Soft Logic" (PSL, for short) where each ground atom in a rule has a truth value in $[0, 1]$, and which uses the Łukasiewicz t-norm and co-norm to handle the fuzzy logical connectives [Fakhraei *et al.*, 2013; Farnadi *et al.*, 2014; Bach *et al.*, 2017]. We are close to the representation concerns of fuzzy answer set programs [Mushthofa *et al.*, 2015]. Besides, there is a need for combining symbolic reasoning with the subsymbolic vector representation of neural networks in order to use gradient descent for training the neural network to infer facts from an incomplete knowledge base, using similarity between vectors [Rocktäschel and Riedel, 2017; Cohen *et al.*, 2017; Cohen, 2016].

6 New representation formats in learning

Machine learning may find some advantages to use advanced representation formats as target languages, such as weighted logics (probabilistic, fuzzy, possibilistic, prioritized, etc). For instance, qualitative possibility theory extends classical logic by attaching lower bounds of necessity degrees and captures nonmonotonic reasoning, while generalized possibilistic logic [Dubois *et al.*, 2017] is more powerful and can capture answer-set programming, or reason about the ignorance of an agent. Can such kinds of qualitative uncertainty modeling, or yet fuzzy or uncertain description logics, uncertainty representation formalisms, weighted logics (Markov logic, probabilistic logic programs, multi-valued logics, possibilistic logic, etc.), be used more extensively in machine learning? Some answers can be found in [Serrurier and Prade, 2007; Kuzelka *et al.*, 2015; 2016; 2017]. This also raises the question of extending version space learning [Mitchell, 1979] to such new representation schemes [Hüllermeier, 2003; Prade and Serrurier, 2008; Prade *et al.*, 2009a].

If-then rules are a popular representation format in relational learning [Rückert and De Raedt, 2008]. But, can we

reason with knowledge extracted from data in a consistent way? For instance, can rules having exceptions extracted from data be processed by a nonmonotonic inference system yielding new default rules? How can we insure that these new rules are still agreeing with the data? The problem is then to extract genuine default rules that hold in a Boolean database. It does not just amount to mine association rules with a sufficiently high confidence level. We have to guarantee that any new default rules that are deducible from the set of extracted default rules are indeed valid with respect to the database. For doing that, we need a probabilistic semantics for nonmonotonic inference. It has been shown [Benferhat *et al.*, 1999] that default rules of the form "if p then generally q", denoted $p|\sim q$, where $|\sim$ obey the postulates of preferential inference [Kraus *et al.*, 1990] have both i) a possibilistic semantics expressed by the constraint $\Pi(p \wedge q) > \Pi(p \wedge \neg q)$, for any max-decomposable possibility measure Π ($\Pi(p \vee q) = \max(\Pi(p), \Pi(q))$), and a probabilistic semantics expressed by the constraint $Prob(p \wedge q) > Prob(p \wedge \neg q)$ or any *big-stepped probability* $Prob$. This is a very special kind of probability such that if $p_1 > p_2 > ... > p_{n-1} \geq p_n$ (where p_i is the probability of one of the n possible worlds), the following inequalities hold $\forall i = 1, n-1,\ p_i > \Sigma_{j=i,n}\, p_j$. Then, one can infer a new default $p|\sim q$ from a set of defaults $\Delta = \{p_k|\sim q_k | k = 1, K\}$ if and only if the constraints modeling Δ entail the constraints modeling $p|\sim q$. Thus, extracting defaults amounts to looking for big-stepped probabilities, by clustering lines describing items in Boolean tables, so as to find default rules, see [Benferhat *et al.*, 2003] for details. Then the rules discovered are genuine default rules that can be reused in a nonmonotonic inference system, and can be encoded in possibilistic logic (assuming rational monotony).

Multiple threshold rules, i.e., selection rules of the form "if $x_1 \geq \alpha_1$ and $\cdots$ $x_j \geq \alpha_j$ and $\cdots$ then $y \geq \gamma$" (or deletion rules of the form 'if $x_1 \leq \beta_1$ and $\cdots$ $x_j \leq \beta_j$ and $\cdots$ then $y \geq \delta$") is another format of interest in ordinal classification / regression problems [Greco *et al.*, 2006; Blaszczyski *et al.*, 2011]. Assuming that data are made of a collection of pairs $(x^k, y_k), k = 1, ..., N$ where x^k is a tuple $(x_1^k, ..., x_n^k)$ of local evaluations item k, and that y increases with the x_i's in the broad sense, it is of interest of describing the data with such rules of various lengths. It has been noticed [Greco *et al.*, 2004; Dubois *et al.*, 2014] that, once the numerical data are normalized between 0 and 1, rules where all (non trivial) thresholds are equal can be represented by Sugeno integrals (a generalization of weighted min and weighted max, which is a qualitative counterpart of Choquet integrals [Grabisch and Labreuche, 2010]). Moreover, it has been shown recently [Couceiro *et al.*, 2017a] that generalized forms of Sugeno integrals are able to describe a global (increasing) function, taking values on a finite linearly ordered scale, under the form of general thresholded rules. Another approach, in the spirit of version space approach, provides a bracketing of an increasing function by means of a pair of Sugeno integrals [Prade *et al.*, 2009b; 2009a].

There are other types of rules that may be of interest as a representation format, for instance, gradual rules, which express statements of the form "the more x is A, the more y is B, where A, and B are gradual properties modeled by fuzzy sets [Serrurier *et al.*, 2007; Nin *et al.*, 2010]. It is also worth remembering that other types of fuzzy rules may provide a rule-based interpretation [d'Alché-Buc *et al.*, 1994] for neural nets. With respect to neural nets, let us also mention a nonmonotonic inference view [Balkenius and Gärdenfors, 1991; Gärdenfors, 1991].

7 Conclusion

Knowledge representation and reasoning on the one hand, and machine learning on the other hand, have been developed largely as independent research trends in artificial intelligence in the last three decades. Still reasoning and learning are two basic capabilities of the human mind that may interact. Similarly the two corresponding AI research areas may benefit from mutual exchanges. Current learning methods derive know-how from data in the form of complex functions involving many tuning parameters, but they should also aim at producing articulated knowledge, so that knowledge repositories, storing interpretable chunks of information, could be fed from data. More precisely, a number of logical-like formalisms, whose explanatory capabilities could be exploited, have been developed in the last 30 years (non-monotonic logics, modal logics, logic programming, probabilistic and possibilistic logics, many-valued logics, etc.) that could be used as target languages for learning techniques, without restricting to first-order logic, or to Bayes nets.

Interfacing classifiers with human users may require some ability to provide high level explanations about recommendations or decisions that are understandable by an end-user. Reasoning methods should handle knowledge and information extracted from data. The joint use of (supervised or unsupervised) machine learning techniques and of inference machineries raises new issues. There is a number of other points, worth mentioning, which have not be addressed in the above discussions:

- *Teachability* A related issue is more generally how to move from machine learning models to knowledge that be communicated to humans, about the way the machine proceeds when solving problems.
- *Using knowledge* Another issue is a more systematic exploitation of symbolic background knowledge in machine learning devices. Can prior causal knowledge help exploiting data and getting rid of spurious correlations? Can an argumentation-based view of learning be developed?
- *Representation learning* Data representation impacts the performance of machine learning algorithms [Bengio *et al.*, 2013]. In that respect, what may be, for instance, the role of vector space embeddings, or conceptual spaces?
- *Unification of learning paradigms* Would it be possible to bridge learning paradigms from transduction to inductive logic programming? Even including formal concept analysis, or rough set theory?

This paper has especially advocated the interest of a cooperation between two basic areas of AI: knowledge representation and reasoning on the one hand and machine learning

on the other hand, reflecting the natural cooperation between two modes, reactive and deliberative of human intelligence. It is also a plea for maintaining a unified view of AI, all facets of which have been present from the very beginning, as recalled in Section 2 of this paper. It is time that AI comes of age as a genuine science, which means stopping unproductive rivalries between different approaches, and fostering a better shared understanding of the basics of AI through open-minded studies bridging sub-areas in a constructive way. In the same spirit, a plea for a unified view of computer science can be found in [Bajcsy and Reynolds, 2002]. Mixing, bridging, hybridizing advanced ideas in knowledge representation, reasoning, and machine learning or data mining should renew basic research in AI and contribute in the long term to a more unified view of AI methodology.

8 Acknowledgements

The authors thank Gilles Richard, Steven Schockaert, and Mathieu Serrurier for useful exchanges on some of the issues surveyed in this paper.

References

[Bach *et al.*, 2017] S. H. Bach, M. Broecheler, B. Huang, and L. Getoor. Hinge-loss Markov random fields and Probabilistic Soft Logic. *J. of Machine Learning Research*, 18:109:1–109:67, 2017.

[Bajcsy and Reynolds, 2002] R. Bajcsy and C. W. Reynolds. Computer science: the science of and about information and computation. *Commun. ACM*, 45(3):94–98, 2002.

[Balkenius and Gärdenfors, 1991]C. Balkenius and P. Gärdenfors. Nonmonotonic inferences in neural networks. In *Proc. 2nd Int. Conf. on Princip. of Knowl. Represent. and Reas. (KR'91). Cambridge, MA*, pages 32–39, 1991.

[Benferhat *et al.*, 1999] S. Benferhat, D. Dubois, and H. Prade. Possibilistic and standard probabilistic semantics of conditional knowledge bases. *J. Log. Comput.*, 9(6):873–895, 1999.

[Benferhat *et al.*, 2003] S. Benferhat, D.Dubois, S. Lagrue, and H. Prade. A big-stepped probability approach for discovering default rules. *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, 11(Supplement-1):1–14, 2003.

[Bengio *et al.*, 2013] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspec-tives. *IEEE Trans. P. A. M. I.*, 35:1798–1828, 2013.

[Blaszczyski *et al.*, 2011] J. Blaszczyski, R. Slowiski, and M. Szelag. Sequential covering rule induction algorithm for variable consistency rough set approaches. *Information Sciences*, 181(5):987 – 1002, 2011.

[Bounhas *et al.*, 2017] M. Bounhas, H. Prade, and G. Richard. Analogy-based classifiers for nominal or numerical data. *Int. J. Approx. Reasoning*, 91:36–55, 2017.

[Cohen *et al.*, 2017] W. W. Cohen, F. Yang, and K. Mazaitis. TensorLog: Deep learning meets probabilistic DBs. *CoRR*, abs/1707.05390, 2017.

[Cohen, 2016] William W. Cohen. TensorLog: A differentiable deductive database. *CoRR*, abs/1605.06523, 2016.

[Couceiro *et al.*, 2017a] M. Couceiro, D. Dubois, H. Prade, and A. Rico. Enhancing the expressive power of Sugeno integrals for qualitative data analysis. In *Proc. 10th Conf. Europ. Soc. Fuz. Log.& Techn. (EUSFLAT'17), Warsaw, Vol. 1*, AISC vol. 641, pages 534–547. Springer, 2017.

[Couceiro *et al.*, 2017b] M. Couceiro, N. Hug, H. Prade, and G. Richard. Analogy-preserving functions: A way to extend boolean samples. In *Proc. 26th Int. Joint Conf. on Artificial Intelligence, (IJCAI'17), Melbourne, Aug. 19-25*, pages 1575– 1581, 2017.

[Couceiro *et al.*, 2018] M. Couceiro, N. Hug, H. Prade, and G. Richard. Behavior of analogical inference w.r.t. Boolean functions. In *Proc. 27th Int. Joint Conf. on Artificial Intelligence, (IJCAI'18), Stockholm, July. 13-19*, 2018.

[Couso and Dubois, 2018] Inés Couso and Didier Dubois. A general framework for maximizing likelihood under incomplete data. *Int. J. Approx. Reasoning*, 93:238–260, 2018.

[Couso *et al.*, 2017] Inés Couso, Didier Dubois, and Eyke Hüllermeier. Maximum likelihood estimation and coarse data. In *Scalable Uncertainty Management - Proc 11th Int. Conf., SUM 2017*, volume 10564 of *Lecture Notes in Computer Science*, pages 3–16. Springer, 2017.

[d'Alché-Buc *et al.*, 1994] F. d'Alché-Buc, V. Andrés, and J.-P. Nadal. Rule extraction with fuzzy neural network. *Int. J. Neural Syst.*, 5(1):1–11, 1994.

[Darwiche, 2017] A. Darwiche. Human-level intelligence or animal-like abilities? *CoRR*, abs/1707.04327, 2017.

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[Denoeux, 2013] Thierry Denoeux. Maximum likelihood estimation from uncertain data in the belief function framework. *IEEE Trans. Knowl. Data Eng.*, 25(1):119–130, 2013.

[Dubois and Hüllermeier, 2007] D. Dubois and E. Hüllermeier. Comparing probability measures using possibility theory: A notion of relative peakedness. *Int. J. Approx. Reasoning*, 45(2):364–385, 2007.

[Dubois and Prade, 1998] D. Dubois and H. Prade. Soft computing, fuzzy logic, and artificial intelligence. *Soft Comput.*, 2(1):7–11, 1998.

[Dubois *et al.*, 2014] D. Dubois, H. Prade, and A. Rico. The logical encoding of Sugeno integrals. *Fuzzy Sets and Systems*, 241:61–75, 2014.

[Dubois *et al.*, 2017] D. Dubois, H. Prade, and S. Schockaert. Generalized possibilistic logic: Foundations and applications to qualitative reasoning about uncertainty. *Artif. Intell.*, 252:139–174, 2017.

[Fahandar and Hüllermeier, 2018] M. A. Fahandar and E. Hüllermeier. Learning to rank based on analogical reasoning. In *Proc. 32th Nat Conf. on Artificial Intelligence (AAAI'18), New Orleans, Feb. 2-7*, 2018.

[Fakhraei *et al.*, 2013] S. Fakhraei, L. Raschid, and L. Getoor. Drug-target interaction prediction for drug repurposing with probabilistic similarity logic. In *SIGKDD 12th Int. Workshop on Data Mining in Bioinformatics (BIOKDD)*. ACM, 2013.

[Farnadi *et al.*, 2014] G. Farnadi, S. H. Bach, M.-F. Moens, L. Getoor, and M. De Cock. Extending PSL with fuzzy quantifiers. In *Statistical Relational Artificial Intelligence, Papers from the 2014 AAAI Workshop, Québec City, Québec, Canada, July 27, 2014*, volume WS-14-13 of *AAAI Workshops*, pages 35–37, 2014.

[Gärdenfors, 1991] P. Gärdenfors. Nonmonotonic inference, expectations, and neural networks. In R. Kruse and P. Siegel, editors, *Proc. Europ. Conf. on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQA), Marseille, Oct. 15-17*, volume 548 of *LNCS*, pages 12–27. Springer, 1991.

[Grabisch and Labreuche, 2010] M. Grabisch and Ch. Labreuche. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. *Annals of Oper. Res.*, 175:247–286, 2010.

[Greco *et al.*, 2004] S. Greco, B. Matarazzo, and R. Slowinski. Axiomatic characterization of a general utility function and its particular cases in terms of conjoint measurement and rough-set decision rules. *Europ.J. of Operational Research*, 158(2):271–292, 2004.

[Greco *et al.*, 2006] S. Greco, M. Inuiguchi, and R. Slowinski. Fuzzy rough sets and multiple-premise gradual decision rules. *Int. J. Approx. Reasoning*, 41(2):179–211, 2006.

[Guillaume *et al.*, 2017] Romain Guillaume, Inés Couso, and Didier Dubois. Maximum likelihood with coarse data based on robust optimisation. In *Proceedings of the Tenth International Symposium on Imprecise Probability: Theories and Applications.*, volume 62 of *Proceedings of Machine Learning Research*, pages 169–180. PMLR, 2017.

[Hebb, 1949] D. O. Hebb. *The Organization of Behaviour*. John Wiley & Sons, 1949.

[Heitjan and Rubin, 1991] Daniel F. Heitjan and Donald B. Rubin. Ignorability and coarse data. *Ann. Statist.*, 19(4):2244–2253, 1991.

[Hobbes, 1839] T. Hobbes. Elements of philosophy, the first section, concerning body. In W. Molesworth, editor, *The English works of Thomas Hobbes of Malmesbury*, volume 1. John Bohn, London, 1839. English transl. of "Elementa Philosophiae I. De Corpore", 1655.

[Hüllermeier and Cheng, 2015] Eyke Hüllermeier and Weiwei Cheng. Superset learning based on generalized loss minimization. In *Machine Learning and Knowledge Discovery in Databases - Proc. Eur. Conf., ECML PKDD 2015, Part II*, volume 9285 of *Lecture Notes in Computer Science*, pages 260–275. Springer, 2015.

[Hüllermeier, 2003] E. Hüllermeier. Inducing fuzzy concepts through extended version space learning. In T. Bilgiç, B. De Baets, and O. Kaynak, editors, *Proc. 10th Int. Fuzzy Systems Association World Cong. (IFSA'03), Istanbul, June 30 - July 2*, volume 2715 of *LNCS*, pages 677–684. Springer, 2003.

[Hüllermeier, 2014] Eyke Hüllermeier. Learning from imprecise and fuzzy observations: Data disambiguation through generalized loss minimization. *Int. J. Approx. Reasoning*, 55(7):1519–1534, 2014.

[Jaeger, 2005] Manfred Jaeger. Ignorability in statistical and probabilistic inference. *J. Artif. Intell. Res.*, 24:889–917, 2005.

[Kahneman, 2011] D. Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, 2011.

[Kraus *et al.*, 1990] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[Kuzelka *et al.*, 2015] O. Kuzelka, J. Davis, and S. Schockaert. Encoding Markov logic networks in possibilistic logic. In M. Meila and T. Heskes, editors, *Proc. 31st Conf. on Uncertainty in Artificial Intelligence (UAI'15), July 12-16, Amsterdam*, pages 454–463. AUAI Press, 2015.

[Kuzelka *et al.*, 2016] O. Kuzelka, J. Davis, and S. Schockaert. Learning possibilistic logic theories from default rules. In S. Kambhampati, editor, *Proc. 25th Int. Joint Conf. on Artificial Intelligence (IJCAI'16), New York, 9-15 July*, pages 1167–1173, 2016.

[Kuzelka *et al.*, 2017] O. Kuzelka, J. Davis, and S. Schockaert. Induction of interpretable possibilistic logic theories from relational data. In C. Sierra, editor, *Proc. 26th Int. Joint Conf. on Artificial Intelligence, IJCAI 2017, Melbourne, Aug. 19-25*, pages 1153–1159, 2017.

[Law *et al.*, 2017] M.T. Law, N. Thome, and M. Cord. Learning a distance metric from relative comparisons between quadruplets of images. *Int. J. of Computer Vision*, 121(1):65–94, 2017.

[LeCun *et al.*, 2015] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[Mamdani and Assilian, 1975] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. of Man-Mach. Stu.*, 7:1–13, 1975.

[Marquis *et al.*, 2014a] P. Marquis, O. Papini, and H. Prade. Eléments pour une histoire de l'intelligence artificielle. In *Panorama de l'Intelligence Artificielle. Ses Bases Méthodologiques, ses Développements, Vol. I*, pages 1–39. Cépaduès, 2014.

[Marquis *et al.*, 2014b] P. Marquis, O. Papini, and H. Prade. Some elements for a prehistory of Artificial Intelligence in the last four centuries. In *Proc. 21st Europ. Conf. on Artif. Intell. (ECAI'14), Prague, 609-614*. IOS Press, 2014.

[McCarthy *et al.*, 2006] J. McCarthy, M. Minsky, N. Roch-ester, and C. E. Shannon. A proposal for the Dartmouth summer research project on artificial intelligence, August 31, 1955. *The AI Magazine*, 27(4):12–14, 2006.

[McCulloch and Pitts, 1943] W. S. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[Miclet and Prade, 2009] L. Miclet and H. Prade. Handling analogical proportions in classical logic and fuzzy logics settings. In *Proc. 10th Eur. Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09)*, pages 638–650. Springer, LNCS 5590, 2009.

[Miclet *et al.*, 2008] L. Miclet, S. Bayoudh, and A. Delhay. Analogical dissimilarity: definition, algorithms and two experiments in machine learning. *JAIR, 32*, pages 793–824, 2008.

[Mitchell, 1979] T. Mitchell. *Version spaces: An approach to concept learning.* 1979. Ph.D. thesis, Stanford University.

[More, 1959] T. More. On the construction of Venn diagrams. *J. Symbolic Logic*, 24(4):303–304, 1959.

[Mushthofa *et al.*, 2015] M. Mushthofa, S. Schockaert, and M. De Cock. Solving disjunctive fuzzy answerset programs. In F. Calimeri, G. Ianni, and M. Truszczynski, editors, *Proc. 13th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'15), Lexington, Sept. 27-30*, volume 9345 of *LNCS*, pages 453–466. Springer, 2015.

[Newell and Simon, 1956] A. Newell and H. A. Simon. *The logic theory machine. A complex information processing system*. The Rand Corporation, Santa Monica, Ca, 1956. Report P-868, 15 June 1956; Proc. IRE Trans. on Information Theory (IT-2), Sept. 1956, pp. 61-79.

[Nilsson, 2010] N. J. Nilsson. *The Quest for Artificial Intelligence : A History of Ideas and Achievements*. Cambridge University Press, 2010.

[Nin *et al.*, 2010] J. Nin, A. Laurent, and P. Poncelet. Speed up gradual rule mining from stream data! A b-tree and owa-based approach. *J. Intell. Inf. Syst.*, 35(3):447–463, 2010.

[Pearl, 2000] J. Pearl. *Causality*. Cambridge University Press, 2000. 2nd revised edition, 2009.

[Pirlot *et al.*, 2016] M. Pirlot, H. Prade, and G. Richard. Completing preferences by means of analogical proportions. In V. Torra, Y. Narukawa, G. Navarro-Arribas, and C. Yañez, editors, *Proc. 13th Int. Conf. on Modeling Decisions for Artificial Intelligence (MDAI'16), Sant Julià de Lòria, Andorra, Sept. 19-21*, volume 9880 of *LNAI*, pages 135–147. Springer, 2016.

[Prade and Richard, 2013] H. Prade and G. Richard. From analogical proportion to logical proportions. *Logica Universalis*, 7(4):441–505, 2013.

[Prade and Richard, 2017] H. Prade and G. Richard. Analogical inequalities. In A. Antonucci, L. Cholvy, and O. Papini, editors, *Proc. 14th Europ. Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'17), Lugano, July 10-14*, volume 10369 of *LNCS*, pages 3–9. Springer, 2017.

[Prade and Serrurier, 2008] H. Prade and M. Serrurier. Bipolar version space learning. *Int. J. Intell. Syst.*, 23(10):1135–1152, 2008.

[Prade *et al.*, 2009a] H. Prade, A. Rico, and M. Serrurier. Elicitation of Sugeno integrals: A version space learning perspective. In J. Rauch, Z. W. Ras, P. Berka, and T. Elomaa, editors, *Proc. 18th Int. Symp. on Foundations of Intelligent Systems (ISMIS'09), Prague, Sept. 14-17*, volume 5722 of *LNCS*, pages 392–401. Springer, 2009.

[Prade *et al.*, 2009b] H. Prade, A. Rico, M. Serrurier, and E. Raufaste. Elicitating Sugeno integrals:Methodology and a case study. In C. Sossai and G. Chemello, editors, *Proc. 10th Eur. Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09), Verona, July 1-3*, volume 5590 of *LNCS*, pages 712–723. Springer, 2009.

[Prade, 2016] H. Prade. Reasoning with data - A new challenge for AI? In S. Schockaert and P. Senellart, editors, *Proc. 10th Int. Conf. on Scalable Uncert. Mgmt. (SUM'16), Nice, 274-288*, LNCS 9858. Springer, 2016.

[Raufaste, 2001] E. Raufaste. *Les Mécanismes Cognitifs du Diagnostic Médical : Optimisation et Expertise*. Presses Universitaires de France (PUF), Paris, 2001.

[Rocktäschel and Riedel, 2017] T. Rocktäschel and S. Riedel. End-to-end differentiable proving. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Proc. 31st Annual Conf. on Neural Information Processing Systems (NIPS'17), 4-9 Dec. Long Beach*, pages 3791–3803, 2017.

[Rosenblatt, 1958] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[Rückert and De Raedt, 2008] U. Rückert and L. De Raedt. An experimental evaluation of simplicity in rule learning. *Artif. Intell.*, 172(1):19–28, 2008.

[Samuel, 1959] A. Samuel. Some studies in machine learning using the game of checkers. *IBM J.*, 3:210–229, 1959.

[Schockaert and Prade, 2011] S. Schockaert and H. Prade. Interpolation and extrapolation in conceptual spaces: A case study in the music domain. In S. Rudolph and C. Gutiérrez, editors, *Proc. 5th Int. Conf. on Web Reasoning and Rule Systems (RR'11), Galway, Aug. 29-30*, volume 6902 of *LNCS*, pages 217–231. Springer, 2011.

[Schockaert and Prade, 2013] S. Schockaert and H. Prade. Interpolative and extrapolative reasoning in propositional theories using qualitative knowledge about conceptual spaces. *Artif. Intell.*, 202:86–131, 2013.

[Selfridge, 1959] O. G. Selfridge. Pandemonium: A paradigm for learning. In D. V. Blake and A. M. Uttley, editors, *Symp. on Mechanisation of Thought Processes, London, Nov. 24-27, 1958*, pages 511–529, 1959.

[Serrurier and Prade, 2007] M. Serrurier and H. Prade. Introducing possibilistic logic in ILP for dealing with exceptions. *Artif. Intell.*, 171(16-17):939–950, 2007.

[Serrurier and Prade, 2013] M. Serrurier and H. Prade. An informational distance for estimating the faithfulness of a possibility distribution, viewed as a family of probability distributions, with respect to data. *Int. J. Approx. Reasoning*, 54(7):919–933, 2013.

[Serrurier and Prade, 2015] M. Serrurier and H. Prade. Entropy evaluation based on confidence intervals of frequency estimates : Application to the learning of decision trees. In F. R. Bach and D. M. Blei, editors, *Proc. 32nd Int. Conf. on Machine Learning, (ICML'15), Lille, July 6-11*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1576–1584, 2015.

[Serrurier *et al.*, 2007] M. Serrurier, D. Dubois, H. Prade, and T. Sudkamp. Learning fuzzy rules with their implication operators. *Data Knowl. Eng.*, 60(1):71–89, 2007.

[Shannon, 1950] C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine (7th series)*, XLI (314):256–275, 1950.

[Solomonoff, 1956] R. J. Solomonoff. An inductive inference machine. Technical report, Tech. Res. Group, New York City, 1956.

[Stroppa and Yvon, 2005] N. Stroppa and F. Yvon. Analogical learning and formal proportions: Definitions and methodological issues. Technical report, June 2005.

[Turing, 1948] A. M. Turing. Intelligent machinery. Technical report, National Physical Laboratory, London, 1948. Reprinted in: Machine Intelligence, Vol. 5, Edinburgh University Press, 3-23, 1969.

[Turing, 1950] A.M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[Ughetto *et al.*, 1999] L. Ughetto, D. Dubois, and H. Prade. Implicative and conjunctive fuzzy rules - A tool for reasoning from knowledge and examples. In J. Hendler and D. Subramanian, editors, *Proc. 16th Nat. Conf. on Artificial Intelligence, Orlando, July 18-22*, pages 214–219, 1999.

[Walley, 1996] Peter Walley. Measures of uncertainty in expert systems. *Artif. Intell.*, 83(1):1–58, 1996.

[Wiener, 1949] N. Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. Wiley, 1949.

[Zadeh, 1950] L. A. Zadeh. Thinking machines. a new field in electrical engineering. *Columbia Engineering Quarterly*, 3:12–13 & 30–31, 1950.

[Zadeh, 1973] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Systems, Man, and Cybernetics*, 3(1):28–44, 1973.

A Symbolic Approach for Explaining Errors in Image Classification Tasks

Marjan Alirezaie[1*], **Martin Längkvist**[1*], **Michael Sioutis**[1], **Amy Loutfi**[1],
[1] Center for Applied Autonomous Sensor Systems, Örebro University, Örebro, Sweden
firstname.lastname@oru.se

Abstract

Machine learning algorithms, despite their increasing success in handling object recognition tasks, still seldom perform without error. Often the process of understanding why the algorithm has failed is the task of the human who, using domain knowledge and contextual information, can discover systematic shortcomings in either the data or the algorithm. This paper presents an approach where the process of reasoning about errors emerging from a machine learning framework is automated using symbolic techniques. By utilizing spatial and geometrical reasoning between objects in a scene, the system is able to describe misclassified regions in relation to its context. The system is demonstrated in the remote sensing domain where objects and entities are detected in satellite images.

1 Introduction

Many machine learning algorithms are trained by optimizing a cost function that continuously measures the training errors during learning, and adapts the model parameters in order to minimize these errors. With this approach, the learning algorithms seem to learn from their errors. However, such learning processes differ from what human advisors usually mean by "*learn-from-your-mistakes*", which entails that the learner is able to understand why the errors occurred and conceptualize them by expressing their characteristics. The training process of minimizing a cost function is not aimed towards explaining the errors or describing why such errors have been made, but instead follows the defined rules for parameter updates given by the selected minimization optimization method.

For satellite image classification, a classifier that only uses the RGB channels as input runs the risk of producing a large amount of misclassifications (errors) due to the visual similarity between certain classes. For example, the class *water* looks similar to *shadows*, and buildings with gray roofs will look similar to roads in the RGB channels. One solution to this problem, that has been addressed in previous works, is to use additional sources of information as input to the classifier, such as Synthetic-aperture radar (SAR), Light detection and ranging, (LIDAR), or Digital Elevation Model (DSM) for the height information, and/or hyperspectral bands, near-infrared (NIR) bands, and synthetic spectral bands for texture and color information [Ma *et al.*, 2017; Cheng *et al.*, 2017]. However, these works are impractical for satellite images that only contain RGB channels, such as Google Maps. Another possible solution to increase the performance is to change the architecture of the classifier in order to increase the capacity, e.g., by using deep Convolutional Neural Networks (DCNNs) [Ball *et al.*, 2017; Zhang *et al.*, 2017; Guirado *et al.*, 2017].

In this paper, instead of adding additional sources of information or experimenting with the architecture of the classifier, we aim to spatially explain the errors in terms of their structure and neighborhood. To this end, we propose a representation of the context that includes symbolic concepts and their relations, in order to reason upon and retrieve the required characteristics of the data.

Integration of data-driven learning methods with symbolic reasoning has been identified in the literature as one of the key challenges in Artificial Intelligence [Garcez *et al.*, 2015]. Depending on the approaches to represent both low and high level data, such integration has been addressed under different names that include abduction-induction in learning [Raymond, 2000], structural alignment [Alirezaie and Loutfi, 2012], and neural-symbolic methods [Besold *et al.*, 2017; Bader and Hitzler, 2005]. With the increasing interest in connectionist learning systems, and in particular in deep learning methods, research on integrated neural-symbolic systems has recently made considerable progress. Such integrations are routinely referred to as explainable Artificial Intelligence (XAI), and used to provide better insights into the learning process [Doran *et al.*, 2017].

1.1 Related Work

As discussed in [Xie *et al.*, 2017], in neural-symbolic systems where the learning is based on a connectionist learning system, one way of interpreting the learning process is to explain the classification outputs using the concepts related to the classifier's decision. The work presented in [Hendricks *et al.*, 2016] introduces a learning system based on a convolutional network LRCN [Donahue *et al.*, 2017] that provides

[*] Equal contribution

explanations over the decisions of the classifier. An explanation is in the form of a justification text. In order to generate the text, the authors have proposed a loss function upon sampled concepts that, by enforcing global sentence constraints, helps the system to construct sentences based on discriminating features of the objects found in the scene. However, in this work, no specific symbolic representation was provided, and the features related to the objects are taken from the sentences already available for each image in the dataset (CUB dataset [Wah *et al.*, 2011]).

With focus on the knowledge model, the work presented in [Sarker *et al.*, 2017] proposes a system that explains the classifier's outputs based on the background knowledge. The key tool of the system, called DL-Learner, works in parallel with the classifier and accepts the same data as input. Using the Suggested Upper Merged Ontology (SUMO)[1] as the symbolic knowledge model, the DL-Learner is also able to categorize the images by reasoning upon the objects together with the concepts defined in the ontology. The compatibility between the output of the DL-Learner and the classifier can be seen as a reliability support and at the same time as an interpretation of the classification process.

Likewise, the work detailed in [Icarte *et al.*, 2017] relies on a general-purpose knowledge model, namely, the ConceptNet Ontology. In this work, the integration of the symbolic model and a sentence-based image retrieval process based on deep learning is used to improve the performance of the learning process. For this, the knowledge about different concepts (e.g., their affordances, their relations with other objects) is aligned with objects derived from the deep learning method.

Although in the aforementioned works, the role of the symbolic knowledge represented by ontologies in regard to improving or interpreting the learning process has been emphasized, they are limited in terms of the symbolic representation models. More specifically, the concepts and their relations in ontologies are simplified, limiting the richness of deliberation in an eventual reasoning process, especially for visual imagery data.

1.2 Contribution

In this work, we propose an ontology-based reasoning approach to assist a neural network classifier for a semantic segmentation task. This assistance can be used in particular to represent typical errors and provide possible explanations which can later be used in correcting misclassification. Our work differentiates from the previous neural-symbol systems in two regards. Firstly, our method is able to find the most likely misclassified data (which can be rephrased as errors realization). Secondly, our model focuses on the misclassifications and uses ontological knowledge (with concepts and their spatial relations) together with a geometrical processing to explain them.

The rest of the paper is structured as follows: In Section 2 we present the steps of our approach. Section 3 provides the technical details on the classification process. The symbolic module including the ontological knowledge model and the reasoning process is explained in Section 4. Our experimental evaluations are presented in Section 5, which is followed by a brief discussion on the future work in Section 6.

[1] http://www.adampease.org/OP/

2 Approach

Figure 1 illustrates our approach of using background (ontological) knowledge to explain the errors from a classifier trained on satellite image data. The process is composed of several steps including: (1) error realization, (2) error characterization using geometrical/spatial reasoning, (3) error generalization based on the frequency, and (4) error explanation by aligning its features with the ontological knowledge (ontological reasoning). The inferred explanation can possibly contribute in the process of error correction (shown as dashed-line) and update the classification results.

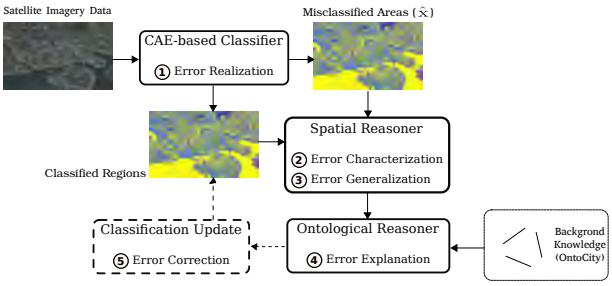


Figure 1: The process of explaining a misclassification in 4 steps: (1) error realization, (2) error characterization, (3) error generalization, and (4) error explanation. This process will contribute in error correction shown in dashed line.

Error realization refers to the process indicating likely misclassified areas (errors) on the map (to be detailed in Section 3). Given theses misclassified areas, a spatial/geometrical processing method characterizes such areas in terms of their structure and also identifies spatial relations within their vicinity. An ontological reasoning process is subsequently applied upon both the retrieved characterization of the errors and domain knowledge about generic spatial constraints in outdoor environments. After generalizing the relations retrieved by the reasoner based on their frequency, their semantics may justify the errors made by the classifier. Algorithm 1 provides further details of the explanation process.

Algorithm 1 Explaining Misclassification

Require: $S = empty, m, R$
S: A hash-map, empty in the beginning
m: The given misclassification matrix
R: The given list of classified regions
$\mathcal{G} \leftarrow extractGeometries(m)$
for each $r \in \mathcal{R}$ **do**
  $t \leftarrow getRegionType(r)$
  for each $g \in \mathcal{G}$ **do**
    $q \leftarrow calculateRCC(g, r)$
    $S(q, t) \leftarrow < q, t >$
  end for
end for
$< Q, T > \leftarrow getMostSeenPair(S)$
$C \leftarrow queryOntology(Q, T)$
$Explanation \leftarrow getRegionType(C)$

The data used in this work consists of a RGB satellite image of central Stockholm, Sweden, with size 4000×8000

pixels and a pixel-resolution of 0.5 meters. The data was divided into patches of 500×500 pixels and divided into train and test sets by a $50 - 50$ split so that both sets contained a similar class distribution. The ground truth used in the classification process has been provided by the Swedish Mapping, Cadastral and Land Registration Authority (Lantmäteriet).

3 Object Detection and Classification

A Convolutional Auto-encoder (CAE) [Masci *et al.*, 2011] is used to classify every pixel in each sub-image of size 500×500 pixels into one of 5 categories, namely, vegetation, road, building, water, and railroad. One layer of a CAE consists of an encoder and a decoder, see Figure 2.

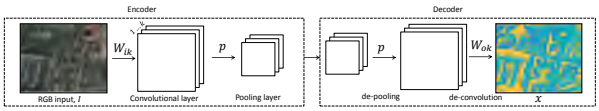


Figure 2: Overview of a one layer of Convolutional Autoencoder (CAE) that consists of an encoder and a decoder. The input is a RGB image and the output is the semantic segmentation.

The k-th feature map in the convolutional layer is calculated as:

$$h^k = \sigma_1 \left(I^i * W_{ik}^k + b^k \right) \tag{1}$$

where I^i is the input image with color channel i, W_{ik}^k is the k-th filter from input channel i and filter k, b^k is the bias for the k-th filter that is applied to the whole map, σ_1 is a non-linear activation function, and $*$ denotes the convolution operation. In this work, we used the Rectified Linear Unit (ReLu) [Nair and Hinton, 2010] as the activation function. For an input image of size $m \times m \times c$ and a filter matrix of size $n \times n \times c \times k$, the convolutional layer is of size $(m-n+1) \times (m-n+1) \times k$.

The pooling layer is obtained by downsampling the convolutional layer by taking the maximum value in each $p \times p$ non-overlapping subregion. The size of the pooling layer is $(m - n + 1)/p \times (m - n + 1)/p \times k$.

The unpooling is performed with switch variables [Zeiler *et al.*, 2011] that remember the position of the maximum value during the pooling operation.

Finally, a deconvolutional operation is performed to obtain the final output, x. For a typical convolutional autoencoder, the output has the same dimensions as the input image, I. However, for our application we want to perform a classification of the input image. Therefore, the output image x has the dimensions $m \times m \times K$ where K is the number of classes. The K-th output layer denotes the probability of each pixel belonging to class K. The output layer is calculated as:

$$x = \sigma_2 \left(o^k * W_{ok}^k + c^K \right) \tag{2}$$

where o^k is the k-th map of the unpooling layer, W_{ok}^k is the k-th filter from unpooling layer o and filter k, c^K is the bias for the K-th output layer, and σ_2 is the softmax non-linear activation function.

Figure 3 shows an overview of the method that is used to identify regions that have a high probability to be misclassified (i.e., the method to realize errors). The system consists of two Convolutional Auto-encoders (CAE) (noted *CAE 1* and *CAE 2* in Figure 3).

The first model, CAE 1, is trained to perform the image-to-image translation from the RGB input to the classified image x. CAE 1 is trained with supervised learning using the ground-truth y, see Section 3.

The second model, CAE 2, is trained unsupervised to reconstruct the input ground-truth y into the reconstruction of the ground-truth $\hat{y}$. The purpose of the model CAE 2 is to learn the overall structure and relation between classes.

The predicted label image x is then used as input to CAE 2 to get a reconstruction of the label image $\hat{x}$. The main idea is that regions that have a high reconstruction error, $(x - \hat{x})^2$, have a higher probability to be misclassified and should be further analyzed by the reasoner in order to explain a possible cause for the misclassification and give a suggestion for a more likely classification.

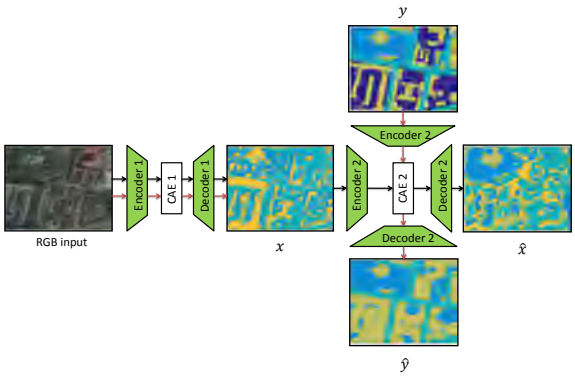


Figure 3: Overview of the method for indicating suspected misclassified regions. The input to the classifier (CAE 1) is an RGB image and produces the semantic segmentation, x. The label reconstructor (CAE 2) is first trained to reconstruct the groundtruth, y, into the reconstruction $\hat{y}$. The classified output x is then reconstructed using the label reconstructor to get the reconstructed classifications $\hat{x}$. The reconstruction error between x and $\hat{x}$ is then used to indicate the misclassified regions. Red arrows indicate the data processing during training and black arrows indicate the process during inference.

One important aspect of our method is the architecture for the label reconstructor in order to identify misclassified regions. On one hand, a single-layered CAE with a small filter size could easily reconstruct any configuration of the predicted map by simply reconstructing the local input pixel-by-pixel. Instead of increasing the filter size, we use a deep network with 5 layers. Due to the subsampling in each layer, this leads to the lower layers learn to reconstruct the local input and the higher layers learn the relation between areas with a larger perceptive field.

The classifier (CAE 1) and the label reconstructor (CAE 2) are constructed with the same architecture and consist of a 5-layer CAE. The filter size for each layer is $[11, 9, 7, 5, 3]$ and the number of filters in each layer is $[10, 20, 30, 40, 50]$. The pooling dimension is set to 2 in each layer and uses max-

pooling. The activation function in each layer is the ReLu-activation function except for last layer that uses a softmax activation function. The parameters were initialized with Xavier initialization [Glorot and Bengio, 2010] and trained using the AdaGrad [Duchi *et al.*, 2011] optimization method until convergence, which took around 50 hours on a GTX 1060 GPU.

4 Reasoning on misclassifications

The misclassification explanation process relies on geometrical and ontological reasoning. Before outlining the details of the explanation process, we first briefly introduce OntoCity which contains the background knowledge model used in this work.

4.1 OntoCity

OntoCity[2] is an extension of the GeoSPARQL[3] ontology that serves as a standard vocabulary for geospatial data by enabling qualitative spatial reasoning upon this type of data. OntoCity, whose representational details can also be found in [Alirezaie *et al.*, 2017], has been designed to represent cities in terms of different aspects including the structural details, conceptual and physical objects, their types (e.g., natural or man-made), and their relations (e.g., spatial constraints, affordances). The main concept in OntoCity is `oc:CityFeature`, which is the subclass of the class `geos:Feature`[4] and defines any spatial object with a geometry in the physical world. According to the following axiom[5], a city feature is (or more specifically is a subclass of) a feature whose geometry is in the form of a polygon and has at least one spatial relation with another city feature:

$$\texttt{oc:CityFeature} \sqsubseteq \texttt{geos:Feature} \sqcap \exists\, \texttt{geos:hasGeomtery.geos:Polygon} \sqcap \exists\, \texttt{oc:hasSpatialRelation.oc:CityFeature}$$

By spatial relation we refer to the 8 relations in RCC-8 (Region Connection Calculus) [Cohn *et al.*, 1997] that are also defined in GeoSPARQL, and are used to specialize the definition of features in a city. In OntoCity there are different types of features defined as subclasses of the `oc:CityFeature` class. For instance, a feature might be with a fixed geometry (`oc:FixedGeometryFeature`) or a dynamic one whose geometry changes in time (`oc:DynamicGeometryFeature`). Likewise, a feature can be physical (`oc:PhysicalFeature`, e.g., a landmark with absolute elevation value measured from the sea floor), conceptual (`oc:ConceptualFeature`, e.g., a rectangular division in a city regardless of their landmarks), mobile (`oc:MobileFeature`, e.g., a car), or stationary (`oc:StationaryFeature`, e.g., a building). The following axioms show some subsumption relations with `oc:CityFeature`:

$$\texttt{oc:DynamicGeometryFeature} \sqsubseteq \texttt{oc:CityFeature}$$
$$\texttt{oc:FixedGeometryFeature} \sqsubseteq \texttt{oc:CityFeature}$$
$$\texttt{oc:MobileFeature} \sqsubseteq \texttt{oc:CityFeature}$$
$$\texttt{oc:StationaryFeature} \sqsubseteq \texttt{oc:CityFeature}$$
$$\texttt{oc:ConceptualFeature} \sqsubseteq \texttt{oc:CityFeature}$$
$$\texttt{oc:PhysicalFeature} \sqsubseteq \texttt{oc:CityFeature} \sqcap \exists\, \texttt{oc:hasAbsoluteElevationValue.xsd:double}$$

Each of the aforementioned subclasses of the class `oc:CityFeature` has its own taxonomy. Due to the lack of space, we only mention a limited number of these axioms. For instance, `oc:Region` as a physical feature with a fixed geometry which is also stationary (i.e., non-mobile) represents a landmark that can per se be categorized into various types such as flat or non-flat regions, or likewise, into man-made or natural ones:

$$\texttt{oc:Region} \sqsubseteq \texttt{oc:PhysicalFeature} \sqcap \texttt{oc:StationaryFeature} \sqcap \texttt{oc:FixedGeometryFeature}$$
$$\texttt{oc:ManmadeRegion} \sqsubseteq \texttt{oc:Region}$$
$$\texttt{oc:NaturalRegion} \sqsubseteq \texttt{oc:Region}$$
$$\texttt{oc:FlatRegion} \sqsubseteq \texttt{oc:Region}$$
$$\texttt{oc:NonFlatRegion} \sqsubseteq \texttt{oc:Region} \sqcap \exists\, \texttt{oc:hasRelativeElevationValue.xsd:double} \sqcap \exists\, \texttt{oc:intersects.oc:Shadow}$$

A non-flat region in OntoCity refers to those landmarks with a relative elevation value, where by relative we mean the height measured from the ground level (in their neighborhood) and not from the absolute sea-level. Due to its height, a non-flat region is also assumed to cast shadows (defined as the class `oc:Shadow` in OntoCity) with which it has a spatial relation `oc:intersects` that subsumes several RCC-8 relations including partially overlapping (`geos:rcc8po`) and externally connected (`geos:rcc8ec`).

The subclasses of the class `oc:Region` can also specify the texture (i.e., type) of the landmark categorized as follows. in the following. It is worth mentioning that some of these region types are used as labels by the classifier to classify regions:

$$\texttt{oc:River} \sqsubseteq \texttt{oc:WaterArea} \sqsubseteq \texttt{oc:Region}$$
$$\texttt{oc:Road} \sqsubseteq \texttt{oc:PavedArea} \sqsubseteq \texttt{oc:ManmadeRegion}$$
$$\texttt{oc:Park} \sqsubseteq \texttt{oc:VegetationArea} \sqsubseteq \texttt{oc:Region}$$
$$\texttt{oc:Building} \sqsubseteq \texttt{oc:ManmadeRegion} \sqcap \texttt{oc:NonFlatRegion}$$

The RCC-8 relations are used in OntoCity to describe more specific features (e.g., bridges, shadows, shores) whose spatial relations with their vicinity are important in their definitions. For instance, a bridge is defined as a man-made region which is not flat (i.e, has elevation) and is partially overlapping (referring to the RCC-8 relation `geos:rcc8po`) at least another region (e.g., a water area, a street):

$$\texttt{oc:Bridge} \sqsubseteq \texttt{oc:ManmadeRegion} \sqcap \texttt{oc:NonFlatRegion} \sqcap \exists\, \texttt{geos:rcc8po.oc:Region}$$

The concept shadow as a spatial feature with a geometry is also defined in OntoCity. Although the shape of shadows depends on the exact position of the source light and also the height value of the casting objects, it is still possible to qualitatively describe shadows in the ontology. In OntoCity, a

[2] https://w3id.org/ontocity/ontocity.owl

[3] http://www.opengeospatial.org/standards/geosparql

[4] The prefixes `oc` and `geos` refer to the URIs of OntoCity and GeoSPARQL, respectively.

[5] The axioms are in description logic (DL) [Baader and Nutt, 2003].

shadow is seen as a conceptual (non-physical) feature whose geometry is dynamic and mobile (i.e., changing depending on the time of the day). The definition of the concept shadow becomes more precise by adding the spatial constrains saying that a shadow is also intersecting (`oc:intersects`) with at least one non-flat region (likely as its casting object):

$$\texttt{oc:Shadow} \sqsubseteq \texttt{oc:ConceptualFeature} \sqcap \texttt{oc:DynamicGeometryFeature} \sqcap \texttt{oc:MobileFeature} \sqcap \exists\, \texttt{oc:intersects.oc:NonFlatRegion}$$

The aforementioned axioms were a sub set of the general knowledge represented in OntoCity. However, the background knowledge can be much more specific and indicate unique features of a specific environment (e.g., "*in the given region there is no building connected to water areas*").

4.2 Explaining the misclassifications

The process of explaining the misclassifications is composed of several steps as shown in Algorithm 1. The algorithm accepts as input the list of the classified regions $\mathcal{R}$ as well as the misclassifications represented in the form of a pixel matrix $\mathtt{m}$ (as the reconstruction error between x and $\hat{x}$ explained in Section 3). In order to (spatially) characterize the errors, first the boundaries of the misclassified areas formed by misclassified pixels need to be extracted (see line 4 in the algorithm). Given the geometry of both the misclassified areas ($\mathcal{G}$) and the classified regions ($\mathcal{R}$) in the form of polygons, the algorithm calculates all the possible (RCC-8) qualitative spatial relations between any pairs of (g, r) where $g \in \mathcal{G}$ is a misclassified area and $r \in \mathcal{R}$ is a classified region in its vicinity. For each pair (g, r), besides the calculated spatial relation q, the algorithm also keeps the type of the region r shown as t. This information for each pair is added to the list S, which at the end of the geometrical calculation process will contain all the spatial relations that exist between the misclassified areas for each specific region type (see lines 5-11). The information provided in S can be also seen as the geometrical characteristics of the misclassified areas (i.e., error characterization).

As the next step, to find a general description indicating why the classifier has been confused, the characteristics of the errors are generalized based on their frequency. Let the pair $< Q, T >$ (see line 12) be the most observed spatial relation Q between the misclassified areas and a specific region type T, and let us view it as a representative feature of the misclassified areas. By applying an ontological reasoner upon OntoCity, we can query the ontology and ask for all the spatial features that are at least in one Q relation with type T, where the DL syntax of the query is: $\exists\, T.Q$. By applying the ontological reasoner the query can also be further generalized from the type T to its super-classes in OntoCity (see line 13). The concept (C) as a spatial feature ($C \sqsubseteq$ `oc:CityFeature`), which is inferred by the reasoner, is considered as an explanation.

5 Empirical Evaluation

The classifier was trained on the training set and applied upon the test data and resulted in $\approx 32K$ segments (or regions). Figure 4, left column, shows a 500×500 pixel large subset of the test data together with the segmentation. Each segment is classified into vegetation, road, building, water, or railroad (middle column). The reconstruction error (right column) identifies the probability that the segment is misclassified, in particular, the darker the segment the less likely it is to be misclassified (i.e., error realization).

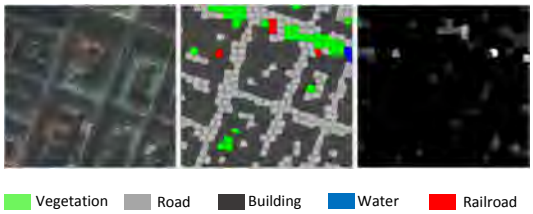


Figure 4: Left: Input RGB image together with the segmentation. Middle: Classified segmentation from the classifier. Right: Average reconstruction error for each segment where bright areas indicate suspected classification errors.

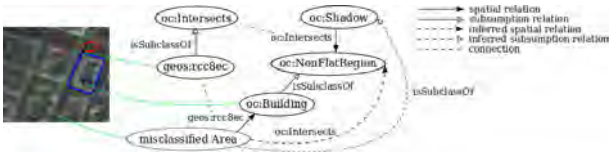


Figure 5: A high level representation of an example error explanation process. The *misclassified area* (in red) is *externally connected* (`geos:rcc8ec`) to the *building* region (in blue). By mapping the 3 aforementioned entities into their equivalent concepts in the ontology, the ontological reasoner infers the direct superclass (i.e., `oc:shadow`) of the misclassified area whose constraints are more general ($\exists$ `oc:intersects.oc:NonFlatRegion`) than the spatial representation of the red misclassified area.

Given the segments and the sorted list of reconstruction errors, the spatial reasoner together with the ontological reasoner are in charge of error explanation. The high level representation of the symbolic process is illustrated in Figure 5. In the following we go through the details of each step requited to achieve the final explanations for the errors. The error characterization process as the first step considers the top 100 misclassified regions to extract their boundaries and their spatial relations with their segmented neighborhood. This step has been implemented using the open-source JTS Topology Suite[6]. Table 1 shows a summary of the error characterization process. To find a representative feature of the misclassified areas (i.e., error generalization), Algorithm 1 takes into account the pair $< Q, T >$ as the most observed spatial relation Q between the misclassified areas and a specific region type T, which in our case, as shown in Table 1, is the pair $< Q =$ `geos:rcc8ec`, $T =$ `oc:Building`$>$ which involves 89 misclassified areas.

The pair $< Q, T >$ is enough to query the ontological concepts with spatial constraints. We have extended and used the reasoner Pellet, as an open-source Java based

[6] https://github.com/locationtech/jts

Type (t) \ Relation (q)	ec	po
oc:Building	**89**	5
oc:Road	41	0
oc:Water	19	1

Table 1: A summary on the error characterization process: Each cell value represents the number of misclassified regions involved in the given spatial relations (q) with the given region type (t), where ec and po refer to the RCC-8 relation *externally connected* and *partially overlapping*, respectively.

OWL 2 ontological reasoner [Sirin *et al.*, 2007]. The extension is in terms of filtering concepts based on their spatial constraints. The DL syntax of the query given to the reasoner is $\exists$ geos:rcc8ec.oc:Building interpreted as "*all the things that are at least in one* geos:rcc8ec *relation with the region type* oc:Building". The ontological reasoner results in a hierarchically linked concepts in the ontology from the most generalized to the most specialized (direct superclass) concepts satisfying the constraint given in the query. As shown in Table 2, the satisfactory concept is explained as "*a mobile conceptual feature with a dynamic geometry*" or more specifically a oc:shadow (as a direct answer of the query). In OntoCity, the concept shadow is defined based on the spatial constraint: $\exists$ oc:intersects.oc:NonFlatRegion, which is found by the reasoner as the direct generalization of the query $\exists$ geos:rcc8ec.oc:Building (where, geos:rcc8ec $\sqsubseteq$ oc:intersects and oc:Building $\sqsubseteq$ oc:NonFlatRegion) (see Figure 5).

Inferred concepts by the reasoner	description
oc:CityFeature	indirect superclass
oc:ConceptualFeature	indirect superclass
oc:DynamicGeometryFeature	indirect superclass
oc:MobileFeature	indirect superclass
oc:Shadow	**direct** superclass

Table 2: Error explanation as the output of the ontological reasoner.

Figure 6 shows two samples taken from the classification output, with some marked misclassified areas. At the first row, the areas marked with number 1 and 2 are misclassified as water. As the RGB image on the left illustrates, the marked areas are connected to buildings which cast shadows. Knowing that an area is under shadow, we can explain that the classifier is confused due to the similarity between the color of the shadow and the color of water (both looked dark). At the second row, the area marked with number 1 is likewise misclassified as water. This area is again (externally) connected with a building whose shadow can explain the misclassification. This area is furthermore located between (i.e., connected with) at least two disconnected regions labeled as roads which are disconnected at the shadow area. It can explain the second most observed relation listed in Table 2, between the misclassified areas and the region type oc:Road. Assuming buildings are often located close to roads (or streets), their shadow are likely casted on some parts of the roads. Therefore, a road instead of being recognized as a single road, is segmented into several roads disconnected at the shadow areas due to the change in their colors. Errors caused by shadows are not always labeled as water. Again in the second row, the areas marked with number 2 and 3 are also connected to buildings and roads, however, misclassified as railroads again due to the fact that the darkness of the shadow at this location is similar to the captured color of railroads in the image data.

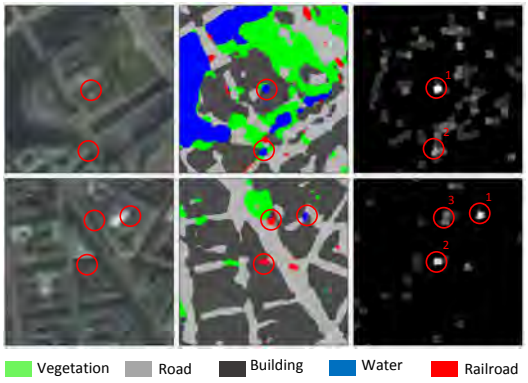


Figure 6: Two examples of the classification output along with their input RGB image, classified segmentation and the average reconstruction error. The misclassified areas marked with numbers are in spatial relations with buildings, roads, vegetation, etc. The ontological reasoner explains the misclassification as the result of the shadow of buildings on their neighborhood.

6 Discussion & Future Work

In this paper, we have proposed an ontology-based reasoning approach that automates the process of making sense of the misclassifications. The symbolic module (i.e., the spatial and ontological reasoning) used in this approach can act as a referee who explains why something has been misclassified. This explanation is made based on the geometrical features of the data which are not used by the classifier that only relies on the RGB channels of satellite image data.

Given the explanation about the errors, we ideally would like the symbolic module to also provide a correction of the misclassifications (see Figure 1). For this, there are a number of issues that have not been addressed in this work. The correction process depends on the inferred concept from the ontology. For example, if the concept as the explanation refers to a specific region type (i.e, a physical concept such as oc:Bridge) we could relabel the misclassified pixels with the region type. However, as we have shown, it can be a conceptual feature for which finding a relevant label to relabel the misclassified pixels might need further processing. If the reasoner infers that the misclassified area is under shadow, for example, the new label for this area is assumed to be the same as the type of the regions surrounding (referring to the RCC-8 relation *tangential proper part*: geos:rcc8tpp) the area under shadow. As the next step, we will focus on the correction process and deal with the aforementioned issues.

Acknowledgments

This work has been supported by the Swedish Knowledge Foundation under the research profile on Semantic Robots, contract number 20140033. The authors would also like to thank Mehul Bhatt at the Center for Applied Autonomous Sensor Systems for useful discussions that contributed to this paper.

References

[Alirezaie and Loutfi, 2012] M. Alirezaie and A Loutfi. Ontology alignment for classification of low level sensor data. In *KEOD*, pages 89–97. SciTePress, 2012.

[Alirezaie *et al.*, 2017] M. Alirezaie, A. Kiselev, M. Längkvist, F. Klügl, and A. Loutfi. An ontology-based reasoning framework for querying satellite images for disaster monitoring. *Sensors*, 17(11):2545, 2017.

[Baader and Nutt, 2003] F. Baader and W. Nutt. The description logic handbook. chapter Basic Description Logics, pages 43–95. Cambridge University Press, 2003.

[Bader and Hitzler, 2005] S. Bader and P. Hitzler. Dimensions of neural-symbolic integration - A structured survey. *CoRR*, abs/cs/0511042, 2005.

[Ball *et al.*, 2017] J.E. Ball, D.T. Anderson, and C.S. Chan. Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*, 11(4):042609, 2017.

[Besold *et al.*, 2017] T.R. Besold, A.S. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K. Kuehnberger, L.C. Lamb, D. Lowd, P.M.V. Lima, L. de Penning, G. Pinkas, H. Poon, and G. Zaverucha. Neural-Symbolic Learning and Reasoning: A Survey and Interpretation. *CoRR*, abs/1711.03902, 2017.

[Cheng *et al.*, 2017] G. Cheng, J. Han, and X. Lu. Remote sensing image scene classification: Benchmark and state of the art. *IEEE*, 2017.

[Cohn *et al.*, 1997] A.G. Cohn, B. Bennett, J. Gooday, and N.M. Gotts. Qualitative spatial representation and reasoning with the region connection calculus. In *Proc. Dimacs Int. WS on Graph Drawing, 1994.*, pages 89–4, 1997.

[Donahue *et al.*, 2017] J. Donahue, L.A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):677–691, 2017.

[Doran *et al.*, 2017] D. Doran, S. Schulz, and T.R. Besold. What Does Explainable AI Really Mean? A New Conceptualization of Perspectives. 2017.

[Duchi *et al.*, 2011] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(Jul):2121–2159, 2011.

[Garcez *et al.*, 2015] A.S. Garcez, T.R. Besold, L.D. Raedt, P. Földiak, P. Hitzler, T. Icard, K. Kühnberger, L.C. Lamb, R. Miikkulainen, and D.L. Silver. Neural-symbolic learning and reasoning: Contributions and challenges. In *AAAI 2015 Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches. T.R. SS-15-03*, 2015.

[Glorot and Bengio, 2010] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. 13th Int. Conf. on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[Guirado *et al.*, 2017] E. Guirado, S. Tabik, D. Alcaraz-Segura, J. Cabello, and F. Herrera. Deep-learning versus obia for scattered shrub detection with google earth imagery: Ziziphus lotus as case study. *Remote Sensing*, 9(12):1220, 2017.

[Hendricks *et al.*, 2016] L.A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell. Generating visual explanations. *Lect. Notes Comput. Sci.*, 9908 LNCS:3–19, 2016.

[Icarte *et al.*, 2017] R.T. Icarte, J.A. Baier, C. Ruz, and A. Soto. How a general-purpose commonsense ontology can improve performance of learning-based image retrieval. *IJCAI*, pages 1283–1289, 2017.

[Ma *et al.*, 2017] L. Ma, M. Li, X. Ma, L. Cheng, P. Du, and Y. Liu. A review of supervised object-based land-cover image classification. *ISPRS*, 130:277–293, 2017.

[Masci *et al.*, 2011] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59, 2011.

[Nair and Hinton, 2010] V. Nair and G.E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. 27th Int. Conf. on machine learning (ICML-10)*, pages 807–814, 2010.

[Raymond, 2000] J.M. Raymond. Integrating abduction and induction in machine learning. In *Abduction and Induction*, pages 181–191. Kluwer Academic Publishers, 2000.

[Sarker *et al.*, 2017] Md. K. Sarker, N. Xie, D. Doran, M. Raymer, and P. Hitzler. Explaining trained neural networks with semantic web technologies: First steps. *CoRR*, abs/1710.04324, 2017.

[Sirin *et al.*, 2007] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Web Semant.*, 5(2):51–53, June 2007.

[Wah *et al.*, 2011] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[Xie *et al.*, 2017] N. Xie, K. Sarker, D. Doran, P. Hitzler, and M. Raymer. Relating Input Concepts to Convolutional Neural Network Decisions. (2016), 2017.

[Zeiler *et al.*, 2011] M.D Zeiler, G.W Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Int. Conf. on Computer Vision (IEEE)*, pages 2018–2025, 2011.

[Zhang *et al.*, 2017] M. Zhang, X. Hu, L. Zhao, Y. Lv, M. Luo, and Sh. Pang. Learning dual multi-scale manifold ranking for semantic segmentation of high-resolution images. *Remote Sensing*, 9(5):500, 2017.

Sugeno Integral for Rule-Based Monotone Classification

Quentin Brabant[1], **Miguel Couceiro**[1], **Didier Dubois**[2], **Henri Prade**[2], **Agnès Rico**[3]
(1) Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
(2) IRIT, CNRS, Université Paul Sabatier 118 route de Narbonne 31062 Toulouse
(3) ERIC, Université Claude Bernard Lyon 1, 43 bld du 11-11, 69100 Villeurbanne
{quentin.brabant, miguel.couceiro}@loria.fr, {dubois, prade}@irit.fr, agnes.rico@univ-lyon1.fr

Abstract

We present a method for modeling empirical data by a rule set in ordinal classification problems. This method is nonparametric and uses an intermediary model based on Sugeno integral. The accuracy of rule sets thus obtained is competitive with other rule-based classifiers. Special attention is given to the length of rules, i.e., number of conditions.

1 Introduction

Let $\mathbf{X} = X_1 \times \cdots \times X_n$, where each X_i is a totally ordered set called *attribute* domain, and let L be a totally ordered set, whose elements are called *classes*. The minimal and maximal element of any totally ordered set X are denoted by 0_X and 1_X, respectively. An *instance* is a pair $(\mathbf{x}, y) \in \mathbf{X} \times L$. A *dataset* is a collection of instances (in which the same instance can appear several times). A *model* of a dataset is a function $f : \mathbf{X} \to L$, and its accuracy is the proportion of instances for which $f(\mathbf{x}) = y$ in agreement with the dataset.

We consider datasets that can be accurately modeled by a nondecreasing function. Such datasets are typically found in Multi-Criteria Decision Aid, where evaluation of alternatives depends on several criteria, but also in some medical diagnosis problems. The task of finding an accurate nondecreasing model of a dataset has been addressed in several ways (see, e.g., [Gutiérrez *et al.*, 2016]). In this short paper, we focus on rule-based models, since rules provide an explicit justification for each class prediction they make. We consider sets of (selection) rules of the form

$$\forall i \in A, x_i \geq \alpha_i \quad \Rightarrow \quad y \geq \delta \tag{1}$$

where $A \subseteq [n]$, $\delta \in L$, and, for all $i \in A$, $\alpha_i \in X_i$. The VC-DomLEM algorithm [Blaszczyński *et al.*, 2011] allows us to learn such a set of rules, which yields a good accuracy compared to other interpretable models.

In [Brabant *et al.*, 2018], we proposed an alternative method for learning rule sets, which relies on Sugeno integrals. This method does not require the tuning of any hyperparameter and is competitive with VC-DomLEM in terms of accuracy. Moreover, this method raises new questions about the relevance of capacities (i.e., monotonically increasing set functions) in data-modeling.

2 Rule-based and capacity based models

Let R be a set of rules of the form (1). For any rule $r \in R$, let

$$f_r(\mathbf{x}) = \delta \textbf{ if } \forall i \in A,\ x_i \geq \alpha_i, \textbf{ and } 0 \textbf{ otherwise.}$$

We define $f_R(\mathbf{x}) = \max_{r \in R} f_r$. Note that f_R is the smallest function such that the identity $f_R(x_1, \ldots, x_n) = y$ is compatible with all rules in R. We will say that a function f is *equivalent* to R if $f = f_R$.

In what follows, we use the notation $[n] = \{1, \ldots, n\}$. Let $\mu : 2^{[n]} \to L$ be a *capacity*, i.e., a set function $2^{[n]} \to L$ such that $\mu(\emptyset) = 0_L$, $\mu([n]) = 1_L$, and $\mu(I) \leq \mu(J)$ for all $I \subseteq J \subseteq [n]$. The Sugeno integral w.r.t. μ is the aggregation function $S_\mu : L^n \to L$ defined by

$$S_\mu(x_1, \ldots, x_n) = \max_{I \subseteq [n]} (\min(\mu(I), \min_{i \in I} x_i)).$$

Note that the Sugeno integral can be a model for ordinal classification only if $X_1 = \cdots = X_n = L$. A *Sugeno utility functional (SUF)* is a more expressive model which can merge values from different scales. Let $\varphi = (\varphi_1, \ldots, \varphi_n)$, where each $\varphi_i : X_i \to L$ is a nondecreasing function such that $\varphi_i(0_{X_i}) = 0_L$ and $\varphi_i(1_{X_i}) = 1_L$. The SUF $S_{\mu,\varphi}$ is the function defined by

$$S_{\mu,\varphi}(x_1, \ldots, x_n) = S_\mu(\varphi_1(x_1), \ldots, \varphi_n(x_n)).$$

It was shown in [Brabant *et al.*, 2018] that:

1. Any SUF is equivalent to a rule set.
2. Any single rule is equivalent to a SUF.
3. Some rule sets are not equivalent to a single SUF.

In other words, some combinations of rules cannot be expressed by one SUF. However, the second assertion allows to say that any rule set is equivalent to some function $M_{\boldsymbol{S}} : \mathbf{X} \to L$ defined by $M_{\boldsymbol{S}}(\mathbf{x}) = \max\{S_{\mu,\varphi}(\mathbf{x}) \mid S_{\mu,\varphi} \in \boldsymbol{S}\}$, where $\boldsymbol{S}$ is a set of SUFs. We call such function a max-SUF. There is no reason to think that a max-SUF provides a better interpretability than its equivalent rule set. However, we will show that it can be used as an intermediate model in the task of modeling a dataset by a rule set.

3 From SUFs to rule sets and vice-versa

Any SUF $S_{\varphi,\mu}$ is equivalent to the rule set

$$\bigcup_{I \subseteq [n]} \bigcup_{\delta \leq \mu(I)} \{\forall i \in I,\ x_i \geq \alpha_i \Rightarrow y \geq \delta\}, \tag{2}$$

where $\alpha_i = \min\{a \in X_i \mid \varphi_i(a) \geq \delta\}$. Note that this set is likely to contain redundant rules. Now let us show a method of translation of a rule set R into a SUF.

1. Initialize μ and $\varphi = (\varphi_1, \ldots, \varphi_n)$ with minimal values.
2. For each rule $x_1 \geq \alpha_1, \ldots, x_n \geq \alpha_n \Rightarrow y \geq \delta$ in R:
   (a) let $A = \{i \in [n] \mid \alpha_i > 0\}$,
   (b) increase $\mu(A)$ up to δ,
   (c) for each $i \in A$, increase $\varphi_i(\alpha_i)$ up to δ

After these steps we always have $S_{\mu,\varphi} \geq f_R$. When $S_{\mu,\varphi} > f_R$, no SUF is equivalent to R.

In some cases, it is not problematic that $S_{\mu,\varphi} > f_R$. For example, if f_R is a model of a dataset $\mathcal{D}$, we may want to find an SUF that best fits $\mathcal{D}$. Obtaining $S_{\mu,\varphi} = f_R$ is not always possible since SUFs are not expressive enough. However, equality can be always achieved using a max-SUF [Brabant *et al.*, 2018]. The method presented in the next section relies on this fact.

4 Learning rules from empirical data

Let $\mathcal{D}$ be a dataset. The following three steps provide a method for modeling $\mathcal{D}$ by a max-SUF.

1. Selection of an order-preserving subset of data. Two instances $(\mathbf{x}, y)$ and $(\mathbf{x'}, y')$ can be *anti-monotonic* together, i.e, $\mathbf{x} \leq \mathbf{x'}$ and $y' \leq y$. We iteratively remove instances from $\mathcal{D}$, starting from those that are anti-monotonic with the highest number of other instances, until no anti-monotonic pair remains. We denote by $\mathcal{D}^-$ the dataset obtained in this way.

2. Modeling $\mathcal{D}^-$ by a rule set R. Initialize R to $\emptyset$. For each instance $((a_1, \ldots, a_n), y)$ in $\mathcal{D}^-$, search for a set $A \subseteq [n]$ with minimal cardinality, such that the rule

$$\forall i \in A,\ x_i \geq a_i \quad \Rightarrow \quad y \geq \delta, \tag{3}$$

is not contradicted by any instance in $\mathcal{D}^-$. Add the rule (3) to R. At the end of this step, the class of each instance in $\mathcal{D}^-$ is exactly predicted by f_R.

3. Translation of R into a max-SUF. See Algorithm 1. The obtained max-SUF is not necessarily equivalent to R, but it fits $\mathcal{D}^-$ precisely.

Algorithm 1: Makes a partition $\mathbf{P}$ of R such that the max-SUF $M_{\mathbf{S}}$ verifies $M_{\mathbf{S}}(\mathbf{x}) = y$ for each instance $(\mathbf{x}, y)$.

```
P ← {}
for each r ∈ R do
    affected ← false
    for each P ∈ P do
        translate P into a SUF S_{μ,φ}
        if S_{μ,φ}(x) ≤ y for all instance (x, y) in D⁻ then
            add r to P
            affected ← true
            break loop
    if affected = false then
        add {r} to P
```

	1	2	3	4	5	6	7	8	9	10	11	12	avg.
Steps 1,2.	72.7	88.7	96.2	77.9	86.0	46.2	77.4	25.6	68.8	63.4	57.2	51.3	67.6
Steps 1,2,3.	76	95.3	97.2	89.3	92.4	65.2	84.5	26.4	69.4	63	56.7	53.2	72.4
VC-DomLEM	76.7	96.3	97.1	91.7	95.4	67.5	87.7	26.9	66.7	55.6	56.4	54.6	72.7

Table 1: Accuracy obtained with each method on each dataset. Datasets are numbered as in [Blaszczyński *et al.*, 2011]

dataset \ Rule length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		1	64	23	11	1										
2	13	72	13	1	1	1	1	1								
3			3	24	41	32										
4	38	47	10	3	1	1										
5	3	25	31	13	5	2	3	2	1	1	3	5	3	1		1
6	2	20	37	22	6	2	1	1	1	9						
7		20	21	21	12	9	15	3								
8	6	62	16	16												
9	6	26	36	32												
10	5	26	42	27												
11	6	2	25	45	9	5	4	1		3						
12	3	24	44	21	5	1				2						

Table 2: Percentage of rules with a given length, for each data set.

Note that the max-SUF given by this method can be translated back into a rule set, which constitutes an equivalent model and is easier to interpret.

5 Empirical study

We compared our method to VC-DomLEM on the 12 datasets in [Blaszczyński *et al.*, 2011]. In order to show the importance of Step 3 in our method, we separately evaluated the rule set given by steps 1 and 2 alone, and the max-SUF given by steps 1,2, and 3. We see that Step 3 increases the accuracy substantially on most datasets. However, it is not clear why translating a rule set into a max-SUF has this positive effect.

The *length* of a rule of the form (1) is the size of A. Shorter rules are easier to interpret and constitute more concise models. The max-SUF obtained from Step 3 can be translated into an equivalent rule set R. The percentage of rules of each length in R is displayed in Table 2. The dual of max-SUFs are the min-SUFs, i.e., sets of (rejection) rules of the form

$$\forall i \in A, x_i \leq \alpha_i \quad \Rightarrow \quad y \leq \delta.$$

When learning min-SUFs by a dual method, the rule-length distribution differs from that obtained by learning max-SUFs. Long rules of one type sometimes go along with short rules of the other type.

References

[Blaszczyński *et al.*, 2011] J. Blaszczyński, R. Slowiński, and M. Szelag. Sequential covering rule induction algorithm for variable consistency rough set approaches. *Information Sciences*, 181(5):987 – 1002, 2011.

[Brabant *et al.*, 2018] Q. Brabant, M. Couceiro, D. Dubois, H. Prade, and A. Rico. Extracting decision rules from qualitative data via Sugeno utility functionals. In *Proc. Int. Conf. on Inf. Proc. & Manag. of Uncertainty, Cadiz, Spain (IPMU'18)*, 2018.

[Gutiérrez *et al.*, 2016] P. A. Gutiérrez, M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernández-Navarro, and C. Hervás-Martínez. Ordinal regression methods: Survey and experimental study. *IEEE Trans. on Knowledge and Data Engineering*, 28(1):127–146, 2016.

Inducing Regular Grammars Using Recurrent Neural Networks

Mor Cohen[*], **Avi Caciularu**[*], **Idan Rejwan**[*], **Jonathan Berant**
School of Computer Science, Tel-Aviv University, Tel-Aviv, 6997801 Israel
morco@outlook.com, avi.c33@gmail.com,
idanrejwan@mail.tau.ac.il, joberant@cs.tau.ac.il

Abstract

Grammar induction is the task of learning a grammar from a set of examples. Recently, neural networks have been shown to be powerful learning machines that can identify patterns in streams of information. In this work we investigate their effectiveness in inducing a regular grammar from data, without any assumptions about the grammar. We train a recurrent neural network to distinguish between strings that are in or outside a regular language, and utilize an algorithm for extracting the learned finite-state automaton. We apply this method to several regular languages and find unexpected results regarding the connections between the network's states that may be regarded as evidence for generalization.

1 Introduction

Grammar induction is the task of learning a grammar from a set of examples, thus constructing a model that captures the patterns within the observed data. It plays an important role in scientific research of sequential phenomena, such as human language or genetics. We focus on the most basic level of grammars - regular grammars. That is, the set of all languages that can be decided by a Deterministic Finite Automaton (DFA).

Inducing regular grammars is an old and extensively studied problem (De la Higuera, 2010). However, most suggested methods involve prior assumptions about the grammar being learned. In this work, we aim to induce a grammar from examples that are in or outside a language, without any assumption on its structure.

Recently, neural networks were shown to be powerful learning models for identifying patterns in data, including language-related tasks (Linzen et al., 2016; Kuncoro and Ballesteros,). This work investigates how good neural networks are at inducing a regular grammar from data. More specifically, we investigate whether RNNs, a neural network that specializes in processing sequential streams, can learn a DFA from data.

RNNs are suitable for this task since they resemble DFAs. At each time step the network has a current state, and given the next input symbol it produces the next state. Formally, let s_t be the current state and x_{t+1} the next input symbol, then the RNN computes the next state $s_{t+1} = \delta(s_t, x_{t+1})$, where δ is the function learned by the RNN. Consequently, δ is actually a transition function between states, similar to a DFA.

This analogy between RNNs and DFAs suggests a way to understand RNNs. It enables us to "open the black box" and analyze the network by converting it into the corresponding DFA and examining the learned language.

Inspired by that, we explore a method for grammar induction. Given a labeled dataset of strings that are in and outside a language, we wish to train a network to classify them. If the network succeeds, it must have learned the latent patterns underlying the data. This allows us to extract the states used by the network and reconstruct the grammar it had learned.

There is one major difference between the states of DFAs and RNNs. While the former are discrete and finite, the latter are continuous. In theory, this difference makes RNNs much more powerful than DFAs (Siegelmann and Sontag, 1995). However in practice, simple RNNs [1] are not strong enough to deal with languages beyond the regular domain

[*] Authors equally contributed to this work.

[1] Without the aid of additional memory such as in LSTMs (Hochreiter and Schmidhuber, 1997)

(Gers and Schmidhuber, 2001).

Similar ideas have already been investigated in the 1990s (Cleeremans et al., 1989; Giles et al., 1990; Elman, 1991; Omlin and Giles, 1996; Morris et al., 1998; Tino et al., 1998)[2]. These works also presented techniques to extract a DFA from a trained RNN. However, most of them included a priori quantization of the RNNs continuous state space, yielding exponential number of states even for simple grammars. Instead, several works used clustering techniques for quantizing the state space, which yielded much smaller number of states (Zeng et al., 1993). However, they fixed the number of clusters in advance. In this work, we introduce a novel technique for extracting a DFA from an RNN using clustering without the need to know the number of cluster. For that purpose, we present a heuristic to find the most suitable number of states for the DFA, making the process much more general and unconstrained.

2 Problem Statement

Given a regular language $\mathcal{L}$ and a labeled dataset $\{(X_i, y_i)\}$ of strings X_i with binary labels $y_i = 1 \Leftrightarrow X_i \in \mathcal{L}$, the goal is to output a DFA A such that A accepts $\mathcal{L}$, i.e., $L(A) = \mathcal{L}$.

Since the target language $\mathcal{L}$ is unknown, we relax this goal to $A(\bar{X}_i) = \bar{y}_i$. Namely, A accepts or rejects correctly on a test set $\{(\bar{X}_i, \bar{y}_i)\}$ that has not been used for training. Accordingly, the accuracy of A is defined as the proportion of strings classified correctly by A.

3 Method

The method consists of the following main steps. [3]

1. **Data Generation** - creating a labeled dataset of positive and negative strings: $15,000$ strings for training the network, a validation set of $10,000$ strings for constructing the DFA, and a test set of $10,000$ strings for evaluating the results.
2. **Learning** - training an RNN within 15 epochs to classify the dataset with high accuracy- in almost all the conducted experiments (fully described in the next section), a nearly perfect validation accuracy is achieved, approximately 99%.
3. **DFA Construction** - extracting the states produced by the RNN, quantizing them and constructing a minimized DFA.

3.1 Data Generation

The following method was used to create a balanced dataset of positive and negative strings. Given a regular expression we randomly generate sequences out of it. As for the negative strings, two different methods were used. The first method was to generate random sequences of words from the vocabulary, such that their length distribution is identical to the positive ones. The other approach was to generate ungrammatical strings which are almost identical to the grammatical ones, making the learning more difficult for the RNN. We generated the negative strings by applying minor modifications such as word deletions, additions or movements. Both methods did not yield any significant difference in the results, thus we show only results for the first method.

3.2 Learning

We used the most basic architecture of RNNs, with one layer and cross entropy loss. In more detail, the RNN's transition function is a single fully-connected layer given by

$$s_t = \tanh\left(\mathbf{W}s_{t-1} + \mathbf{U}x_t + v\right).$$

Prediction is made by another fully-connected layer which gets the RNN's final state s_n as an input and returns a prediction $\hat{y} \in [0, 1]$ by,

$$\hat{y} = \sigma(\mathbf{A}\sigma(\mathbf{B}s_n + c) + d).$$

where σ stands for the sigmoid function, $\mathbf{W}, \mathbf{U}, \mathbf{A}, \mathbf{B}$ are learned matrices and v, c, d are learned vectors.
The loss used for training is cross entropy,

$$l = -\frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log\left(1 - \hat{y}_i\right),$$

where $y_1, \ldots, y_n$ are the true labels and $\hat{y}_1, \ldots, \hat{y}_n$ are the network's predictions. To optimize our loss, we employ the Adam algorithm (Kingma and Ba, 2014)

Our goal is to reach perfect accuracy on the validation set, in order to make sure the network succeeded in generalizing and inducing the regular grammar underlying the dataset. This assures that the DFA we extract later is reliable as much as possible.

[2] For a more thorough survey of relevant papers, see section 5.13 in (Schmidhuber, 2015)

[3] Python code is available at `https://github.com/acrola/RnnInduceRegularGrammar`

3.3 DFA Construction

When training is finished, we extract the DFA learned by the network. This process consists of the following four steps.

Collecting the states First, we collect the RNN's continuous state vectors by feeding the network with strings from the validation set and collecting the states it outputs while reading them.

Quantization After collecting the continuous states, we need to transform them into a discrete set of states. We can achieve this by simply using any conventional clustering method with the Euclidean distance measure. More specifically, we use the K-means clustering algorithm, where K is taken to be the minimal value such that the quantized graph's classifications match the network's ones with high rate. That is, for each K we build the quantized DFA (as we describe later) and count the number of matches between the DFA's classifications and the network's over a validation set. We return the minimal K that exceeds 99% matches. It should be noted that the initial state is left as is and is not associated with any of the clusters.

Building the DFA Given the RNN's transition function δ and the clustering method c, we use the following algorithm to build the DFA.

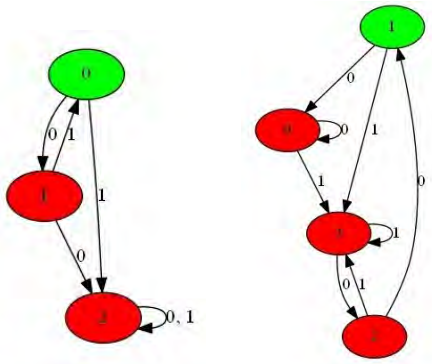
Algorithm 1 DFA Construction

$V, E \leftarrow \phi, \phi$
for each sequence X_i **do**
  $s_{t-1}, v_{t-1} \leftarrow s_0, c(s_0)$
  for each symbol $x_{t+1} \in X_i$ **do**
    $s_t \leftarrow \delta(s_{t-1}, x_t)$
    $v_t \leftarrow c(s_t)$
    Add v_t to V
    Add $(v_{t-1}, x_t) \rightarrow v_t$ to E
    $s_{t-1}, v_{t-1} \leftarrow s_t, v_t$
  end for
  Mark v_t as accept if $\hat{y}_i = 1$
end for
return V, E

Finally, we use the Myhill-Nerode algorithm in order to find the minimal equivalent DFA (Downey and Fellows, 2012).

4 Experiments

To demonstrate our method, we applied it on the following few regular expressions.



(a) $(01)*$ (b) $(0|1)^*100$

Figure 1: Minimized DFAs for binary regexes

4.1 Simple Binary Regexes

The resulting DFAs for the two binary regexes $(01)*$ and $(0|1)*100$ are shown in Figure 1.

It can be observed that the method produced perfect DFAs that accept exactly the given languages. The DFAs accuracy was indeed 100%.

A finding worth mentioning is the emergence of cycles within the continuous states transitions. That is, the RNN before quantization mapped new states into the exact same state it has already seen before. This is surprising because if we think of the continuous states as random vectors, the probability to see an exact vector twice is zero. This finding, which was reproduced for several regexes, may be an evidence for generalization as we discuss later.

4.2 Part of Speech Regex

To test our model on a more complicated grammar, we created a synthetic regex inspired by natural language. The regex we used describes a simplified part-of-speech grammar,

DET? ADJ$*$NOUN VERB (DET? ADJ$*$NOUN)?

The resulting DFA is shown in Figure 2.

The DFA's accuracy was 99.6%, i.e. the learned language is not exactly the target one, but is very close. For example, it accepts sentences like

The nice boy kissed a beautiful lovely girl

and rejects sentences like

The boy nice

However, by examining the DFA we can find sentences that the network misclassifies. For example, it accepts sentences like

The the boy stands

Inspecting the DFA's errors might be meaning-

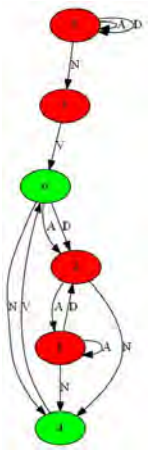


Figure 2: Minimized DFA for synthetic POS regex

ful also for training, as a technique for targeted data augmentation. By the pumping lemma, each of the states where the network is wrong stands for an infinite class of sequences that end at the same state. In other words, those states are actually a "formula" for generating as much data as we want, such that the network is wrong. This way, the network can be re-trained on its own errors.

5 Discussion

Learning

The network reached 100% accuracy quickly on the synthetic datasets. This may indicate that deciding a regular language is a reasonable task for an RNN, and illustrates the similarity between RNNs and DFAs discussed earlier.

Emergence of cycles

The emergence of cycles within the continuous states, mentioned in experiment 4.1 , might be understood as an evidence for generalization. Having learned those cycles means that for an infinite set of sequences the network will always traverse the same path and predict the same label. In other words, the model has generalized for sequences of arbitrary length *before* quantization.

It should be noted that such cycles have also been noticed in (Tino et al., 1998). Nevertheless, in their work the learning objective was predicting the next state rather than classifying the whole sentence. As a result, the emergence of cycles is expected, since the network was forced to learn them by the supervision. This differs from the case of classification, as learning the states and the cycles is not supervised.

Quantization

The process used for quantizing the states may introduce conflicts, if two different states in the same cluster lead to two different clusters for the same input. However, all of our experiments did not yield any conflict. This means that the clusters do reflect well the RNN's different states.

This is reasonable due to the continuity of the RNN's transition function, which maps "close" states into "close" states in terms of Euclidean distance. Thus, states within the same cluster will have similar transitions and therefore be mapped into the same cluster.

Another way to confirm the clusters validity is to check their compatibility with the network's decisions, i.e., whether accepting or rejecting states are clustered together. Figre 3 presents the continuous vectors for the binary regex $(0|1) * 100$[4]. Clearly, the states are divided into five distinct clusters and only one of them is accepting.

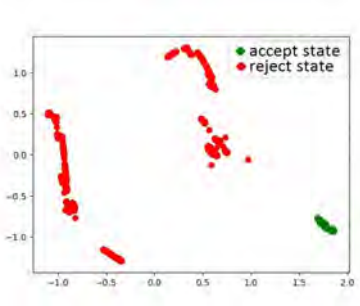


Figure 3: RNN's continuous states

6 Conclusions

We investigated a method for grammar induction using recurrent neural networks. This method gives some insights about RNNs as a learning model, and raises several questions, for example regarding the cycles within the continuous states.

Quantization via clustering proved itself to reflect well the true states of the network. Identifying an infinite set of vectors as one state may reduce the network's sensitivity to noise. Thus, states quantization during or after training may be considered as a technique for injecting some robustness to the model and reducing overfitting.

Finally, this method may serve as a tool for scientific research, by finding regular patterns within a real-world sequence. For example, it would be interesting to use this method for natural language, more specifically to induce grammar rules

[4]To reduce the vectors dimensions we used PCA.

of phonology, which is claimed to be regular (Kaplan and Kay, 1994). Another example is to use it to find regularity within the structure of DNA, which is also regarded as regular (Gusfield, 1997).

7 Acknowledgements

This work was supported by the Yandex Initiative in Machine Learning.

References

[Cleeremans et al.1989] Axel Cleeremans, David Servan-Schreiber, and James L McClelland. 1989. Finite state automata and simple recurrent networks. *Neural computation*, 1(3):372–381.

[De la Higuera2010] Colin De la Higuera. 2010. *Grammatical inference: learning automata and grammars*. Cambridge University Press.

[Downey and Fellows2012] Rodney G Downey and Michael Ralph Fellows. 2012. *Parameterized complexity*. Springer Science & Business Media.

[Elman1991] Jeffrey L Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225.

[Gers and Schmidhuber2001] Felix A Gers and E Schmidhuber. 2001. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

[Giles et al.1990] C Lee Giles, Guo-Zheng Sun, Hsing-Hen Chen, Yee-Chun Lee, and Dong Chen. 1990. Higher order recurrent networks and grammatical inference. In *Advances in neural information processing systems*, pages 380–387.

[Gusfield1997] Dan Gusfield. 1997. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge university press.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Kaplan and Kay1994] Ronald M Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378.

[Kingma and Ba2014] D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kuncoro and Ballesteros] Adhiguna Kuncoro and Miguel Ballesteros. Lingpeng kong, chris dyer, graham neubig, and noah a. smith. 2017. what do recurrent neural network grammars learn about syntax. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 1, pages 1249–1258.

[Linzen et al.2016] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368*.

[Morris et al.1998] William C Morris, Garrison W Cottrell, and Jeffrey Elman. 1998. A connectionist simulation of the empirical acquisition of grammatical relations. In *International Workshop on Hybrid Neural Systems*, pages 175–193. Springer.

[Omlin and Giles1996] Christian W Omlin and C Lee Giles. 1996. Extraction of rules from discrete-time recurrent neural networks. *Neural networks*, 9(1):41–52.

[Schmidhuber2015] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

[Siegelmann and Sontag1995] Hava T Siegelmann and Eduardo D Sontag. 1995. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150.

[Tino et al.1998] Peter Tino, Bill G Horne, and C Lee Giles. 1998. Finite state machines and recurrent neural networks–automata and dynamical systems approaches. Technical report.

[Zeng et al.1993] Zheng Zeng, Rodney M Goodman, and Padhraic Smyth. 1993. Learning finite state machines with self-clustering recurrent networks. *Neural Computation*, 5(6):976–990.

Using Sequence to Sequence Neural Networks for Solving Similar Mathematical Problems

Ali Davody, Mihai Sebastian Baba
Romanian Institute of Science and Technology
ali.davody@rist.ro, baba@rist.ro,

Abstract

Deep neural networks have enjoyed great success in recognizing patterns among large datasets. On the other hand, proofs of lots of mathematical theorems are very similar to each other. In this paper, by representing problems as directed graphs, we provide a concrete definition of similarity notion between problems. Then we examine the performance of deep sequential models to predicting solutions of similar mathematical problems.

1 Introduction

In recent years, *Deep Neural Network* (DNN) models have attracted much attention due to their great success in various tasks, including image recognition and classification [Krizhevsky *et al.*, 2012; Szegedy *et al.*, 2015; Ioffe and Szegedy, 2015; Simonyan and Zisserman, 2014; He *et al.*, 2015; Szegedy *et al.*, 2014], and speech recognition [Hinton *et al.*, 2012; Amodei *et al.*, 2016]. In all of these cases, one needs to recognize similar patterns in a collection of data samples, and this is done very well by DNNs.

On the other hand, when doing research in science or learning a topic, we often encounter a situation where we should solve a new problem which is similar to another problem with a known solution. For example, showing that $\sqrt{3}$ is irrational, is very similar to the irrationality proof of $\sqrt{2}$ and we may get some idea for our new problem by looking at the solution of $\sqrt{2}$ problem. Motivated by the success of DNNs in pattern recognition, it is natural to use them for solving similar problems in different branches of science.

In this work we take the first step in this direction by using DNNs for proving similar theorems in logic. We first give a precise meaning to the notion of similarity between problems. This is achieved by representing proof of theorems as directed graphs. Then by comparing the shape and nodes of graphs, corresponding to different problems, we would be able to define several levels of similarity.

Furthermore, the premises and proofs can be represented by ordered lists, which suggests sequence to sequence models as the first candidates to be applied for this task. We test the learning ability of sequence to sequence models on these similarity classes. We show that there is a hierarchy of model performance on these classes, and therefore our approach can be used to measure the intelligence level of learning models for proving logical theorems. Also we find some criteria for the statistics of data sets, in order to achieve a high accuracy. The main contributions of this work are as follows:

a) Introducing a concrete definition of pattern similarity in proofs and showing that it is useful for understanding the creativity of learning models.

b) Using sequence to sequence models for proving similar theorems in propositional logic, in an end-to-end fashion. To the best of our knowledge, this is the first work in this direction.

2 Related Work

Previous researches on applications of deep learning techniques in theorem proving are mainly focused on guiding automatic theorem provers, like E [Schulz, 2002], by premise selection. These activities have been initiated by [Alemi *et al.*, 2016], who used neural sequence models for premise selection. More recently, [Loos *et al.*, 2017] embedded deep networks inside prover E, as a proof guider. It has been shown there that guided automatic prover can prove more theorems under the same computational conditions. Also a new dataset (HolStep) for such kind of tasks has been introduced in [Kaliszyk *et al.*, 2017]. The state-of-the-art results on this dataset are presented in [Wang *et al.*, 2017a].

In addition, machine-learning methods have been used for fine-tuning the running parameters of automated theorem provers, known as strategy selection (see [Bridge *et al.*, 2014] and references therein).

There are also plenty of works on solving math-word problems including [Wang *et al.*, 2017b; Hosseini *et al.*, 2014; Amnueypornsakul and Bhat, 2014; Mitra and Baral, 2016] and reasoning in general (see for example [Rocktäschel *et al.*, 2015]).

Learning based on similarity is introduced and developed in [Indurkhya, 1991]. In particular, an algebraic approach has been invented to formalizing the underlying process of learning [Indurkhya, 1999].

Automatic theorem provers have been also used for checking the correctness of softwares and compilers. In this way, we can guarantee that the software is bug-free and implemented correctly. For example [Leroy, 2009] used the

Coq proof assistant for checking the correctness as well as programing the CompCert compiler (see also [Klein *et al.*, 2009]).

Here we try to adopt an end-to-end learning strategy for proving similar conjectures in mathematical logic. One advantage of this method is its applicability to solving exploratory problems, where the final answer of the problem is not given. As we shall see, such kind of extensions do not introduce conceptual (or even technical) difficulties in this approach.

It is worth mentioning that the output probability distribution of our model can be used for guiding a searching module in finding the final solutions. Since we do not use any search algorithm in this work, instead of comparing our model with classical automatic provers, we focus mainly on concept and usefulness of similarity notion in learning procedure. In particular, the introduced model in this work can be used as an actor network to provide optimal policy for a reinforcement learning based agent whose task is solving similar mathematical problems.

3 Inference in Mathematical Logic

3.1 Inference Graph

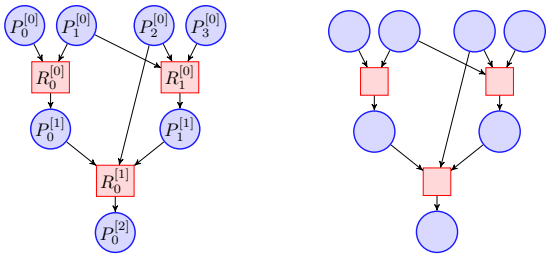
On very general ground, mathematical problems commonly fall into two categories: *Proving-like problems*, where the questioner should provide a proof of the desired statement given in the problem, and *exploring-like problems*, where the final answer is not provided and one should essentially solve a puzzle.

In both cases, the procedure of solving a mathematical problem consists of applying a sequence of *inference rules* to mathematical propositions, starting with a set of assumptions. In strict mathematical terminology, inference rules usually refer to the standard logical rules which one can use to infer any valid conclusion from some premises. Here by *inference rules* we mean any piece of thought action that one takes during the solution procedure. This includes using a mathematical definition, selecting a particular strategy of solution (like proof by contradiction), substitution in a formula, or applying a standard logical rule and so on.

Any rule can be considered a function which maps a set of inputs (this can be empty) to an output[1]. For logical rules, the inputs and output are propositional forms.

It turns out to be helpful to represent the solution of logical problems by a graph, which we call *inference graph*. The nodes of these graphs are mathematical propositions and inference rules. Figure 1a shows an example of inference graph. In this example, starting from top, the solution begins with four propositions $P_0^{[0]}, P_1^{[0]}, P_2^{[0]}$ and $P_3^{[0]}$ which are given as assumptions. In the next step we use inference rules $R_0^{[0]}$ and $R_1^{[0]}$ to conclude $P_0^{[1]}$ and $P_1^{[1]}$ respectively. Finally by applying inference rule $R_0^{[1]}$ to $P_0^{[1]}, P_2^{[0]}$ and $P_1^{[1]}$ we arrive at the result of problem, $P_0^{[2]}$. We label propositions, $P_i^{[l]}$, and

[1]In order to simplify the notation we assume that rules have just one output, in the case of several outputs we can define several copies of the rule.



(a) A typical inference graph. (b) The corresponding shape of inference graph on the left side.

Figure 1

rules, $R_i^{[l]}$, by their corresponding layer-number, l, counted from up to down, and position in the layer, i, counted from left to right in each layer.

More formally, we may represent the inference graph as an ordered list which is constructed by stacking proposition and rule layers from top to down. For example the inference graph of Figure 1a can be represented by the following list:

$$\Big[[P_0^{[0]}, P_1^{[0]}, P_2^{[0]}, P_3^{[0]}], [R_0^{[0]}(P_0^{[0]}, P_1^{[0]}), R_1^{[0]}(P_1^{[0]}, P_2^{[0]}, P_3^{[0]})], \\ , [P_0^{[1]}, P_1^{[1]}], [R_0^{[1]}(P_0^{[1]}, P_2^{[0]}, P_1^{[1]})], [P_0^{[2]}]\Big] \\ \equiv \Big[P^{[0]}, R^{[0]}, P^{[1]}, R^{[1]}, P^{[2]}\Big]$$

where $P^{[l]}$ and $R^{[l]}$ are lists of propositions and rules in the layer l. In a similar way, a general inference graph can be written as:

$$\Big[P^{[0]}, R^{[0]}, P^{[1]}, R^{[1]}, \cdots, P^{[L]}\Big]. \tag{1}$$

In order to compare different inference graphs, we need to introduce some definitions. Let us define the depth of the inference diagram to be the layer index of the last proposition, namely L. So, the depth of the inference graph of Figure 1a is two. Another useful notion is the shape of inference graph. We define the shape of inference graph to be a placeholder which is obtained by removing particular propositions and rules used in that solution. The shape of inference graph of Figure 1a is depicted in Figure 1b.

3.2 Similarity of Logic inferences

We are now in a position to define several levels of similarity between mathematical problems by comparing their corresponding inference graphs:

Zero-type similarity: Two problems are zero-type similar if their inference graphs have the same shape and rule layers. So the only difference between them is propositional layers. An example is given in Figures 2a and 2b.

Shape-type similarity: Two problems are similar of shape-type if their corresponding inference graphs have the same shape. Figures 2a and 2c show an example.

Depth-type similarity: Two problems are similar of depth type if their corresponding inference graphs have equal depth, like graphs which have been shown in Figures 2a and 2d.

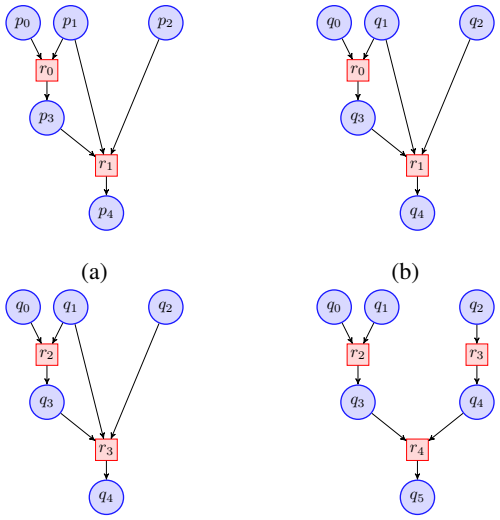


Figure 2: Inference-graphs (a) and (b) are similar of zero type. (a) and (c) are similar of shape type. Finally (d) is in depth similarity class of (a).

Intuitively, we expect that problems in a zero-type similarity class be more similar to each other than problems in shape-type or depth-type class of similarity. Indeed the level of similarity decreases from zero-type to depth-type class of problems. Before proceeding further, it is worth mentioning that the solution of a mathematical problem is not unique in general and problems can be solved in several ways. Accordingly, in the above definitions we compare inference graphs of specific solutions.

3.3 Inference rules of propositional logic

In this proof-of-concept study, for the sake of simplicity, we prefer to consider problems in natural deduction. Indeed natural deduction is a fundamental module of any reasoning procedure. Therefore the current work provides a backbone for future development in this direction. Actually we don't use any particular property of natural deduction system except the specific form of inference rules.

Table 10 lists down the inference rules of natural deduction with one and two number of inputs, respectively. We can consider these rules as building blocks of deep inference graphs in natural deductions.

4 Sequence to sequence neural network as theorem prover

As we have mentioned in the last section the solutions of mathematical problems can be written as sequences of mathematical propositions and inference rules. Also the problem itself may be represented by a list of assumptions followed by the conclusion of the problem (in the case of proving-like problems):

$$\left[P_0^{[0]}, P_1^{[0]}, \cdots, P_n^{[0]}, P_0^{[C]}\right]. \tag{2}$$

Also for exploring problems we may replace the result of the problem, $P_0^{[C]}$, in the above list with an encoded token which specifies the aim of problem.

Therefore a natural choice of deep-learning model to be used as a problem solver would be sequence to sequence neural network [Sutskever *et al.*, 2014; Cho *et al.*, 2014]. Figure 3 shows the network that we employ in this work. Encoder module is a bidirectional recurrent neural network with LSTM cells. We feed the ordered list 2, character by character, as one-hot vectors to the encoder. For the decoder we also use a RNN with LSTM cells. During the training procedure we feed the ordered list of the rules used for inferring $P_0^{[C]}$ as the target sequence of decoder. More precisely we use the label of rules followed by their index of inputs to specify this target list. For example the target list of inference graph depicted in Figure 2d is as follows:

$$\left[r_2, 0, 1, r_3, 2, r_4, 3, 4\right]. \tag{3}$$

In the case that the rule has some other parameters, we concatenate these parameters in the target list by putting them after the inputs of the corresponding rule. It is important to note that by having the input 2 and output 3 of theorem, we can completely reconstruct the inference graph.

In addition to simple sequence to sequnce model (Seq2Seq), we also consider sequence to sequence model equipped with the attention mechanism (Seq2SeqAttn), which has been shown has a great effect on the performance of seq2seq model [Luong *et al.*, 2015; Weston *et al.*, 2014]. We train these models using the TensorFlow framework [Abadi *et al.*, 2015] .

5 Zero-Similarity Class

Let us start by examining the performance of these models on solving problems within the zero-similarity class. The task on network in this case is recognizing building blocks of inference rules. We first consider the basic rules of natural deduction, namely problems with depth one. Actually the only meaningful similarity class that these problems belong to is the zero one.

5.1 Depth-One problems

Since the number of applied rules in this case is just one, we may employ a more simplified representation of decoder targets. Indeed the labels of input arguments are redundant and solution of a problem is completely specified by giving the label of applied rule, $[R_0^{[0]}]$.

The total number of listed rules in table 10 is $N_1 = 40$ and therefore we have forty depth-one distinct zero-similarity classes[2] . For each class we generate M_1 samples of problems. To do that, we randomly generate a formula (string) with maximum length of ten for each individual proposition. We use $p_{\text{train}} \times M_1$ of them to train the network, $p_{\text{dev}} \times M_1$ of them as development (dev) set and the rest $p_{\text{test}} \times M_1$ samples

[2]By distinct problems we mean problems with different shapes or rule lists. In other words two problems are distinct if they are not belong to the same zero-similarity class.

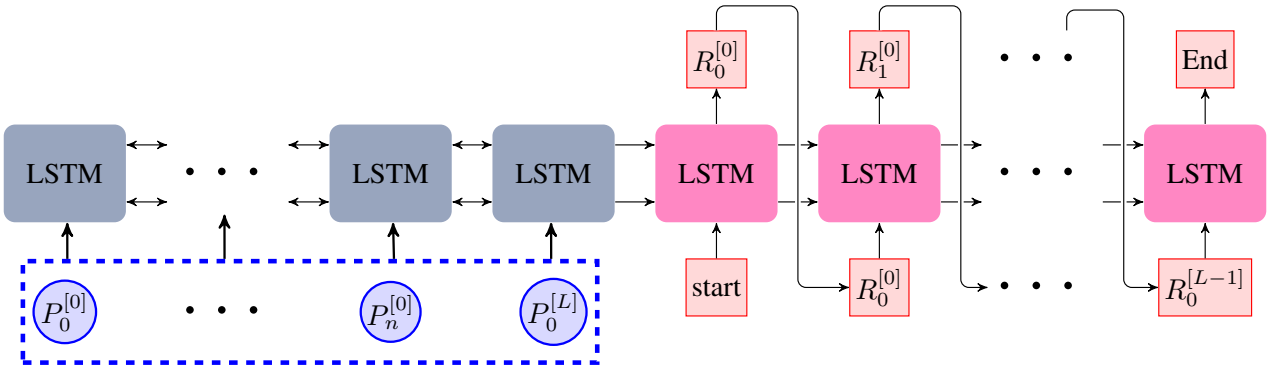


Figure 3: Architecture of seq2seq model we use in this work.

to measure the accuracy of the neural networks. In practice we use $p_{\text{train}} = 0.8$ and $p_{\text{dev}} = p_{\text{test}} = 0.1$. The table below shows the summary of data sets:

Table 1: Content of depth-one data sets.

data set	number of distinct problems	number of samples per class	total number of samples
total generated data	N_1	M_1	$M_1 \times N_1$
train	N_1	$p_{\text{train}} \times M_1$	$p_{\text{train}} \times M_1 \times N_1$
dev	N_1	$p_{\text{dev}} \times M_1$	$p_{\text{dev}} \times M_1 \times N_1$
test	N_1	$p_{\text{test}} \times M_1$	$p_{\text{test}} \times M_1 \times N_1$

Here are two examples of generated problems with zero-similarity:

Example 1:

$P_0^{[0]} = (((\mathtt{f} \vee (\mathtt{d} \vee \mathtt{d})) \to (\mathtt{d} \vee (\neg \mathtt{d}))) \wedge ((\mathtt{c} \vee \mathtt{f}) \to (\mathtt{b} \leftrightarrow \mathtt{f})))$

$P_1^{[0]} = ((\mathtt{f} \vee (\mathtt{d} \vee \mathtt{d})) \vee (\mathtt{c} \vee \mathtt{f}))$

$P_0^{[1]} = ((\mathtt{d} \vee (\neg \mathtt{d})) \vee (\mathtt{b} \leftrightarrow \mathtt{f}))$

rules list = decoder target = $[\mathtt{CD}]$

Example 2:

$P_0^{[0]} = (((\mathtt{a} \to (\neg \mathtt{c})) \to (\mathtt{b} \vee \mathtt{e})) \wedge ((\mathtt{d} \to \mathtt{d}) \to ((\neg \mathtt{c}) \wedge \mathtt{a})))$

$P_1^{[0]} = ((\mathtt{a} \to (\neg \mathtt{c})) \vee (\mathtt{d} \to \mathtt{d}))$

$P_0^{[1]} = ((\mathtt{b} \vee \mathtt{e}) \vee ((\neg \mathtt{c}) \wedge \mathtt{a}))$

rules list = decoder target = $[\mathtt{CD}]$

Table 2 displays the experimental results with different values of generated samples per class, M_1. As it is expected, the accuracy increases by feeding more examples to the network and almost saturates around $M_1 \simeq 60$, where the networks have learned to solve depth-one problems with $\%96$ accuracy by training them with $0.8 \times 60 = 48$ samples per problem, for a total of $48 \times 40 = 1920$ samples. In the next two subsections we consider deeper problems with zero-type similarity.

5.2 Cascade graphs

By definition a zero-similarity class consists of problems with the same shape and rule list. However, in general, we may have several (zero-) similarity classes in our data set. For the purpose of illustration, let us first generate our data with fixed-shape cascade graphs (see Figure 4a). These graphs are

Table 2: Accuracy of models on depth-one problems.

M_1	10	20	30	40	50	60
Seq2Seq	0.650	0.850	0.925	0.925	0.955	0.967
Seq2SeqAttn	0.675	0.850	0.908	0.944	0.925	0.966

well-defined for each depth and enable us to compare performance of network for different depths.

Like the last subsection, we are working with fixed shape problems and therefore we may specify the decoder targets just by the label of rules. For example the output of network for the graph corresponding to Figure 4b can be represented as $[R_1, R_2, R_3]$. Before we proceed further, we provide an example of generated problems with depth two:

Example 3:

$P_0^{[0]} = (((\mathtt{c} \wedge \mathtt{b}) \vee \mathtt{e}) \to ((\neg \mathtt{a}) \vee ((\neg \mathtt{f}) \to \mathtt{c})))$

$P_1^{[0]} = ((\mathtt{c} \wedge \mathtt{b}) \vee \mathtt{e})$

$P_2^{[0]} = ((\neg \mathtt{a}) \to (\neg \mathtt{b}))$

$P_3^{[0]} = (\neg(\neg \mathtt{b}))$

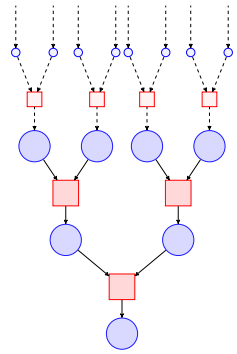
$P_0^{[1]} = ((\neg \mathtt{f}) \to \mathtt{c})$

rules list = $[\mathtt{MP}, \mathtt{MT}, \mathtt{DS}]$

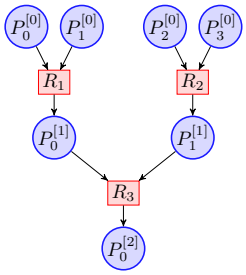
We consider N_{C_d} distinct cascade problems for each depth d. In particular, $N_{C_2} = 71$ and $N_{C_3} = 200$ cascade problems have been generated with depth two and three respectively. Dividing these problems to train, dev and test set is the same as the last section (see Table 3). We also add the $M_1 = 50$ data set of depth one problems (section 5.1) to the training set. This is reasonable, because these problems teach the network the basic rules.

data set	number of distinct problems with depth d	number of distinct shapes	number of samples per problem	number of depth-one samples
total	N_{C_d}	1	M_d	$M_1 \times N_1$
train	N_{C_d}	1	$p_{\text{train}} \times M_d$	$M_1 \times N_1$
dev	N_{C_d}	1	$p_{\text{dev}} \times M_d$	-
test	N_{C_d}	1	$p_{\text{test}} \times M_d$	-

Table 3: Summary of cascade data sets used for zero-similarity class.



(a) cascade shape graphs



(b) cascade graph with depth two

Figure 4

Table 4 depicts the performance of networks on test set in depth two and three. Interestingly, the accuracies in depths two and three are higher than their depth-one counterparts, with the same number of samples (Table 2). Indeed we arrive at a high performance around $\%99$ just by training networks with 8 samples per problem.

Table 4: Performance of model on depth two and three cascade problems with zero-similarity.

(a) depth 2

M_2	10	20	30
Seq2Seq	0.985	0.996	0.998
Seq2SeqAttn	0.984	0.981	0.996

(b) depth 3

M_3	10	20	30
Seq2Sqe	0.993	0.998	0.998
Seq2SqeAttn	0.98	0.992	0.993

5.3 General graphs

Let us now consider a more difficult task, where the data sets consist of problems with several shapes. For simplicity, we consider graphs where rule nodes have no common inputs, i.e. propositional nodes. At each depth, N_{S_d} number of such shapes have been generated. In particular we consider $N_{S_2} = 10$ and $N_{S_3} = 20$ different shapes in depth two and three. Figure 5 shows the depth-two shapes that we have generated. Let us denote the total number of distinct problems corresponding to N_{S_d} shapes by N_d. Using this notation, Table 5 represents the content of data sets.

Table 5: Summary of zero-similarity data sets with several shapes. In practice we use $N_2 = N_3 = 100$, $N_{S_2} = 10$, $N_{S_3} = 20$ and $p_{\text{dev}} = p_{\text{test}} = 0.1$.

data set	number of distinct problems with depth d	number of distinct shapes	number of samples per problem
total	N_d	N_{S_d}	M_d
train	N_d	N_{S_d}	$p_{\text{train}} \times M_d$
dev	N_d	N_{S_d}	$p_{\text{dev}} \times M_d$
test	N_d	N_{S_d}	$p_{\text{test}} \times M_d$

Here is an example of generated problems with the shape of Figure 5d:

Example 4:

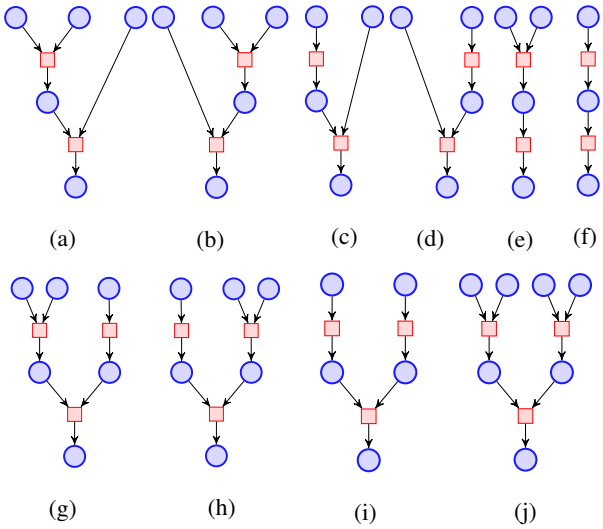


Figure 5: Generated shapes with depth two.

$$P_0^{[0]} = ((\mathtt{b} \wedge (\neg \mathtt{d})) \rightarrow (\neg \mathtt{b}))$$
$$P_1^{[0]} = (\neg(\neg(\mathtt{b} \wedge (\neg \mathtt{d}))))$$
$$P_0^{[2]} = (\neg \mathtt{b})$$
$$\text{rules list} = [\mathtt{InvDN}, 1, \mathtt{MP}, 0, 2]$$

Table 6 reports the experimental results. Again we see a high performance, even for this multi-shape task, by feeding a few samples to the networks. It is worth mentioning that the accuracy of random selection of rules is about $\frac{1}{\#\text{ tokens}} \simeq 0.015$. In conclusion, it seems that the sequential models are very robust for solving problems with zero-similarity.

Table 6: Performance of models on zero-similarity classes with several shapes.

(a) depth 2

M_2	10	20	30
Seq2Seq	0.932	0.956	0.969
Seq2SeqAttn	0.948	0.951	0.975

(b) depth 3

M_3	10	20	30
Seq2Seq	0.956	0.959	0.969
Seq2SeqAttn	0.945	0.976	0.976

6 Shape-Similarity Class

The task of networks in this section is to solve a problem by looking at the solutions of problems with the same shape but different rule lists. Like in the last section let us start with cascade problems.

6.1 Cascade graphs

Following our notation in section 5.2, we use N_{tr} out of N_{C_d} cascade problems to build our training set. The rest $N_{C_d} - N_{\text{tr}}$ problems have been used to generate dev and test sets. Contents of data sets are shown in the table 7. The performance of the models as a function of percentage of used problems, $\frac{N_{\text{tr}}}{N_{C_d}}$, is shown in Figure 6. As can be seen in this figure, the attention mechanism has a great impact on the performance

Table 7: Statistics of cascade-problems data sets with shape similarity. In practice we use $M_2 = M_3 = 30$, $N_{C_2} = 71$ and $N_{C_3} = 200$.

data set	number of distinct problems with depth d	number of distinct shapes	number of samples per problem
total	N_{C_d}	1	M_d
train	N_{tr}	1	M_d
dev	$\frac{1}{2}(N_{C_d} - N_{\text{tr}})$	1	M_d
test	$\frac{1}{2}(N_{C_d} - N_{\text{tr}})$	1	M_d

of network. In particular by using thirty percent of available data as training set, we are able to reach to accuracies more than sixty and eighty percent in depth two and three, respectively.

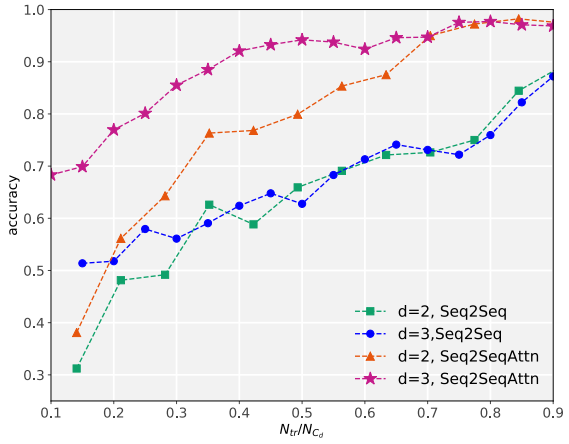


Figure 6: Performance of models on cascade problems with shape similarity.

6.2 General graphs

In a more general setup we may consider problems with several distinct shapes. If we define $P_d(sh)$ to be the number of generated distinct problems for each specific shape, we select N_{tr} out of $P_d(sh)$ problems to generate our training set. The remaining $P_d(sh) - N_{\text{tr}}$ problems will be used for producing dev and test sets, as shown in the table 8. The experimental results are shown in Figure 7. Again we see that the Seq2SeqAttn model has a better performance. However, due to presence of multiple classes with different shapes, the accuracy of both models decreases with respect to cascade case (with the same value of $N_{\text{tr}}/N_{\text{total}}$).

7 Depth Similarity Class

Finally we arrive at the most difficult task: the network should predict the shape and rule list of a given problem, by looking at the solution of training set problems with other shapes.

Among all generated shapes in each depth, N_{S_d}, we use N_{tr} of them to generating our training set and the rest are used for dev and test sets, as shown in the table 9. Figure 8 depicts

Table 8: Summary of data sets with shape similar problems. In practice we use $P_3(sh) = P_2(sh) = 20$, $N_{S_3} = N_{S_2} = 10$ and $M_3 = M_2 = 20$.

data set	number of distinct problems with depth d	number of distinct shapes	number of samples per problem
total	N_d	N_{S_d}	M_d
train	$N_{S_d} \times N_{\text{tr}}$	N_{S_d}	M_d
dev	$\frac{1}{2} \times N_{S_d} \times (P_d(sh) - N_{\text{tr}})$	N_{S_d}	M_d
test	$\frac{1}{2} \times N_{S_d} \times (P_d(sh) - N_{\text{tr}})$	N_{S_d}	M_d

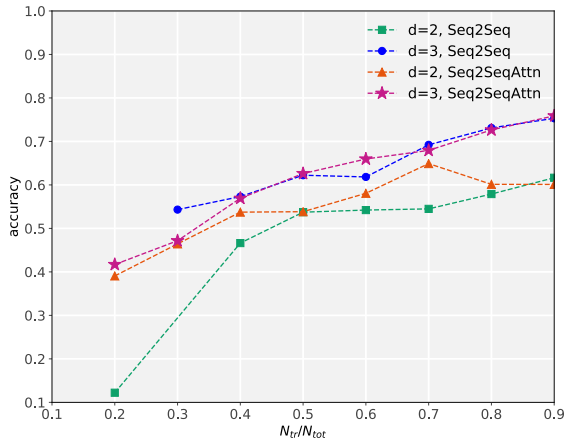


Figure 7: Performance of models on problems with shape similarity.

the experimental results. We observe that the networks perform poorly on the depth-similar problems in compared with the shape class (Figure 7), as we have expected. Indeed the accuracy of models is reduced by a factor of two. Also the attention mechanism has a little impact on the performance of the network.

8 Discussion

In this work we have explored the learning ability of sequential models in proving similar mathematical problems. To this end, we represented solutions as ordered graphs, which led to a natural definition of similarity notion between problems. We found that these models are very robust for solving problems with zero-type similarity. Also their performance on shape similar problems is fair enough. On the other hand solving problems with new shapes is the most difficult task for these models and we observed a lower performance compared to zero and shape similarity classes.

This end-to-end approach based on graph representation is very general and can be applied practically to any branches of mathematics. Its main advantages are:

- **Problem Engineering:** Data augmentation is standard technique for generating new data and making the model more robust against overfitting. In the same way, we can generate new problems (graphs) from existing ones in

Table 9: Statistics of depth similar data sets. In practice we use $N_{S_2} = 10$, $N_{S_3} = 100$, $P_2(sh) = 10$, $P_3(sh) = 2$, $M_3 = M_2 = 20$.

data set	number of distinct problems with depth d	number of distinct shapes	number of samples per problem
total	N_d	N_{S_d}	M_d
train	$N_{\text{tr}} \times P_d(sh)$	N_{tr}	M_d
dev	$\frac{1}{2} \times (N_d - N_{\text{tr}} \times P_d(sh))$	$\frac{1}{2}(N_{S_d} - N_{\text{tr}})$	M_d
test	$\frac{1}{2} \times (N_d - N_{\text{tr}} \times P_d(sh))$	$\frac{1}{2}(N_{S_d} - N_{\text{tr}})$	M_d

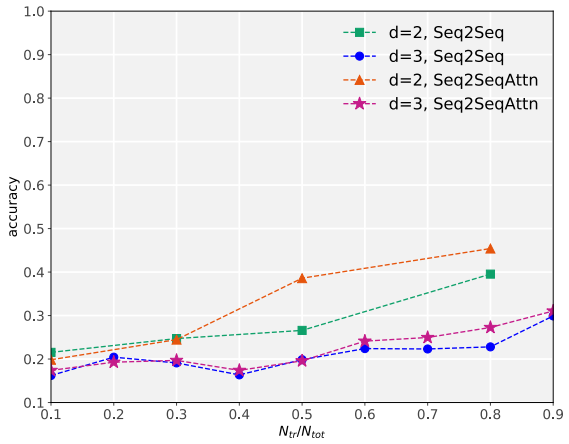


Figure 8: Performance of models on depth similarity classes.

several ways. For example we may join graphs, cutting a graph or changing the applied rules and so on.

- **Exploring problems:** This end-to-end method is also applicable to exploring problems. For example consider the following example:

  Find the roots of $x^2 + ax + b = 0$.

  This problem can be represented by $[\texttt{solve}, x^2 + ax + b = 0]$, where `solve` is a special token corresponding to *"Find the roots"*. The solution of these kind of problems can also be represented as graph. However data augmentation is more tricky in this case.

It would be interesting to improve the models that we have used in this work and achieving more higher accuracies. For example we may try other sophisticated architectures, using beam search [Wiseman and Rush, 2016] and so on. Another promising direction for future investigations is to use these models for performing a brute-force search or guiding a reinforcement learning agent and comparing the results with the ones of automatic theorem provers. We leave these extensions as well as studying exploring problems to the future works.

Acknowledgements

We would like to thank Bipin Indurkhya, Lehel Csat and Homa Davoudi for reading the paper thoroughly and giving useful comments. We would also like to thank Razvan Florian for encouragements and reading the paper. This work was supported by the European Regional Development Fund and the Romanian Government through the Competitiveness Operational Programme 20142020, project ID P_37_679, MySMIS code 103319, contract no. 157/16.12.2016.

A Inference Rules

id	Rule	Inputs	output
1	Modus Ponens (MP)	$p \to q, p$	q
2	Modus Tolens (MT)	$p \to q, \neg q$	$\neg p$
3	Constructive Dilemma (CD)	$(p \to q) \wedge (r \to s), p \vee r$	$q \vee s$
4	Destructive Dilemma (DD)	$(p \to q) \wedge (r \to s), \neg q \vee \neg s$	$\neg p \vee \neg r$
5	Disjunctive Syllogism (DS)	$p \vee q, \neg p$	q
6	Hypothetical Syllogism (HS)	$p \to q, q \to r$	$p \to r$
7	Conjunction (Conj)	p, q	$p \wedge q$
8	Addition2 (Add2)	p, q	$p \vee q$
9	Simplification (Simp)	$p \wedge q$	p
10	Addition1 (Add1)	p	$p \vee q$
11	Associative Laws (AssocConj)	$p \wedge q \wedge r$	$p \wedge (q \wedge r)$
12	Inverse Associative Laws (InvAssocConj)	$p \wedge (q \wedge r)$	$p \wedge q \wedge r$
13	Associative Laws (AssocDisj)	$p \vee q \vee r$	$p \vee (q \vee r)$
14	InAssociative Laws (InAssocDisj)	$p \vee (q \vee r)$	$p \vee q \vee r$
15	Commutative Laws (ComConj)	$p \wedge q$	$q \wedge p$
16	Commutative Laws (ComDisj)	$p \vee q$	$q \vee p$
17	Distributive Laws (Distr)	$p \wedge (q \vee r)$	$p \wedge q \vee p \wedge r$
18	InvDistributive Laws (InvDistr)	$p \wedge q \vee p \wedge r$	$p \wedge (q \vee r)$
19	Distributive Laws (Distr)	$p \vee (q \wedge r)$	$(p \vee q) \wedge (p \vee r)$
20	InvDistributive Laws (InvDistr)	$(p \vee q) \wedge (p \vee r)$	$p \vee (q \wedge r)$
21	Contrapositive Law (Contra)	$p \to q$	$\neg q \to \neg p$
22	InvContrapositive Law (InvContra)	$\neg q \to \neg p$	$p \to q$
23	Double Negation (DN)	p	$\neg\neg p$
24	InvDouble Negation (InvDN)	$\neg\neg p$	p
25	De Morgans Laws (DeMConj)	$\neg(p \wedge q)$	$\neg p \vee \neg q$
26	InvDe Morgans Laws (InvDeMConj)	$\neg p \vee \neg q$	$\neg(p \wedge q)$
27	De Morgans Laws (DeMDisj)	$\neg(p \vee q)$	$\neg p \wedge \neg q$
28	InvDe Morgans Laws (InvDeMDisj)	$\neg p \wedge \neg q$	$\neg(p \vee q)$
29	Idempotency (IdemConj)	$p \wedge p$	p
30	InvIdempotency (InvIdemConj)	p	$p \wedge p$
31	Idempotency (IdemDisj)	$p \vee p$	p
32	InvIdempotency (InvIdemDisj)	p	$p \vee p$
33	Material Equivalence (EquivConj)	$p \leftrightarrow q$	$(p \to q) \wedge (q \to p)$
34	InvMaterial Equivalence (InvEquivConj)	$(p \to q) \wedge (q \to p)$	$p \leftrightarrow q$
35	Material Equivalence (EquivDisj)	$p \leftrightarrow q$	$(p \wedge q) \vee (\neg p \wedge \neg q)$
36	InvMaterial Equivalence (InvEquivDisj)	$(p \wedge q) \vee (\neg p \wedge \neg q)$	$p \leftrightarrow q$
37	Material Implication (Impl)	$p \to q$	$\neg p \vee q$
38	InvMaterial Implication (InvImpl)	$\neg p \vee q$	$p \to q$
39	Exportation (Exp)	$p \wedge q \to r$	$p \to (q \to r)$
40	InvExportation (InvExp)	$p \to (q \to r)$	$p \wedge q \to r$

Table 10: Inference rules.

References

[Abadi *et al.*, 2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Alemi *et al.*, 2016] Alexander A. Alemi, François Chollet, Geoffrey Irving, Christian Szegedy, and Josef Urban. Deepmath - deep sequence models for premise selection. *CoRR*, abs/1606.04442, 2016.

[Amnueypornsakul and Bhat, 2014] Bussaba Amnueypornsakul and Suma Bhat. Machine-guided solution to mathematical word problems. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*, 2014.

[Amodei *et al.*, 2016] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.

[Bridge *et al.*, 2014] James P Bridge, Sean B Holden, and Lawrence C Paulson. Machine learning for first-order theorem proving. *Journal of automated reasoning*, 53(2):141–172, 2014.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[Hinton *et al.*, 2012] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[Hosseini *et al.*, 2014] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, 2014.

[Indurkhya, 1991] Bipin Indurkhya. On the role of interpretive analogy in learning. *New Generation Computing*, 8(4):385–402, 1991.

[Indurkhya, 1999] Bipin Indurkhya. An algebraic approach to modeling creativity of metaphor. In *Computation for metaphors, analogy, and agents*, pages 292–306. Springer, 1999.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[Kaliszyk *et al.*, 2017] Cezary Kaliszyk, François Chollet, and Christian Szegedy. Holstep: A machine learning dataset for higher-order logic theorem proving. *CoRR*, abs/1703.00426, 2017.

[Klein *et al.*, 2009] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, et al. sel4: Formal verification of an os kernel. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 207–220. ACM, 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[Leroy, 2009] Xavier Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7):107–115, July 2009.

[Loos *et al.*, 2017] Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. *CoRR*, abs/1701.06972, 2017.

[Luong *et al.*, 2015] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[Mitra and Baral, 2016] Arindam Mitra and Chitta Baral. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2144–2153, 2016.

[Rocktäschel *et al.*, 2015] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664, 2015.

[Schulz, 2002] Stephan Schulz. E–a brainiac theorem prover. *Ai Communications*, 15(2, 3):111–126, 2002.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[Szegedy *et al.*, 2014] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[Szegedy *et al.*, 2015] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[Wang *et al.*, 2017a] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. In *Advances in Neural Information Processing Systems*, pages 2783–2793, 2017.

[Wang *et al.*, 2017b] Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, 2017.

[Weston *et al.*, 2014] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014.

[Wiseman and Rush, 2016] Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. *CoRR*, abs/1606.02960, 2016.

Refining Manually-Designed Symbol Grounding and High-Level Planning by Policy Gradients

Takuya Hiraoka[12], **Takashi Onishi**[12], **Yoshimasa Tsuruoka**[23],
[1] NEC Central Research Laboratories
[2] National Institute of Advanced Industrial Science and Technology
[3] The University of Tokyo
t-hiraoka@ce.jp.nec.com, t-onishi@bq.jp.nec.com, tsuruoka@logos.t.u-tokyo.ac.jp

Abstract

Hierarchical planners that produce interpretable and appropriate plans are desired, especially in its application to supporting human decision making. In the typical development of the hierarchical planners, higher-level planners and symbol grounding functions are manually created, and this manual creation requires much human effort. In this paper, we propose a framework that can automatically refine symbol grounding functions and a high-level planner to reduce human effort for designing these modules. In our framework, symbol grounding and high-level planning, which are based on manually-designed knowledge bases, are modeled with semi-Markov decision processes. A policy gradient method is then applied to refine the modules, in which two terms for updating the modules are considered. The first term, called a reinforcement term, contributes to updating the modules to improve the overall performance of a hierarchical planner to produce appropriate plans. The second term, called a penalty term, contributes to keeping refined modules consistent with the manually-designed original modules. Namely, it keeps the planner, which uses the refined modules, producing interpretable plans. We perform preliminary experiments to solve the Mountain car problem, and its results show that a manually-designed high-level planner and symbol grounding function were successfully refined by our framework.

1 Introduction

Hierarchical planners have been widely researched in artificial intelligence communities. One of the main reasons for that is that the hierarchical planers can divide complex planning problems, which flat planners cannot solve, into a series of more simple sub-problems, by using high-level knowledges about the planning problem (e.g., [Nilsson, 1984; Choi and Amir, 2009; Kaelbling and Lozano-Pérez, 2011]).

A hierarchical planner is composed of multiple planner layers that are typically divided into two types: high-level and low-level. A low-level planner performs micro-level planning, and it deals with raw information about an environment. In contrast, a high-level planner performs macro-level planning, and it deals with more abstract symbolic information. The raw and abstract symbolic information are mapped to each other by *symbol grounding* functions. Imagine that a hierarchical planner is used for controlling a humanoid robot to put a lemon on a board. Here, the high-level planner makes a plan such a "Pick a lemon up, and then put it on a board." The low-level planner makes a plan for controlling the robot's motors according to sensor inputs, to achieve sub-goals given by the high-level planner (e.g., "Pick a lemon up"). As the low-level planner cannot understand what "Pick a lemon up" means, the symbol ground function converts it into actual values, in the environment, which the low-level planner can understand.

Hierarchical planners are often used for supporting human decision making (e.g., in supply chain [Özdamar *et al.*, 1998] or clinical operations [Fdez-Olivares *et al.*, 2011]). In such cases, people make decisions on the basis of a plan, and thus it is necessary that 1) they understand the plan (especially one of a high-level planner) and 2) they can reach satisfying outcomes by following the plan (i.e., the hierarchical planner gives appropriate plans).

In many previous studies on hierarchical planners, symbol grounding functions and high-level planners were designed manually [Nilsson, 1984; Malcolm and Smithers, 1990; Cambon *et al.*, 2009; Choi and Amir, 2009; Dornhege *et al.*, 2009; Wolfe *et al.*, 2010; Kaelbling and Lozano-Pérez, 2011]. Although this makes it possible for people to understand the plans easily, much human effort is needed to carefully design a hierarchical planner that provides appropriate plans.

Konidaris et al. [2014; 2015; 2016] have proposed frameworks for automatically constructing symbol grounding functions and high-level planners, but they require a human to carefully analyze them to understand the plans. These constructed modules are often complicated and, in such cases, analysis becomes a burden.

In this paper, we propose a framework that automatically refines manually-designed symbol grounding functions and high-level planners, with a policy gradient method. Our framework differs from frameworks proposed in the aforementioned previous studies on the basis of the following points:

- Unlike the hierarchical planners based solely on manually-designed symbol grounding functions and

high-level planners [Nilsson, 1984; Malcolm and Smithers, 1990; Cambon *et al.*, 2009; Choi and Amir, 2009; Dornhege *et al.*, 2009; Wolfe *et al.*, 2010; Kaelbling and Lozano-Pérez, 2011], our framework refines these modules without human intervention. This automated refinement reduces the design workload for the modules.

- Unlike the frameworks that automatically construct symbol grounding functions and high-level planners [Konidaris *et al.*, 2014; 2015; Konidaris, 2016], our framework refines these while attempting to keep the resulting symbol grounding consistent with prior knowledge of the definition of the symbols as much as possible (see Section 4). Therefore, a person can understand the plan that high-level planners output without careful analysis of the refined modules.

In this paper, we first explain our hierarchical planner (including the high-level planner and symbol grounding functions), and how these are designed (Section 3). Then, we introduce the framework designed to refine them (Section 4). Finally, we experimentally demonstrate the effectiveness of our framework (Section 5).

2 Preliminaries

Our framework, introduced in Section 4, is based on semi-Markov decision processes (SMDPs) and policy gradient methods.

2.1 Semi-Markov Decision Processes

SMDPs are a framework for modeling a decision problem in an environment where a sojourn time in each state is a random variable, and it is defined as a tuple $\langle S, O, R, P, \gamma \rangle$. $S \subseteq \mathbb{R}^n$ is the n-dimensional continuous state space; $O(s)$ is a function that returns a finite set of options [Sutton *et al.*, 1999] available in the environment's state $s \in S$; $R(s', t|s, o)$ is the reward received when option $o \in O(s)$ is executed at s; arriving in state $s' \in S$ after t time steps; $P(s', t|s, o)$ is a probability of $s' \in S$, and t after executing o in s; and $\gamma \in [0, 1]$ is a discount factor.

Given SMDPs, our interest is to find an optimal policy over options $\pi^*(o|s)$:

$$\pi^* = \arg\max_{\pi} V_\pi(s_0), \quad (1)$$

$$V_\pi(s_0) = E_\pi\left[R(s_1, t_0|s_0, o_0) + \gamma^{t_0} R(s_2, t_1|s_1, o_1) + ...\right] \quad (2)$$

where (s_0, o_0, t_0, s_1) and (s_1, o_1, t_1, s_2) are transitions of a state, an option, time steps elapsed while executing the option, and the arriving state after executing the option.

2.2 Policy Gradient

To find π^*, we use a policy gradient method [Sutton *et al.*, 2000]. In a policy gradient method, a policy $\pi_\theta(o|s)$ parameterized by θ is introduced to approximate π^*, and the approximation is performed by updating θ with a gradient. Although there are many policy gradient implementations (e.g., [Kakade, 2002; Silver *et al.*, 2014; Schulman *et al.*, 2015]), we use REINFORCE [Williams, 1992]. In REINFORCE, θ is updated as follows:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(\tilde{s_0}, \tilde{o_0}) V_{\pi_\theta}(\tilde{s_0}), \quad (3)$$

$$V_{\pi_\theta}(\tilde{s_0}) = R(\tilde{s_1}, \tilde{t_0}|\tilde{s_0}, \tilde{o_0}) + \gamma^{\tilde{t_0}} R(\tilde{s_2}, \tilde{t_1}|\tilde{s_1}, \tilde{o_1}) + ..., \quad (4)$$

where α is a learning rate and $(\tilde{s_0}, \tilde{o_0}, \tilde{t_0}, \tilde{s_1})$ and $(\tilde{s_1}, \tilde{o_1}, \tilde{t_1}, \tilde{s_2})$ are transitions of state, the executing option, elapsed time steps, and arriving state, which are sampled on the basis of π_θ in a time horizon. Other variables and functions are the same as those introduced in Section 2.1. We decided to use REINFORCE for our work because it has successfully worked in recent work [Silver *et al.*, 2016; Das *et al.*, 2017].

3 Hierarchical Planner with Symbol Grounding Functions

In this section, we first describe the outline of a hierarchical planner (including the high-level planner) with symbol grounding functions, which are manually designed. We then provide concrete examples of them. The high-level planner and symbol grounding functions described here are refined by the framework, which is proposed in Section 4.

The hierarchical planner (Figure 1) is composed of two symbol grounding functions (one for abstraction and the other for concretization), a high-level planner, a low-level planner, and two knowledge bases (one each for the high-level and low-level planners). These modules work as follows:

Step 1 : The symbol grounding function for abstraction receives raw information, abstracts it to a symbolic information on the basis of its knowledge base, and then outputs the symbolic information.

Step 2 : The high-level planner receives the abstract symbolic information, makes a plan using its knowledge base, and then outputs abstract symbolic information as a sub-goal, which indicates the next abstract state to be achieved.

Step 3 : The symbol grounding function for concretization receives the abstract symbolic information, concretizes it to raw information about a sub-goal, which specifies an actual state to be achieved, then outputs the raw information on the sub-goal. This module performs the concretization on the basis of its the knowledge base.

Step 4 : The low-level planner receives the raw information about a sub-goal and then interacts with the environment to achieve the given sub-goal. In the interaction, the low-level planner outputs primitive actions in accordance with the raw information given by the environment. The interaction continues until the low-level planner achieves the given sub-goal, or until the total number of elapsed time steps reaches a given threshold.

Step 5 : If the raw information from the environment is not a goal or terminal state, return to the Step 1.

The knowledge bases for symbol grounding functions and the high-level planners are designed manually.

Knowledge base for high-level planners is described as a simple planning domain definition language (PDDL) [McDermott *et al.*, 1998]. In a PDDL, objects, predicates, goals, and operators are manually specified.

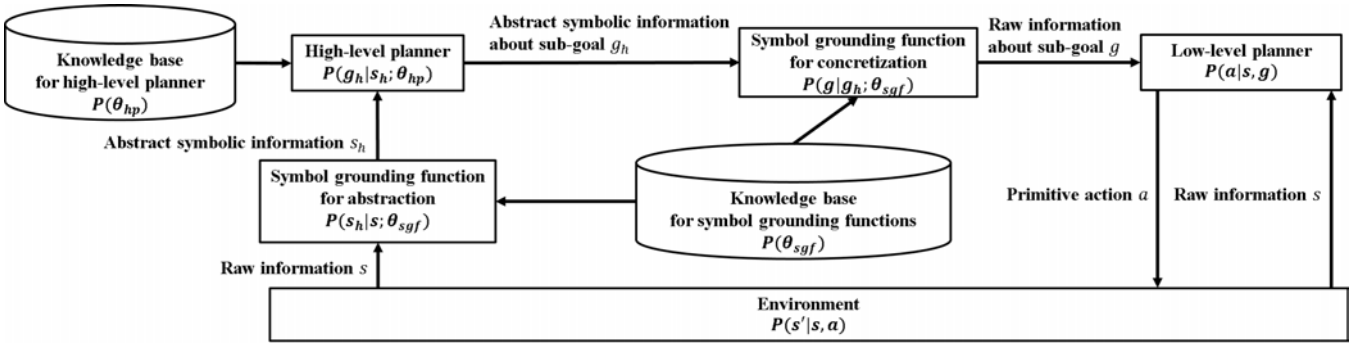


Figure 1: Outline of hierarchical planner with grounding functions.

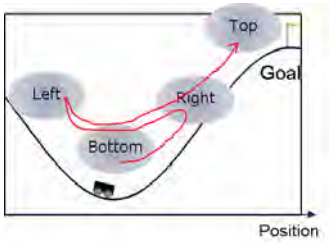


Figure 2: Mountain car with abstract symbols.

The objects and predicates are for building logical formulae, which specify the possible states in the planning domain. The operators are represented as a pair of preconditions and effects. The preconditions represent the states required for applying the operator, and the effects represent the arriving states after applying the operators. We use PDDLs in this work because they are widely used for describing a knowledge base for symbolic planners.

Knowledge base for symbol grounding functions is described as a list of maps between abstract symbolic information and corresponding raw information. In this paper, to simplify the problem, we assume that each item of abstract symbolic information is mapped into one interval of raw information. Despite its simplicity, it is useful for representing, for example, typical spatial information.

Here, we describe the knowledge bases and how the hierarchical planner works to solve the mountain car problem [Moore, 1991] (Figure 2). In this problem, a car is placed within a deep valley, and its goal is to drive out by going up the right side hill. However, as the car's engine is not strong enough, it needs to first drive back and forth between the two hills to generate momentum. In this problem, the hierarchical planner receives raw information (the position and velocity of the car) from the environment and is required to make a plan to move it to the goal (the top of the right side hill).

An example of knowledge for the high-level planner is shown in Table 1. In this example, objects are composed of only a "Car." Predicates are composed of four instances ("Bottom_of_hills(x), On_right_side_hill(x), On_left_side_hill(x), and At_top_of_right_side_hill(x)"). For example, "On_right_side_hill(Car)" means that the car is on the right side hill. Operators are composed of three types that refer to a transition of the objects on the hills. For example, "Opr.1" refers to the transition that object x has moved from the bottom of the hills to the right side hill.

Table 1: Example knowledge for high-level planners. Upper part describes examples of objects, predicates, and goals. Lower part describes examples of operators.

Objects	$x = \{\text{Car}\}$
Predicates	Bottom_of_hills(x), On_right_side_hill(x), On_left_side_hill(x), At_top_of_right_side_hill(x)
Goals	At_top_of_right_side_hill(Car)

Operators	Preconditions	Effects
Opr.1	Bottom_of_hills(x)	On_right_side_hill(x)
Opr.2	On_right_side_hill(x)	On_left_side_hill(x)
Opr.3	On_left_side_hill(x)	At_top_of_right_side_hill(x)

Table 2: Example knowledge for symbol grounding functions.

Abstract symbolic informations	Interval of raw information
Bottom_of_hills(Car)	position $\in [-0.6, -0.4]$
On_right_side_hill(Car)	position $\in [-0.2, 0.4]$
On_left_side_hill(Car)	position $\in [-1.2, -0.8]$
At_top_of_right_side_hill(Car)	position $\in [0.6, 0.8]$

An example of the knowledge for symbol grounding functions is shown in Table 2. This example shows mappings between abstract symbolic information (the location of the car), and corresponding intervals of raw information (the actual value of the car's position). For example, "Bottom_of_hills(Car)" is mapped to the position of the car is in the interval [-0.6, -0.4].

Given the knowledge described in Tables 1 and 2, an example of how the hierarchical planner works is shown as follows:

Example of Step 1: The symbol grounding function for abstraction receives raw information (position=-0.5 and velocity=0). The position is in the interval [-0.6, -0.4], which corresponds to the "Bottom_of_hills(Car)" in Table 2. Therefore, the symbol grounding function outputs "Bottom_of_hills(Car)."

Example of Step 2: The high-level planner receives "Bottom_of_hills(Car)," and makes a plan to achieve the goal ("At_top_of_right_side_hill(Car)"). By using the knowledge in Table 1, the high-level planner makes the plan [Bottom_of_hills(Car) $\rightarrow$

On_right_side_hill(Car) → On_left_side_hill(Car) → At_top_of_right_side_hill(Car)], which means "Starting at the bottom of the hills, visit, in order, the right side hill, the left side hill, and the top of the right side hill." After following the plan, the high-level planner outputs "On_right_side_hill(Car)."

Example of Step 3: The symbol grounding function receives "On_right_side_hill(Car)," and concretizes it to raw information about sub-goal (position= 0.1, velocity=*). Here, the position in the raw information is determined as the mean of the corresponding interval $[-0.2, 0.4]$ in Table 2. In addition, the mask (represented by "*") is putted to filter out factors in raw information, which is irrelevant in the sub-goal (i.e., velocity in this example).

Example of Step 4: The low-level planner receives position= 0.1 and the mask. To move the car to the given sub-goal (position=0.1), the low-level planner makes a plan to accelerate the car. This planning is performed by model predictive control [Camacho and Alba, 2013]. The low-level planner terminates itself when the car arrives at the given sub-goal (position=0.1), or when it takes a primitive action 20 times.

4 Framework for Refining Grounding Function and High Level Planner

In this section, we propose a framework for refining the symbol grounding functions and the high-level planner introduced in the previous section. In our framework, symbol grounding and high-level planning, which are based on manually-designed knowledge bases, are modeled with SMDPs. Refinement of the symbol grounding functions and the high-level planner is achieved by applying policy gradients to the model. First, we introduce an abstract model and then provide an example of its implementation in the mountain car problem. Finally, we explain how the policy gradient method is applied to the model.

4.1 Modeling Symbol Grounding and High-Level Planning with SMDPs

We model symbol grounding and high-level planning, which are based on manually-designed knowledge bases, with SMDPs. The symbol grounding functions and the high-level planner are modeled as components of the parameterized policy. In addition, the knowledge bases are modeled as priors for the policy's parameters.

We first assume that information and modules, which appear in hierarchical planning, are represented as random variables and probability functions, respectively (Figure 1). Suppose that S_h is a set of all possible symbols the symbol grounding functions and the high-level planner deal with, raw information is represented as an n-dimensional vector, and $A \subset \mathbb{R}^m$ is a set of all possible primitive actions. We denote raw information by $s, s' \in \mathbb{R}^n$ [1], abstract symbol information by $s_h \in S_h$, abstract symbolic information about a sub-goal by $g_h \in S_h$, raw information about a sub-goal $g \in \mathbb{R}^n$, and a primitive action by $a \in A$. In addition, we denote the symbol grounding function for abstraction by $P(s_h|s;\theta_{sgf})$, the symbol grounding function for concretization by $P(g|g_h;\theta_{sgf})$, the high-level planner by $P(g_h|s_h;\theta_{hp})$, the low-level planner by $P(a|s,g)$, the environment by $P(s'|s,a)$, the knowledge base for high-level planners by $P(\theta_{hp})$, and the knowledge base for the high-level planner by $P(\theta_{sgf})$. Here, θ_{sgf} and θ_{hp} are the parameters for the symbol grounding functions and the high-level planner, respectively.

[1] The denotation is the same as that of the state described in Section 2.1 because the raw information is modeled as the state.

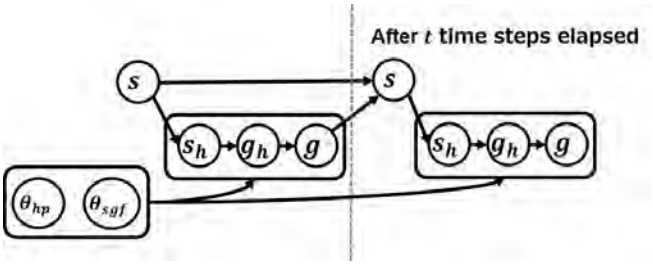


Figure 3: SMDPs for our framework.

High-level planning and symbol grounding based on the knowledge base are modeled as SMDPs (Figure 3). In this model, the components of SMDPs (i.e., an option, a state, a reward, and a transition probability) are implemented as follows:

Option o**:** is implemented as a tuple $\langle s_h, g_h, g \rangle$ of abstract symbolic information s_h, abstract symbolic information about a sub-goal g_h, and raw information about a sub-goal g.

State s**:** is implemented as raw information.

Reward $R(s', t|s, o)$**:** is the cumulative reward given by the environment $P(s'|s,a)$, while the low-level planner $P(a|s,g)$ is interacting with $P(s'|s,a)$.

Transition probability $P(s', t|s, o)$**:** is implemented as a function, which represents the state transition proceeded by the interaction between the low-level planner $P(a|s,g)$ and the environment $P(s'|s,a)$. Note that although the transition probability receives option $\langle s_h, g_h, g \rangle$, only g is used in the transition probability.

In this model, the parameterized policy π_θ is implemented to control abstraction of raw information, high-level planning, and concretization of abstract symbolic information, in accordance with the knowledge bases. Formally, π_θ is implemented as follows:

$$\begin{aligned} \pi_\theta(g, g_h, s_h|s) &= P(g, g_h, s_h, \theta_{sgf}, \theta_{hp}|s) \\ &= P(g|g_h;\theta_{sgf})P(g_h|s_h;\theta_{hp})P(s_h|s;\theta_{sgf}) \\ &\quad \cdot P(\theta_{sgf})P(\theta_{hp}). \end{aligned} \tag{5}$$

The right term in the second line can be derived by decomposing the joint probability in the first line, in accordance with the probabilistic dependency shown in Figure 2. Note that, in this equation, θ is represented as $\theta_{sgf}||\theta_{hp}$, i.e., a concatenation of θ_{sgf} and θ_{hp}. By using this representation for π_θ, we can derive an update expression, which can refines θ_{sgf} and θ_{hp} keeping them consistent with $P(\theta_{sgf})$ and $P(\theta_{hp})$. See Section 4.3 for details.

$P(\theta_{sgf})$ and $P(\theta_{hp})$ are needed to reflect the manually-designed knowledge bases. To do so, first, $P(\theta_{sgf})$ and $P(\theta_{hp})$ are implemented as parametric distributions $P(\theta_{sgf};\theta^{'}_{sgf})$ and $P(\theta_{hp};\theta^{'}_{hp})$, respectively, and their hyper-parameters $\theta^{'}_{sgf}$ and $\theta^{'}_{hp}$ are determined to replicate manually-designed symbol grounding functions and high-level planners. More formally, we use $\theta^{'*}_{sgf}$ and $\theta^{'*}_{sgf}$ as the optimal parameters of $\theta^{'}_{sgf}$ and $\theta^{'}_{hp}$, respectively, acquired by the following equations:

$$\theta^{'*}_{sgf} = \arg\min_{\theta^{'}_{sgf}} D_{sgf}\left(P(g|g_h;\theta_{sgf})P(\theta_{sgf};\theta^{'}_{sgf})\right), \quad (6)$$

$$\theta^{'*}_{hp} = \arg\min_{\theta^{'}_{hp}} D_{hp}\left(P(g_h|s_h;\theta_{hp})P(\theta_{hp};\theta^{'}_{hp})\right), \quad (7)$$

where D_{sgf} and D_{hp} are a divergence (e.g., KL divergence) from the manually-designed symbol grounding function and high-level planner, respectively. D_{sgf} and D_{hp} are abstract criteria, and thus, there are many implementations of functionals "$\arg\min_{\theta^{'}_{sgf}} D_{sgf}(\cdot)$" and "$\arg\min_{\theta^{'}_{hp}} D_{hp}(\cdot)$."

4.2 An Example of Model Implementation to Solve the Mountain Car Problem

We introduced an abstract model for symbol grounding and high-level planning with knowledge bases in the previous section. In this section, we provide an example of an implementation of the model to solve the mountain car problem.

First, S_h and A are implemented as follows:

$$S_h := \left\{\begin{array}{c}\text{Bottom\_of\_hills(Car), On\_right\_side\_hill(Car),}\\ \text{On\_left\_side\_hill(Car), At\_top\_of\_right\_side\_hill(Car)}\end{array}\right\} \quad (8)$$

$$A := [-1.0, 1.0]. \quad (9)$$

S_h is implemented in accordance with the knowledge shown in Table 2. A is implemented in accordance with the definition of actions to solve the mountain car problem, and represented as a set of values for the acceleration of the car.

Second, the probabilities of the modules in the hierarchical planner are implemented as follows:

$$P(s_h|s;\theta_{sgf}) := \frac{N(s|\mu_{s_h},\sigma_{s_h})}{\Sigma_{s^{'}_h\in S_h} N(s|\mu_{s^{'}_h},\sigma_{s^{'}_h})}, \quad (10)$$

$$P(g|g_h;\theta_{sgf}) := N(g|\mu_{g_h},\sigma_{g_h}), \quad (11)$$

$$P(g_h|s_h;\theta_{hp}) := \frac{\exp(\phi(s_h)\boldsymbol{w}^T_{g_h})}{\Sigma_{g^{'}_h\in S_h}\exp(\phi(s_h)\boldsymbol{w}^T_{g^{'}_h})}. \quad (12)$$

$P(s_h|s;\theta_{sgf})$ is implemented as the normalized likelihood of a normal distribution (Eq. (10)), and $P(g|g_h;\theta_{sgf})$ is implemented as a normal distribution (Eq. (11)). In Eq. (10) and Eq. (11), $N(s|\mu_{s_h},\sigma_{s_h})$ represents a normal distribution for s, which is parameterized by mean μ_{s_h} and standard deviation σ_{s_h}, s.t, $\forall s_h \in S_h$. Note that μ_{s_h} and σ_{s_h} are identical to μ_{g_h} and σ_{g_h}, respectively. $P(s_h|s;\theta_{sgf})$ is implemented as a softmax function (Eq. (12)). In Eq. (12), $\phi(s_h)$ is a base function that returns a one-hot vector $\in \{0,1\}^{|S_h|}$ in which only one element corresponding to the value of s_h is set to a value of 1, and the other elements are set to a value of 0. $\boldsymbol{w}_{g_h} \in \mathbb{R}^{|S_h|}$ is a weight vector, s.t., $\forall g_h \in S_h$. In this implementation, θ_{sgf} is a vector composed of μ_{s_h},σ_{s_h}, s.t., $s_h \in S_h$, and θ_{hp} is a vector composed of the set $\boldsymbol{w}_{g_h}$, s.t., $g_h \in S_h$. $P(a|s,g)$ and $P(s'|s,a)$ are implemented as deterministic functions, which represent the simulator of environment[2] and the model predictive controller.

Third, the reward function $R(s',t|s,o)$ is implemented as follows:

$$R(s',t|s,o) := \Sigma^t_{i=0}\gamma^i r(s_i,a_i), \quad (13)$$

$$r(s_i,a_i) = \begin{cases}100 & \text{(if car position in } s_i > 0.6)\\ -a_i & \text{(otherwise)}\end{cases} \quad (14)$$

where s_i and a_i are a state and a primitive action sampled from the environment i time steps later from the executing option o, respectively. Eq. (14) represents "low-level" reward $r(s_i)$, which is fed in accordance with a_i and the car position included in s_i.

Fourth, $P(\theta_{sgf};\theta^{'}_{sgf})$ and $P(\theta_{hp};\theta^{'}_{hp})$ are implemented as follows:

$$P(\theta_{sgf};\theta^{'}_{sgf}) := \prod_{s_h\in S_h} N(\mu_{s_h}|\mu^{'}_{s_h},1)\cdot Uni(\sigma_{s_h}), \quad (15)$$

$$P(\theta_{hp};\theta^{'}_{hp}) := \prod_{g_h\in S_h}\prod^{|s_h|}_{i=0} N(w_{g_h,i}|\mu^{'}_{w_{g_h,i}},1). \quad (16)$$

Eq. (15) represents a distribution for μ_{s_h} and σ_{s_h}. The component for μ_{s_h} is a normal distribution, which has mean $\mu^{'}_{s_h}$ and standard deviation 1, and the component for σ_{s_h} is a uniform distribution $Uni(\sigma_{s_h})$. In addition, Eq. (16) represents the normal distribution for $w_{g_h,i}$, which is the i-th element of $\boldsymbol{w}_{g_h}$. This distribution has mean $\mu^{'}_{w_{g_h,i}}$ and standard deviation 1. Note that, in this implementation, $\theta^{'}_{sgf}$ and $\theta^{'}_{hp}$ are $\mu^{'}_{s_h}$ and $\mu^{'}_{w_{g_h,i}}$, respectively.

Finally, functionals in Eq. (6) and Eq. (7), are implemented as follows:

Implementation of $\arg\min_{\theta^{'}_{sgf}} D_{sgf}(\cdot)$ **:** Using Eq. (15), $\mu^{'}_{s_h}$ is set as the mean of the corresponding interval, which is defined in the knowledge base for grounding functions. For example, $\mu^{'}_{\text{Bottom\_of\_hills(car)}}$ is determined as -0.5, the mean of $[-0.6,-0.4]$ in Table 2.

Implementation of $\arg\min_{\theta^{'}_{hp}} D_{hp}(\cdot)$**:** Using Eq. (7), $\mu^{'}_{w_{g_h}}$ is determined by Algorithm 1. The algorithm is outlined as follows: first initialize $\mu_{w_{g_h}}$ with val_{nin} (line 1–3), and if the operator, in which s_h refers to the preconditions and $s^{'}_h$ refers to the effects, is contained in knowledge base KB_{hp}, the corresponding weight is initialized with Val_{in} (line 4–11). KB_{hp} is initialized in accordance with Table 1 before it is passed to the algorithm.

[2]Open AI gym was used as the simulator: `https://github.com/openai/gym/wiki/MountainCarContinuous-v0`

Algorithm 1 Implementation of $\arg\min_{\theta'_{hp}} D_{hp}$

Require: The following variables are given:
(1) Set of abstract symbolic information S_h.
(2) Set of operators K_{hp} included in the knowledge base for the high-level planner. Each operator is represented as a tuple (precondition, effect).
(3) Set of hyper parameters $\mu'_{w_{g_h}}$ for all possible abstract symbolic information s_h.
(4) Weight value val_{in} to be assigned to the weight of an operator, which is included in the knowledge base.
(5) Weight value val_{nin} to be assigned to the weight of an operator, which is not included in the knowledge base.
(6) Index function $I(s_h)$ that maps s_h to index $i \in \mathbb{I}$. i used to access the element of $\mu'_{w_{g_h}}$.

for $s_h \in S_h$ **do**
  Initialize $\mu'_{w_{g_h}}$ with val_{nin}
end for
for $s_h \in S_h$ **do**
  $i \leftarrow 0$
  for $s'_h \in S_h$ **do**
    if $(s_h, s'_h) \in K_{hp}$ **then**
      $\mu'_{w_{g_h}, I(s'_h)} \leftarrow val_{in}$
    end if
  end for
end for

4.3 Refining Symbol Grounding and High-Level Planning with Policy Gradients

Refining the high-level planner $P(g_h|s_h;\theta_{hp})$ and symbol grounding functions ($P(s_h|s;\theta_{sgf})$ and $P(g|g_h;\theta_{sgf})$) is achieved by a parameter update in Eq. (17). This equation contains two unique terms: a **reinforcement term** and a **penalty term**. The reinforcement term contributes to updating the parameters to maximize the expected cumulative reward, as in standard reinforcement learning. The penalty term contributes to keeping the parameters consistent with the priors (i.e., manually-designed knowledge bases). This update is derived by substituting $\theta_{sgf}||\theta_{hp}$ and Eq. (5) for θ and Eq. (3), respectively. Using the example described in Section 4.2, μ_{s_h}, σ_{s_h} and $\boldsymbol{w}_{g_h}$ are updated in this equation. In this case, the penalty term prevents μ_{s_h} and $w_{g_h,i}$, for all s_h, g_h, and i, from moving far away from μ'_{s_h} and $\mu'_{w_{g_h},i}$, respectively.

5 Experiments

In this section, we perform an experimental evaluation to investigate whether the symbol grounding functions and high-level planner are refined successfully by using the framework we proposed in the previous section. In Section 5.1, we focus on the evaluation for refining the symbol grounding functions only. Then, in Section 5.2, we evaluate the effect of jointly refining symbol grounding functions and the high-level planner.

5.1 Refinement of Symbol Grounding

We evaluate how the symbol grounding functions are refined by our framework to solve the mountain car problem. The experimental set up to implement the planner and our framework is the same as that in the example introduced in Section 3 and Section 4.

For the evaluation, we prepared three types of method:

Baseline: A hierarchical planner that uses the grounding functions and a high-level planner, which are manually designed. This planner is identical to that introduced in the example in Section 3.

NoPenalty: The framework that refines the symbol grounding functions without the penalty term in Eq. (17). In this method, the high-level planner is the same as that in Baseline.

Proposed: The framework that refines the symbol grounding functions with the penalty term. In this method, the high-level planner is the same as that in Baseline.

These methods were evaluated on the basis of two criteria: an average cumulative reward over episodes, and a parameter divergence. The former is to evaluate if the hierarchical planner produces a more appropriate plan by refining its modules, and the latter is to evaluate the interpretability of the refined modules. The parameter divergence represents how much the policy's parameters (μ_{s_h})[3] refined by the framework differ from the initial parameters. In this paper, this divergence is measured by the Euclidean distance between the refined parameter (μ_{s_h}) and its initial parameter (μ'_{s_h}). Initial values for μ_{s_h} and σ_{s_h} are given, shown as "Init" in Table 3. μ_{s_h} is initialized with μ'_{s_h}, which is determined on the basis of the implementation of the functional in Eq. (6) (see Section 4.2), and σ_{s_h} is manually determined. We consider 50 episodes as one epoch and performed refinement over 2000 epochs.

The experimental results (shown in Figures 4 and 5) show that 1) refining the grounding functions improves the performance (average cumulative reward) of hierarchical planners, and 2) considering the penalty term keeps the refined parameters within a certain distance from the initial parameters. Regarding 1), Figure 4 shows the methods in which the grounding functions are refined (NoPenalty and Proposed) outperform Baseline. This result indicates the refinement for grounding functions successfully improves its performance. Regarding 2), Figure 5 shows that the parameter in NoPenalty moves away from the original parameter in refining, while in Proposed, the parameter stays close to the original one.

An example of the refined parameter for the grounding functions for Proposed is shown in Table 3, which indicates that the parameter is updated to achieve high-performance planning while staying close to the original parameter. In this example, the mean and standard deviation of "On_right_side_hill(Car)" is changed significantly through refinement. The mean for grounding On_right_side_hill(Car) is biased to a more negative position, and also flattened to make the car climb up the left side hill quickly (Figure 6). This change makes the symbol grounding function more flattened and considers the center position as

[3] We assume μ_{s_h} dominatingly determines the behaviors of symbol grounding functions.

$$\theta_{sgf}||\theta_{hp} \leftarrow \theta_{sgf}||\theta_{hp} + \alpha V_{\pi_{\theta_{sgf}||\theta_{hp}}}(\tilde{s_0})\{\underbrace{\nabla_{\theta_{sgf}||\theta_{hp}} \log P(\tilde{g_0}|g_{\tilde{h},0};\theta_{sgf})P(g_{\tilde{h},0}|s_{\tilde{h},0};\theta_{hp})P(s_{\tilde{h},0}|\tilde{s_0};\theta_{sgf})}_{\text{reinforcement term}} + \underbrace{\nabla_{\theta_{sgf}||\theta_{hp}} \log P(\theta_{sgf})P(\theta_{hp})}_{\text{penalty term}}\} \quad (17)$$

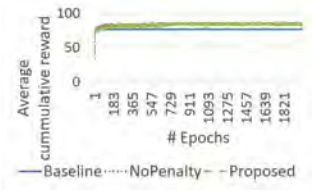


Figure 4: Learning curves for each methods. The vertical axis represents average cumulative rewards and the horizontal axis represents epochs (50 episodes for each epoch).

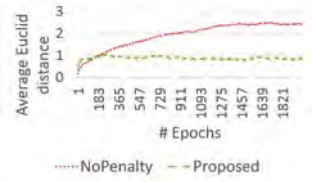


Figure 5: Parameter divergences. The vertical axis represents the average Euclidean distance and the horizontal axis represents learning epochs.

Table 3: Refined parameters.

μ_{s_h}	*Bottom_of_hills*	*At_top_of_right_side_hill*	*On_right_side_hill*	*On_left_side_hill*
Init	-0.5	0.6	0.2	-1.1
Refined	-0.5	0.46	**-0.39**	-1.1

σ_{s_h}	*Bottom_of_hills*	*At_top_of_right_side_hill*	*On_right_side_hill*	*On_left_side_hill*
Init	0.4	0.1	0.4	0.3
Refined	0.4	0.12	1.42	0.11

"On_right_side_hill(Car)." The main interpretation of this result is that the symbol grounding function was refined to reduce the redundancy in high-level planning. In the original symbol grounding functions, the center position is grounded to "Bottom_of_hills (Car)," and the high-level planner makes a plan [Bottom_of_hills(Car) → On_right_side_hill(Car) → On_left_side_hill(Car) → At_top_of_right_side_hill(Car)], which means "Starting at the bottom of hills, visit, in order, the right side hill, the left side hill, and the top of the right side hill." However, this plan is redundant; the car does not need to visit the right side hill first. The refined symbol grounding function considers the center position as "Right_side_hill(Car)," and thus the high-level planner produces the plan [Bottom_of_hills(Car) → On_right_side_hill(Car) → On_left_side_hill(Car) → At_top_of_right_side_hill(Car)], in which the redundancy is removed. It should also be noted that the order of the refined means is intuitively correct. For example, the value of $\mu_{\text{On\_right\_side\_hill}}$ is higher than the value of $\mu_{\text{On\_left\_side\_hill}}$ (i.e., $\mu_{\text{On\_right\_side\_hill}}$ means the place on more right-side than $\mu_{\text{On\_left\_side\_hill}}$). It cannot be seen in the Baseline and NoPenalty cases. This result supports the fact that our framework refines the modules by maintaining their interpretability.

5.2 Joint Refinement of Symbol Grounding and High-Level Planning

In this section, we refine both the symbol grounding functions and the high-level planner. The setup of the hierarchical planner and the problem are the same as those of the previous section, except for the knowledge base for the high-level planner. We removed "Opr.2" (as shown in Table 1) and used this degraded version as the knowledge base for the experiment.

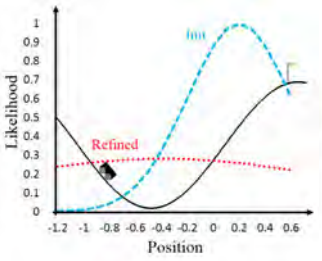


Figure 6: Example of refining result symbol grounding function for *On_right_side_hill*.

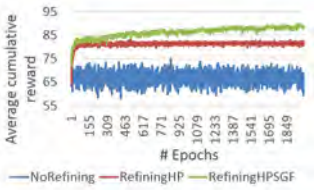


Figure 7: Learning curve.

This degradation makes a space for refining the knowledge base for the high-level planner. In addition, we put a small coefficient of the penalty term for $\boldsymbol{w}_{g_h}$, because we found that considering this term too much makes the refinement worse in a preliminary experiment. As long as the results of the symbol grounding functions are interpretable, the result of the high-level planner is interpretable as well. $\boldsymbol{w}_{g_h}$ is initialized with $w'_{g_h,i}$, which is determined by (i.e., Algorithm 1) where we set -1.3 as val_{in}, and -0.02 as val_{nin}. The resulting $w'_{g_h,i}$ is shown as "Init" in Table 4.

We prepared three types of methods:

NoRefining: A hierarchical planner with the degraded version of the knowledge base for high-level planner. The knowledge base for the symbol grounding function is the same to that shown in Table 2.

RefiningHP: The framework that refines the high-level planner only. In this method, symbol grounding functions are the same as those in NoRefining.

RefiningHPSGF: The framework that refines both symbol grounding functions and the high-level planner.

From the experimental result (Figure 7), we can confirm that our framework successfully refines both symbol grounding functions and the high-level planner, from the viewpoint of performance. RefiningHP outperforms NoRefining, and RefiningHPSGF outperforms the other methods.

Table 4 provides an example of how the high-level planner was refined. It indicates that the dropped knowledge (i.e., Opr. 2) was successfully acquired in refinement, and knowledge is discovered that makes high-level planning more efficient. Considering the form of Eq. (12), the operator, which corresponds to the element of a weight with a higher value, contributes more to high-level planning. Therefore, these corresponding operators are worthwhile as knowledge for high-level planning. In Table 4, the refined weight of the operator (preconditions=On_right_side_hill, effects=On_left_side_hill) is higher than those of other operators in which the precondition contains On_right_side_hill. This operator was once initially removed and later acquired by the refinement. Similarly, the operator (preconditions=Bottom_of_hills, effects=On_left_side_hill), which is not shown in Table 1, was newly acquired.

Table 4: Example of high-level planner improvement. Refined weights $\boldsymbol{w}_{g_h}$ are shown for each precondition (column) and effect (row). Initial weights are shown in parentheses.

Refined (Init)	Bottom_of_hills	At_top_of_right_side_hill	On_right_side_hill	On_left_side_hill
Bottom_of_hills	-5.88 (-1.3)	-6.34 (-1.3)	-3.15 (-1.3)	-6.65 (-1.3)
At_top_of_right_side_hill	-9.04 (-1.3)	-9.75 (-1.3)	-4.76 (-1.3)	**2.5** (-0.02)
On_right_side_hill	-0.98 (-0.02)	**1** (-1.3)	-2.03 (-1.3)	-1.34 (-1.3)
On_left_side_hill	**0.85** (-1.3)	-2.12 (-1.3)	**1.74** (-1.3)	-11.71 (-1.3)

6 Conclusion

In this paper, we proposed a framework that refines manually-designed symbol grounding functions and a high-level planner. Our framework refines these modules with policy gradients. Unlike standard policy gradient implementations, our framework additionally considers the penalty term to keep parameters close to the prior parameter derived from manually-designed modules. Experimental results showed that our framework successfully refined the parameters for the modules; it improves the performance (cumulative reward) of the hierarchical planner, and keeps the parameters close to those derived from the manually-designed modules.

One of the limitations of our framework is that it deals only with predefined symbols (such "Bottom_of_hills"), and it does not discover new symbols. We plan to address this drawback in future work. We also plan to evaluate our framework in a more complex domain where primitive actions and states are high-dimensional, and the knowledge base is represented in a more complex description (e.g., precondition contains multiple states).

References

[Camacho and Alba, 2013] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[Cambon *et al.*, 2009] Stéphane Cambon, Rachid Alami, and Fabien Gravot. A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126, 2009.

[Choi and Amir, 2009] Jaesik Choi and Eyal Amir. Combining planning and motion planning. In *Proc. of ICRA-09*, pages 238–244. IEEE, 2009.

[Das *et al.*, 2017] Abhishek Das, Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. arXiv:1703.06585, 2017.

[Dornhege *et al.*, 2009] Christian Dornhege, Marc Gissler, Matthias Teschner, and Bernhard Nebel. Integrating symbolic and geometric planning for mobile manipulation. In *Proc. of SSRR-09*, pages 1–6. IEEE, 2009.

[Fdez-Olivares *et al.*, 2011] Juan Fdez-Olivares, Luis Castillo, Juan A Cózar, and Oscar García Pérez. Supporting clinical processes and decisions by hierarchical planning and scheduling. *Computational Intelligence*, 27(1):103–122, 2011.

[Kaelbling and Lozano-Pérez, 2011] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *Proc. of ICRA-11*, pages 1470–1477. IEEE, 2011.

[Kakade, 2002] Sham M Kakade. A natural policy gradient. In *Proc. of NIPS-02*, pages 1531–1538, 2002.

[Konidaris *et al.*, 2014] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. In *Proc. of AAAI-14*, 2014.

[Konidaris *et al.*, 2015] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Symbol acquisition for probabilistic high-level planning. In *Proc .of IJCAI-15*, 2015.

[Konidaris, 2016] George Konidaris. Constructing abstraction hierarchies using a skill-symbol loop. In *Proc. of IJCAI-16*, 2016.

[Malcolm and Smithers, 1990] Chris Malcolm and Tim Smithers. Symbol grounding via a hybrid architecture in an autonomous assembly system. *Robotics and Autonomous Systems*, 6(1-2):123–144, 1990.

[McDermott *et al.*, 1998] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL-the planning domain definition language. 1998.

[Moore, 1991] Andrew Moore. Efficient memory-based learning for robot control. March 1991.

[Nilsson, 1984] Nils J Nilsson. Shakey the robot. Technical report, SRI INTERNATIONAL MENLO PARK CA, 1984.

[Özdamar *et al.*, 1998] Linet Özdamar, M Ali Bozyel, and S Ilker Birbil. A hierarchical decision support system for production planning (with case study). *European Journal of Operational Research*, 104(3):403–422, 1998.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proc. of ICML-15*, pages 1889–1897, 2015.

[Silver *et al.*, 2014] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proc. of ICML-14*, 2014.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[Sutton *et al.*, 1999] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[Sutton *et al.*, 2000] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proc. of NIPS-00*, pages 1057–1063, 2000.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

[Wolfe *et al.*, 2010] Jason Andrew Wolfe, Bhaskara Marthi, and Stuart J Russell. Combined task and motion planning for mobile manipulation. In *Proc. of ICAPS-10*, 2010.

Inductive Logic Programming, Ontology Reasoning, and Spatial Knowledge: A Short Survey of 15-Years Research*

Francesca A. Lisi
Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro", Italy
FrancescaAlessandra.Lisi@uniba.it

Abstract

This short paper overviews 15 years of work by the author at the intersection between the two AI areas of Machine Learning and Knowledge Representation. The distinguishing feature of her research has been the extension of the methodological apparatus of Inductive Logic Programming along a couple of directions towards Ontology Reasoning. The former was concerned with learning hybrid rules tightly integrating Datalog and Description Logics, whereas the latter was concerned with learning axioms in fuzzy Description Logics. Both turned out to be alternative suitable ways to treat spatial knowledge in several applications.

1 Introduction

Inductive Logic Programming (ILP) [Muggleton, 1990] was born at the intersection between Logic Programming (LP) [Lloyd, 1987] and Concept Learning [Mitchell, 1982]. It provides a bunch of techniques for structuring, searching, and bounding the space of hypotheses represented as Horn clauses [Nienhuys-Cheng and de Wolf, 1997]). ILP has been historically concerned with learning Horn rules from examples and background knowledge with the aim of prediction (see, *e.g.*, the system FOIL [Quinlan, 1990]). However, ILP has also been applied to tasks - such as association rule mining - other than classification where the scope of induction is description rathen than prediction. A notable example of this kind of ILP systems is WARMR [Dehaspe and Toivonen, 1999] which mines frequent DATALOG queries.

With the advent of the Semantic Web new challenges and opportunities have been presented to ILP. In particular, ontologies and their logical foundations in the family of Description Logics (DLs) [Baader *et al.*, 2003] raised several issues for the direct application of existing ILP systems, thus urging the extension and/or adaptation of the ILP methodological apparatus to the novel context.

In the following two sections I will survey my work in ILP over the past 15 years, first on learning so-called onto-relational rules (Section 2) and later on learning fuzzy ontology axioms (Section 3).

*I would like to thank all my coauthors of the papers cited here.

2 Learning onto-relational rules with ILP

LP and DLs are both based on fragments of First Order Logic (FOL). However, they are characterized by different semantic assumptions [Motik and Rosati, 2010]. Though a partial overlap exists between LP and DLs, even more interesting is a combination of the two via several integration schemes that are aimed at designing very expressive FOL languages and ultimately overcoming the aforementioned semantic mismatch (see, *e.g.*, [Drabent *et al.*, 2009] for a survey). A popular example of this class of hybrid KR formalisms is $\mathcal{AL}$-LOG [Donini *et al.*, 1998] which tightly integrates DATALOG and $\mathcal{ALC}$. Several works in ILP testify the great potential of these formalisms also from the perspective of machine learning and inductive reasoning [Rouveirol and Ventos, 2000; Kietz, 2003; Lisi, 2008; 2010; 2014]. Originally motivated by a spatial data mining application [Appice *et al.*, 2003; Lisi and Malerba, 2004] and inspired by WARMR, $\mathcal{AL}$-QUIN [Lisi, 2011] is an ILP system for mining association rules at multiple levels of granularity by performing taxonomic reasoning in the KR framework of $\mathcal{AL}$-LOG.

3 Learning fuzzy ontology axioms with ILP

Spatial notions such as the distance between two sites can be naturally represented with fuzzy sets if one is interested in their human perception rather than in precise measurements. In order to deal with imprecision in Ontology Reasoning several fuzzy extensions of DLs have been proposed (see, *e.g.*, [Straccia, 2015] for an overview). However, the problem of automatically managing the evolution of fuzzy DL ontologies still remains relatively unaddressed [Konstantopoulos and Charalambidis, 2010; Iglesias and Lehmann, 2011]. Lisi and Straccia [2013] propose *Soft*FOIL, a FOIL-like method for learning fuzzy $\mathcal{EL}$ GCI axioms from fuzzy DL assertions. In [Lisi and Straccia, 2014], the same authors present FOIL-$\mathcal{DL}$, another FOIL-like method which, conversely, is designed for learning fuzzy $\mathcal{EL}(\mathbf{D})$ GCI axioms from crisp DL assertions. As opposite to *Soft*FOIL, FOIL-$\mathcal{DL}$ has been implemented and tested [Lisi and Straccia, 2015], notably in a real-world tourism application. More recently, a granular computing method for OWL 2 ontologies has been proposed in [Lisi and Mencar, 2018] with the ultimate goal of optimizing the learning process when dealing with a huge number of relations, *e.g.*, those concerning the distance between places.

References

[Appice *et al.*, 2003] Annalisa Appice, Michelangelo Ceci, Antonietta Lanza, Francesca A. Lisi, and Donato Malerba. Discovery of spatial association rules in geo-referenced census data: A relational mining approach. *Intelligent Data Analysis*, 7(6):541–566, 2003.

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[Dehaspe and Toivonen, 1999] L. Dehaspe and H. Toivonen. Discovery of frequent DATALOG patterns. *Data Mining and Knowledge Discovery*, 3:7–36, 1999.

[Donini *et al.*, 1998] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[Drabent *et al.*, 2009] Wlodzimierz Drabent, Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner, Thomas Lukasiewicz, and Jan Maluszynski. Hybrid Reasoning with Rules and Ontologies. In François Bry and Jan Maluszynski, editors, *Semantic Techniques for the Web, The REWERSE Perspective*, volume 5500 of *Lecture Notes in Computer Science*, pages 1–49. Springer, 2009.

[Iglesias and Lehmann, 2011] Josue Iglesias and Jens Lehmann. Towards integrating fuzzy logic capabilities into an ontology-based inductive logic programming framework. In *Proc. of the 11th Int. Conf. on Intelligent Systems Design and Applications*. IEEE Press, 2011.

[Kietz, 2003] Jörg-Uwe Kietz. Learnability of description logic programs. In Stan Matwin and Claude Sammut, editors, *Inductive Logic Programming, 12th International Conference, ILP 2002, Sydney, Australia, July 9-11, 2002. Revised Papers*, volume 2583 of *Lecture Notes in Computer Science*, pages 117–132. Springer, 2003.

[Konstantopoulos and Charalambidis, 2010] S. Konstantopoulos and A. Charalambidis. Formulating description logic learning as an inductive logic programming task. In *Proc. of the 19th IEEE Int. Conf. on Fuzzy Systems*, pages 1–7. IEEE Press, 2010.

[Lisi and Malerba, 2004] Francesca A. Lisi and Donato Malerba. Inducing Multi-Level Association Rules from Multiple Relations. *Machine Learning*, 55:175–210, 2004.

[Lisi and Mencar, 2018] Francesca A. Lisi and Corrado Mencar. A granular computing method for OWL ontologies. *Fundamenta Informaticae*, 159(1–2):147–174, 2018.

[Lisi and Straccia, 2013] Francesca A. Lisi and Umberto Straccia. A logic-based computational method for the automated induction of fuzzy ontology axioms. *Fundamenta Informaticae*, 124(4):503–519, 2013.

[Lisi and Straccia, 2014] Francesca Alessandra Lisi and Umberto Straccia. A FOIL-like Method for Learning under Incompleteness and Vagueness. In Gerson Zaverucha, Vítor Santos Costa, and Aline Paes, editors, *Inductive Logic Programming - 23rd International Conference, ILP 2013, Rio de Janeiro, Brazil, August 28-30, 2013, Revised Selected Papers*, volume 8812 of *Lecture Notes in Computer Science*, pages 123–139. Springer, 2014.

[Lisi and Straccia, 2015] Francesca A. Lisi and Umberto Straccia. Learning in description logics with fuzzy concrete domains. *Fundamenta Informaticae*, 140(3-4):373–391, 2015.

[Lisi, 2008] Francesca A. Lisi. Building Rules on Top of Ontologies for the Semantic Web with Inductive Logic Programming. *Theory and Practice of Logic Programming*, 8(03):271–300, 2008.

[Lisi, 2010] Francesca A. Lisi. Inductive Logic Programming in Databases: From Datalog to $\mathcal{DL}$+log. *Theory and Practice of Logic Programming*, 10(3):331–359, 2010.

[Lisi, 2011] Francesca A. Lisi. $\mathcal{AL}$-QUIN: An Onto-Relational Learning System for Semantic Web Mining. *International Journal on Semantic Web and Information Systems*, 7(3):1–22, 2011.

[Lisi, 2014] Francesca A. Lisi. Learning onto-relational rules with inductive logic programming. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, volume 18 of *Studies on the Semantic Web*, pages 93–111. IOS Press/AKA, 2014.

[Lloyd, 1987] John W. Lloyd. *Foundations of Logic Programming*. Springer, 2nd edition, 1987.

[Mitchell, 1982] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[Motik and Rosati, 2010] Boris Motik and Riccardo Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.

[Muggleton, 1990] Stephen H. Muggleton. Inductive logic programming. In S. Arikawa, S. Goto, S. Ohsuga, and T. Yokomori, editors, *Proceedings of the 1st Conference on Algorithmic Learning Theory*. Springer/Ohmsma, 1990.

[Nienhuys-Cheng and de Wolf, 1997] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.

[Quinlan, 1990] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Rouveirol and Ventos, 2000] Céline Rouveirol and Véronique Ventos. Towards Learning in CARIN-$\mathcal{ALN}$. In James Cussens and Alan M. Frisch, editors, *Inductive Logic Programming, 10th International Conference, ILP 2000, London, UK, July 24-27, 2000, Proceedings*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 191–208. Springer, 2000.

[Straccia, 2015] Umberto Straccia. All about fuzzy description logics and applications. In Wolfgang Faber and Adrian Paschke, editors, *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures*, volume 9203 of *Lecture Notes in Computer Science*, pages 1–31. Springer, 2015.

Spatio-Temporal Awareness for Wireless Telecommunication Networks

H. Joe Steinhauer, Tove Helldin, Gunnar Mathiason
School of Informatics, University of Skövde, Sweden
{joe.steinhauer, tove.helldin, gunnar.mathiason}@his.se

Abstract

The processes within a wireless telecommunication system are spatio-temporal and context dependent. For example, the root cause of a failure needs to be identified in terms of where, when and why it happened. To identify and resolve errors in the system, human telecommunication operators need to be assisted by decision support tools which the operators can understand, trust, and where they can input their expert knowledge. In this paper we describe our ongoing work and some of the challenges we are facing when designing such tools.

1 Introduction

To maintain a high quality of service is crucial for wireless telecommunication networks. However, identifying degradations of service in these complex systems, before they manifest themselves into noticeable problems, is a challenging task that is part of the everyday work of human telecommunication operators. Since each telecommunication base station keeps track of a multitude of different network runtime variables, data is available for automatic process recognition and anomaly detection. Hence, telecommunication companies are to date investigating how to incorporate machine learning into a decision support system that aids the human operator to monitor and analyze the telecommunication systems.

The processes within a telecommunication system are highly context sensitive, however, typical patterns can be observed in the data. Yet, what can be regarded as normal network traffic depends on many context variables (e.g. time of the day, day of the week, season, weather conditions, social events such as festivals, road conditions such as traffic jams, etc.). To understand if the current behavior of the network is normal or abnormal and, in case of the latter, if it is in need of intervention is difficult to determine. A good understanding of how telecommunication networks and processes work, together with detailed knowledge of the particular network at hand and its surrounding context variables is necessary for efficient and accurate network monitoring and diagnosis. However, with the increase of network load and complexity, semi-automated tools are needed to support the operators with keeping the telecommunication systems functioning with good quality.

Our ongoing work consists of developing a transparent decision support system for monitoring a wireless telecommunication system where the human operator and the system work together as a team. Expert and domain knowledge as well as statistical correlations in the telecommunication data need to be brought together in order to build a good situation awareness of such a complex system and to monitor and maintain it successfully. This includes reasoning in space and time and the presentation of spatio-temporal information in a way that alleviates the team reasoning approach between the system and the human operator.

2 Previous work

In our previous research, we applied machine learning and exploratory data analysis tools to extract useful information from the network data. In particular, we used Restricted Boltzmann Machines (RBM) [Smolensky, 1986] for root cause localization of failures [Steinhauer *et al.*, 2016] and topic modeling [Blei, 2012] for performance monitoring [Steinhauer *et al.*, 2017; Helldin *et al.*, 2018] and anomaly detection [Steinhauer *et al.*, 2018]. For topic modeling we used the latent Dirichlet allocation (LDA) for exploratory data analysis, as described in [Helldin *et al.*, 2018]. We also used the LDAvis visualization [Sievert and Shirley, 2014] for validation of the model together with a domain expert.

The results obtained so far are promising. In case of topic modeling for performance monitoring and anomaly detection, the domain expert could interpret the statistical correlations found in the data as corresponding to causal dependencies within the telecommunication processes. This interpretation comes, of course, with the risk of confirmation bias. In general, statistical relationships are difficult for humans to interpret and fuse correctly with their expert knowledge, hence, our work has also focused on how to present the results from the statistical analyses to a human operator. For example, we have designed a dashboard visualization addressing the various information needs of the operators [Helldin *et al.*, 2018]. Following the design framework proposed by [Koh *et al.*, 2011], where early and iterative prototyping together with domain experts is advocated for, we have continuously evaluated our results as well as have given the experts a chance of understanding what information can be extracted from the models.

LDAvis is one of the visualization tools used, however, as

described in [Smith *et al.*, 2017] the results of a topic model do not necessarily provide meaning to the analysts and manual interpretation together with experts is needed. Several ways of visualizing topic models have been suggested, where the most common representations are either graph [Gretarsson *et al.*, 2012], matrix or text based [Chuang *et al.*, 2012; Chaney and Blei, 2012]. Especially interesting for our future work are approaches that focus more on identifying thematic changes over time (e.g. [Havre *et al.*, 2002]). However, the choice of visualizations needs to be carefully evaluated together with novice and experienced operators.

3 Challenges

The temporal, spatial and contextual relationships that occur in wireless telecommunication networks are manifold. An anomaly is most often not detected or even noticeable until it has proliferated within the network, and its cause and/or location can be difficult to pinpoint. The anomaly might not manifest itself at one specific time point or place, instead it can, for example, appear as a slight elevation of certain variables distributed geographically over several base stations. Furthermore, many so called anomalies correspond in fact, if context variables are taken into account, to normal network behavior. For example, a base station undergoing an update will show a decrease in performance. In the context of the update, this behavior is to be considered normal and even indicating a well-functioning network. However, if no update is underway, this might indicate a serious problem which needs intervention.

Today, key performance indicators (KPIs) are used to measure important performance aspects which alert the operator if the measurements reach certain thresholds. However, due to situations like the one described above, KPIs need to be robust enough to not send false positive alerts (as in case of the update). Unfortunately, this robustness has downsides, such as decreased sensitivity when it comes to the early detection of trends in the network.

In our work, there are several challenges to be faced. One is to build models for normal network traffic that capture the development of network behavior over time, space (geographical as well as conceptual) and context. There are several approaches to investigate that have a temporal component, for example, time series analysis, evolutionary topic modeling and recurrent neural networks (RNN) with long short term memory (LSTM). The second challenge is hence to include spatial, contextual and expert knowledge into the model. Besides that, a third challenge lies in being able to convey the information needed to the human operators and at the same time allow them to follow their own chain of reasoning when investigating the network problems as well as add their expert knowledge into the reasoning and analysis performed.

References

[Blei, 2012] David M. Blei. Probabilistic Topic Models. *Communications of the ACM*, 55(4):77–84, April 2012.

[Chaney and Blei, 2012] Allison June-Barlow Chaney and David M Blei. Visualizing topic models. In *6th Inter. AAAI Conf. on Weblogs and Social Media*, 2012.

[Chuang *et al.*, 2012] Jason Chuang, Christopher D Manning, and Jeffrey Heer. Termite: Visualization techniques for assessing textual topic models. In *Proc. of the Inter. Working Conf. on Advanced Visual Interfaces*, pages 74–77. ACM, 2012.

[Gretarsson *et al.*, 2012] Brynjar Gretarsson, John Odonovan, Svetlin Bostandjiev, Tobias Höllerer, Arthur Asuncion, David Newman, and Padhraic Smyth. Topicnets: Visual analysis of large text corpora with topic modeling. *ACM Trans. on Intelligent Systems and Technology (TIST)*, 3(2):23, 2012.

[Havre *et al.*, 2002] Susan Havre, Elizabeth Hetzler, Paul Whitney, and Lucy Nowell. Themeriver: Visualizing thematic changes in large document collections. *IEEE Trans. on Visualization and Computer Graphics*, 8(1):9–20, 2002.

[Helldin *et al.*, 2018] T. Helldin, H. J. Steinhauer, A. Karlsson, and G. Mathiason. Situation awareness in telecommunication networks using topic modeling. In *21st Inter. Conf. on Information Fusion*, 2018.

[Koh *et al.*, 2011] Lian Chee Koh, Aidan Slingsby, Jason Dykes, and Tin Seong Kam. Developing and applying a user-centered model for the design and implementation of information visualization tools. In *15th Inter. Conf. on Information Visualisation*, pages 90–95. IEEE, 2011.

[Sievert and Shirley, 2014] Carson Sievert and Kenneth E Shirley. LDAvis: A method for visualizing and interpreting topics. In *Proc. of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70, 2014.

[Smith *et al.*, 2017] Alison Smith, Tak Yeon Lee, Forough Poursabzi-Sangdeh, Jordan Boyd-Graber, Niklas Elmqvist, and Leah Findlater. Evaluating visual representations for topic understanding and their effects on manually generated labels. *Trans. of the Association for Computational Linguistics*, 5:1–15, 2017.

[Smolensky, 1986] P. Smolensky. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. pages 194–281. MIT Press, Cambridge, MA, USA, 1986.

[Steinhauer *et al.*, 2016] H. Joe Steinhauer, A. Karlsson, G. Mathiason, and T. Helldin. Root-cause localization using restricted boltzmann machines. In *19th Inter. Conf. on Information Fusion (FUSION)*, pages 248–255, 2016.

[Steinhauer *et al.*, 2017] H. J. Steinhauer, T. Helldin, A. Karlsson, and G. Mathiason. Topic modeling for situation understanding in telecommunication networks. In *Inter. Telecommunication Network and Application Conference*, pages 1–6. IEEE, 2017.

[Steinhauer *et al.*, 2018] H. Joe Steinhauer, T. Helldin, G. Mathiason, and A. Karlsson. Anomaly detection in telecommunication networks using topic models. In *Submitted to the 15th Inter. Conf. on Modeling Decisions for Artificial Intelligence (MDAI18)*, 2018.